Coling 2010

23rd International Conference on Computational Linguistics

Proceedings of the Conference
Volume 2

Chu-Ren Huang and Dan Jurafsky

23 – 27 August 2010
Beijing International Convention Center
Beijing, China

Published by
Tsinghua University Press
All rights reserved for hard copy production.
Block A, Xue Yan Building, Tsinghua University
Beijing, 100084
China

To order hard copies of this proceedings, contact:

Chinese Information Processing Society of China
No.4, Southern Fourth Street
Haidian District, Beijing, 100190
China
Tel: +86-010-62562916
Fax: +86-010-62562916
cips@iscas.ac.cn

Sponsorship

The COLING 2010 very gratefully acknowledges the following commitments in sponsorship:

Platinum Sponsors

-National Natural Science Foundation of China

-Department of Language Information Administration, Ministry of Education, PRC

Gold Sponsor

BaiDu

Silver Sponsors

Google

Fujitsu R&D Center CO., LTD.

Microsoft Research

Beijing TRS Information Technology Co., Ltd

Shenyang Globla Envoy software Co.,Ltd.

Supporters

Asian Federation of Natural Language Processing

Institute of Automation
Chinese Academy of Sciences

Institute of Computing Technology
Chinese Academy of Sciences

Institute of Software
Chinese Academy of Sciences

Harbin Institute of Technology

Peking University

Tsinghua University

Preface

You will find in this volume papers from the 23rd International Conference on Computational Linguistics (COLING 2010) held in Beijing, China on August 23-27, 2010 under the auspices of the International Committee on Computational Linguistics (ICCL), and organized by the Chinese Information Processing Society (CIPS) of China. For this prestigious natural language processing conference to be held in China is a significant event for computational linguistics and for colleagues in China, demonstrating both the maturity of our field and the development of academic areas in China.

COLING started as a friendly gathering in New York in 1965, and has grown steadily since. Yet COLINGs aspiration to be a different conference remains the same. COLING strives to maintain its key qualities of embracing different theories and encouraging young scholars in spite of its growing size. A new component introduced at COLING 2010 underlines this quality. A RefreshINGenious (RING) session, organized by Aravind Joshi, our General Chair, allows new and un-orthodox ideas to be presented before they are fully developed in order to generate more discussion and stimulate other new ideas. We hope that this can become an important feature of COLING in the future.

The 155 oral papers included in the hardcopy proceedings published by Tsinghua University Press, as well as the 334 papers included in the electronic proceedings (the same 155 oral papers plus 179 poster papers) are selected from among 815 effective submissions among the more than 840 submissions received. The very selective acceptance rate of 19.02% for oral presentations (155/815 submissions) indicates the extremely high quality of the papers. An additional 21.96% (179/815) are selected for poster presentations to bring the overall acceptance rate to 40.98% (334/815).

We would like to thank the program committee area chairs for their dedicated and efficient review work, and our 738 reviewers for giving us very high quality reviews with a very short turnaround time, allowing us to maintain both the review quality and schedule even given the extraordinary number of submissions. Of course we thank the authors of the 840 papers for submitting their labor of love to COLING. Although we were only able to accept a minority of the submitted papers, we do hope that all authors and reviewers benefit from this process of indirect dialogue. We are especially grateful to the incredibly hard-working team of Stanford volunteers Jenny Finkel, Adam Vogel, and Mengqiu Wang, and HIT volunteers Sam Liang and Lemon Liu, who provided timely and efficient support for the two program chairs at every step of the review and publication processes.

Last but not least, we would like to thank the people who made COLING 2010 and this volume possible. We thank local arrangement committee co-chairs Professor Chengqing Zong and Professor Le Sun for their tireless work which will make COLING-2010 a sure success. Our special appreciation goes to the Chinese Information Processing Society (CIPS) and Professor Youqi Cao for their generous support as the COLING 2010 organizer. Lastly, Professor Qin Lu and Professor Tiejun Zhao should be recognized for their meticulous preparation for editing and publication, which brought this volume to reality.

Chu-Ren Huang and Dan Jurafsky,
COLING 2010 Program Committee Co-chairs

July 8, 2010

COLING 2010 is organized by the Chinese Information Processing Society of China (**CIPS**) and under the auspices of the International Committee on Computational Linguistics (**ICCL**).

General Chair:

Aravind K .Joshi (University of Pennsylvania)

Program Chairs:

Chu-Ren Huang (The Hong Kong Polytechnic University)
Dan Jurafsky (Stanford University)

Advisors to Organizing Committee:

Youqi Cao (The Chinese Information Processing Society of China)
Zhendong Dong (The Chinese Information Processing Society of China)
Changning Huang (Microsoft Research Asia)
Sheng Li (Harbin Institute of Technology)
TianshunYao (Northeastern University)
Shiwen Yu (Peking University)
Zhiwei Feng (Institute of Applied Linguistics, Ministry of Education)
Kaiying Liu (Shanxi University)

OrganizationChairs:

Chengqing Zong (Institute of Automation, Chinese Academy of Sciences)
Le Sun (Institute of Software, Chinese Academy of Sciences)

Publication Chairs:

Qin Lu (The Hong Kong Polytechnic University)
Tiejun Zhao (Harbin Institute of Technology)

Tutorial Chairs:

Dan Gildea (University of Rochester)
Xuanjing Huang (Fudan University)

Workshop Chairs:

Noah Smith (Carnegie Mellon University)
Takenobu Tokunaga (Tokyo Institute of Technology)
Haifeng Wang (BaiDu)

Publicity Chairs:

Hal Daumé III (University of Utah)
Bin Wang (Institute of Computing Technology, Chinese Academy of Sciences)
Minghui Dong (Institute for Infocomm Research)
Monica Monachini (Institute for Computational Linguistics)

Aline Villavicencio (Federal University of Rio Grande do Sul)

Sponsorship Chairs:

Tingting He (Huazhong Normal University)
Hinrich Schütze (University of Stuttgart)
Key-Sun Choi (Korea Advanced Institute of Science and Technology)
Rion Snow (Stanford University)
Takehito Utsuro (University of Tsukuba)
Renata Vieira (Pontifical Catholic University of Rio Grande do Sul)
Hao Yu (Fujitsu(China) R&D Center)

Demo Chairs:

Ting Liu (Harbin Institute of Technology)
Yang Liu (The University of Texas at Dallas)

Program Committee Members/AreaChairs:

Nianwen Xue (Brandeis University)
Rajeev Sangal (India Institute of Information Technology)
Roger Levy (University of California, San Diego)
Justine Cassell (Northwestern University)
Caroline Sporleder (Saarland University)
Gary Geunbae Lee (Pohang University of Science and Technology)
Rosie Jones (Yahoo! Research)
Nicoletta Calzolari (Istituto di Linguistica Computazionale)
Chris Callison-Burch (Johns Hopkins University)
Qun Liu (Institute of Computing Technology)
Pierre Isabelle (National Research Council of Canada)
Sadao Kurohashi (Kyoto University)
Sebastian Padó (Universität Stuttgart)
Takenobu Tokunaga (Tokyo Institute of Technology)
Bo Pang (Yahoo! Research)
Haizhou Li (Institute for Infocomm Research)
David A Smith (University of Massachusetts, Amherst)
Donia Scott (University of Sussex)
Emily Bender (University of Washington)
Ming Zhou (Microsoft Research Asia)

Organization Committee Members:

Juanzi Li (Tsinghua University)
KWONG Olivia (Hong Kong City University)
Bin Sun (Peking University)
Houfeng Wang (Peking University)
Xiaojie Wang (Beijing University of Posts and Telecommunications)
Endong Xun (Beijing Language and Culture University)
Erhong Yang (Beijing Language and Culture University)
Jun Zhao (Institute of Automation, Chinese Academy of Sciences)
Jingbo Zhu (Northeastern University)

Program Committee Members/Reviewers:

Rob Abbott
Takeshi Abekawa
Omri Abend
Steven Abney
Eytan Adar
Guadalupe Aguado
Khalid Al-Kofahi
Cecilia Ovesdotter Alm
Omar Alonso
Bharat Ram Ambati
Massih-Reza Amini
Xiangdong An
Sophia Ananiadou
Jaime Arguello
Masayuki Asahara
Nicholas Asher
Tania Avgustinova
Necip Fazil Ayan
Lakshmi Bai
Jason Baldridge
Timothy Baldwin
Carmen Banea
Srinivas Bangalore
Colin Bannard
Ken Barker
Marco Baroni
Regina Barzilay
John Bateman
Beata Beigman Klebanov
Daisuke Bekki
Nria Bel
Yinon Bentor
Sabine Bergler
Aditya Bhargava
Pushpak Bhattacharyya
Timothy Bickmore
Klinton Bicknell
Alexandra Birch
Philippe Blache
Alan Black
John Blitzer
Michael Bloodgood
Phil Blunsom
Ondrej Bojar
Gemma Boleda
Francis Bond
Johan Bos
Marisa Boston
Pierrette Bouillon
Julien Bourdaillet
Jordan Boyd-Graber
S. R. K. Branavan
Antonio Branco
Eric Breck
Chris Brew
Sabine Buchholz
Stefan Buettcher
Hung Bui
Razvan Bunescu
Aljoscha Burchardt
Stephan Busemann
Miriam Butt
William Byrne
Lynne Cahill
Jamie Callan
Bin Cao
Yunbo Cao
Guiseppe Carenini
Jean Carletta
Marine Carpuat
Xavier Carreras
John Carroll
Ben Carterette
Francisco Casacuberta
Steve Cassidy
Eric Castelli
Alexandru Ceausu
Nick Cercone
Jeong-Won Cha
Vineet Chaitanya
Yllias Chali
Nate Chambers
Baobao Chang
Pi-Chuan Chang
Wanxiang Che
Ciprian Chelba
Aitao Chen
Bin Chen
Boxing Chen
Chien Chin Chen
Harr Chen
Hsin-Hsi Chen
Jinying Chen
Keh-Jiann Chen
Wenliang Chen
Xiao Chen
Charibeth K. Cheng
Colin Cherry
Tee Kiah Chia
David Chiang
Yejin Choi
Kenneth Church
Massimiliano Ciaramita
Philipp Cimiano
Shay Cohen
Kevyn Collins-Thompson
John Conroy
Bonaventura Coppola
Ed Cormany
Dan Cristea
Bruce Croft
Dick Crouch
Montse Cuadros
Silviu-Petru Cucerzan
Oliver Čulo
Aron Culotta
Beatrice Daille
Hercules Dalianis
Sandipan Dandapat
Hoa Dang
Dana Dannells
Dipanjan Das
Sajib Dasgupta
Hal Daume
Dmitry Davidov
Adria de Gispert
Eric De La Clergerie
Valeria de Paiva
Maarten de Rijke
Thierry Declerck
Vera Demberg
Steve DeNeefe
John DeNero
Pascal Denis
Mona Diab
Georgiana Dinu
Pinar Donmez
Iustin Dornescu
Ascander Dost
Antoine Doucet
Doug Downey
Markus Dreyer
Gregory Druck
Xiangyu Duan
Amit Dubey
Kevin Duh
Michael Dukes
Chris Dyer
Marc Dymetman
Markus Egg

Koji Eguchi
David Eichmann
Andreas Eisele
Jacob Eisenstein
Jason Eisner
Noemie Elhadad
Charles Elkan
Micha Elsner
Alexandre entiev
Katrin Erk
Andrea Esuli
Richard Evans
Roger Evans
Patrick Fan
Alex Fang
Benoit Favre
Marcello Federico
Christiane Fellbaum
Katja Filippova
Radu Florian
Sandiway Fong
George Foster
Gil Francopoulo
Stefan Frank
Alex Fraser
Marjorie Freedman
Maria Fuentes
Hagen Fuerstenau
Atsushi Fujii
Sumio Fujita
Pascale Fung
Ryan Gabbard
Evgeniy Gabrilovich
Karthik Gali
Michel Galley
Michael Gamon
Kuzman Ganchev
Jianfeng Gao
Albert Gatt
Eric Gaussier
Ruifang Ge
Byron Georgantopoulos
Pablo Gervas
Sanjukta Ghosh
Daniel Gildea
Dan Gillick
Jesus Gimenez
Kevin Gimpel
Roxana Girju
Sharon Goldwater
Tianxia Gong
Julio Gonzalo
Cyril Goutte
Brigitte Grau
Stephan Greene
Mark Greenwood
Gregory Grefenstette
Ralph Grishman
Cecile Grivaz
Jiafeng Guo
Ben Hachey
Aria Haghighi
Udo Hahn
Jan Hajic
Eva Hajicova
Dilek Hakkani-tur
John Hale
David Hall
Greg Hanneman
Max Harper
Chikara Hashimoto
Ahmed Hassan
Claudia Hauff
Yoshihiko Hayashi
Ben He
daan he
Daqing He
Xiaodong He
Yulan He
John Henderson
Iris Hendrickx
Amac Herdagdelen
Ulf Hermjakob
Raquel Hervas
Dirk Heylen
Andrew Hickl
Graeme Hirst
Jerry Hobbs
Katja Hofmann
Steven Chu Hong Hoi
Kristy Hollingshead
Mark Hopkins
Eric Horvitz
Veronique Hoste
Yunhua Hu
Jimmy Huang
XuanJing Huang
Yifen Huang
Yun Huang
Matt Huenerfauth
Juan Huete
Sarmad Hussain
Rebecca Hwa
Gonzalo Iglesias
Steven Ikier
Diana Inkpen
Kentaro Inui
Elena Irimia
Amy Isard
Abe Ittycheriah
Tatsuya Izuha
Adam Jatowt
Jiwoon Jeon
Heng Ji
Sittichai Jiampojamarn
Daxin Jiang
Hongfei Jiang
Jiepu Jiang
Jing Jiang
Long Jiang
Wenbin Jiang
Valentin Jijkoun
Richard Johansson
Howard Johnson
Rie Johnson
Kristiina Jokinen
Doug Jones
Rosie Jones
Hanmin Jung
Vijay K. Shanker
Min-Yen Kan
Hiroshi Kanayama
Damianos Karakos
Nikiforos Karamanis
Rohit Kate
Tsuneaki Kato
Daisuke Kawahara
Tatsuya Kawahara
Junichi Kazama
Shahram Khadivi
Chloe Kiddon
Jin-Dong Kim
Min Kim
Chunyu Kit
Kevin Knight
Youngjoong Ko
Philipp Koehn
Rob Koeling
Dimitrios Kokkinakis
Greg Kondrak

Moshe Koppel
Valia Kordoni
Lili Kotlerman
Eric Kow
Zornitsa Kozareva
Emiel Kramer
Steven Krauwer
Gerhard Kremer
Canasai Kruengkrai
Lun-Wei Ku
Taku Kudo
Jonas Kuhn
Roland Kuhn
Peter Khnlein
Amba Kulkarni
Ravi Kumar
Shankar Kumar
A Kumaran
Oren Kurland
K. L. Kwok
Olivia Kwong
Wai Lam
Man Lan
Philippe Langlais
Francois Lareau
Martha Larson
Alex Lascarides
Alberto Lavelli
Alon Lavie
Florian Laws
Guy Lebanon
Changki Lee
John Lee
Joo Young Lee
Lillian Lee
Seungwoo Lee
Tan Lee
Jochen Leidner
Yves Lepage
Kevin Lerman
James Lester
Gregor Leusch
Gina-Anne Levow
Hang Li
Jiye Li
Lei Li
Mu Li
Shiqi Li
Shoushan Li
Wei Li
Wen Jie Li
Zhifei Li
Dekang Lin
Lucian Vlad Lita
Diane Litman
Bing Liu
Shui Liu
Ting Liu
Xiaohua Liu
Yan Liu
Yang Liu
Yang Liu
Yupeng Liu
Zhanyi Liu
Zhaopengx Liu
Anna Lobanova
Qiu Long
Adam Lopez
Qin Lu
Yue Lu
Harald Lngen
Xiaoqiang Luo
Tomoya Lwakura
Bin Ma
Qing Ma
Yanjun Ma
Wolfgang Macherey
Nitin Madnani
Hala Maghout
B. Mallikarjun
Gideon Mann
Daniel Marcu
Mitch Marcus
Joseph Mariani
Katja Markert
Erwin Marsi
M. Antonia Marti Antonin
Jean-Claude MARTIN
Andre Martins
Yuval Marton
Yuji Masumoto
Shigeki Matsubara
Tomoko Matsui
Yuji Matsumoto
Takuya Matsuzaki
Evgeny Matusov
Mausam
Arne Mauser
Marshall Mayberry
Diana McCarthy
David McClosky
Ryan McDonald
Tara McIntosh
Kathy McKeown
Susan McRoy
Yashar Mehdad
Qiaozhu Mei
Chris Mellish
Amlia Mendes
Helen Meng
Donald Metzler
Haitao Mi
Jun Miao
Lukas Michelbacher
Jeffrey Micher
Eleni Miltsakaki
David Mimno
Zhaoyan Ming
Shachar Mirkin
Jeff Mitchell
Vibhu Mittal
Yusuke Miyao
Marie-Francine Moens
Dan Moldovan
Diego Moll Aliod
Christof Monz
Raymond J. Mooney
Bob Moore
Roser Morante
Louis-Philippe Morency
Alessandro Moschitti
Isabelle Moulinier
Animesh Mukherjee
Stefan Mller
Art Munson
Dragos Munteanu
Masaki Murata
Vanessa Murdock
Smaranda Muresan
Gabriel Murray
Pradeep Muthukrishnan
Sobha Nair
Testuji Nakagawa
Vivi Nastase
Martina Naughton
Roberto Navigli
Mark-Jan Nederhof
Vasek Nemcik
Ani Nenkova
Hwee Tou Ng

Vincent Ng
Huyen Nguyen Thi Minh
Patrick Nguyen
Alex Niculescu-Mizil
Jian-Yun Nie
Rodney Nielsen
Takashi Ninomiya
Toyoaki Nishida
Cheng Niu
Zheng-Yu Niu
Diarmuid Saghdha
Franz Och
Michael ODonnell
Stephan Oepen
Naoki Okazaki
Lilja Ovrelid
Derya Ozkan
Ulrike Pado
Alexis Palmer
Sinno Jialin Pan
Patrick Pantel
Simone Paolo Ponzetto
Jong Park
Kristen Parton
Marius Pasca
Siddharth Patwardhan
Michael Paul
Soma Paul
Matthias Paulik
Yves Peirsman
Gerald Penn
Marco Pennacchiotti
Bhaskararao Peri
Aina Peris
Wim Peters
Kay Peterson
Slav Petrov
Daniele Pighin
Prasad Pingali
Stelios Piperidis
Guillaume Pitel
Paul Piwek
Luiz Pizzato
Massimo Poesio
Hoifung Poon
Ana-Maria Popescu
Maja Popovic
Christopher Potts
Richard Power
Rashmi Prasad
Laurent Prevot
Matthew Purver
Haoliang Qi
Toh Zhi Qiang
Tao Qin
Yang Qu
Chris Quirk
Bhiksha Raj
S. Rajendran
Dan Ramage
S.V. Ramanan
P. V. S. Rambabu
Owen Rambow
Delip Rao
Ari Rappoport
Paul Rayson
Michaela Regneri
Nils Reiter
Norbert Reithinger
Giuseppe Riccardi
Sebastian Riedel
German Rigau
Hae-Chang Rim
Lucia Rino
Hammam Riza
Horacio Rodrłguez
Gerhard Rolletschek
Lorenza Romano
Mathias Rossignol
Antti-Veikko Rosti
Michael Roth
Alex Rudnick
Marta Ruiz
Kenji Sagae
Horacio Saggion
Harabagiu Sanda
Baskaran Sankaran
Ratna Sanyal
Murat Saraclar
Sudeshna Sarkar
Manabu Sassano
Sekine Satoshi
Roser Sauri
Helmut Schmid
William Schuler
Sabine Schulte im Walde
Ineke Schuurman
Lane Schwartz
Wolfgang Seeker
sekine Sekine
Stephanie Seneff
Violeta Seretan
Hendra Setiawan
Chirag Shah
Dipti Sharma
Dou Shen
Libin Shen
Wade Shen
Kiyoaki Shirai
Eyal Shnarch
Advaith Siddharthan
Candy Sidner
Khalil Simaan
Michel Simard
Kiril Simov
Anil Kumar Singh
Smriti Singh
Samar Sinha
Sharon Small
Jason Smith
Nathaniel Smith
Matthew Snover
Benjamin Snyder
Stephen Soderland
Swapna Somasundaran
Yan Song
Young-In Song
Virach Sornlertlamvanich
Lucia Specia
Jennifer Spenader
Valentin Spitkovsky
Richard Sproat
Ed Stabler
Manfred Stede
Mark Steedman
Amanda Stent
Mark Stevenson
Matthew Stone
Svetlana Stoyanchev
Veselin Stoyanov
Kristina Striegnitz
Michael Strube
Tomek Strzalkowski
Jian Su
Jinsong Su
Keh-Yih Su
Nam Kim Su
Rajen Subba
K. V. Subbarao
L. V. Subramaniam

Zhifang Sui
Eiichiro Sumita
Congkai Sun
Jun Sun
Maosong Sun
Hisami Suzuki
Gyorgy Szarvas
Hiroya Takamura
Koichi Takeuchi
David Talbot
Chew Lim Tan
Jie Tang
Chua Tat-Seng
Paul Tepper
Simone Teufel
Stefan Thater
Mariet Theune
Paul Thomas
Vu Thuy
Joerg Tiedemann
Christoph Tillmann
Ivan Titov
Katrin Tomanek
Yoichi Tomiura
Sara Tonelli
Kentaro Torisawa
Kristina Toutanova
Roy Tromble
Huihsin Tseng
Shu-Chuan Tseng
Benjamin Tsou
Yuta Tsuboi
Jun-ichi Tsujii
Koji Tsukamoto
Yoshimasa Tsuruoka
Dan Tufis
Gokhan Tur
Kiyotaka Uchimoto
Yuya Unno
Lonneke Van der Plas
ielka van der sluis
Josef van Genabith
Gertjan van Noord
Lucy Vanderwende
Sebastian Varges
Sriram Venkatapathy
Marc Verhagen
Yannick Versley
Cristina Vertan
Marta Vila
David Vilar
Aline Villavicencio
Stephan Vogel
Clare Voss
Piek Vossen
Marilyn Walker
Hanna Wallach
Xiaojun Wan
Wenting Wang
Haifeng Wang
Hsin-Min Wang
Kai Wang
Rui Wang
Wei Wang
Xuanhui Wang
Leo Wanner
Nigel Ward
Taro Watanabe
Yotaro Watanabe
Nick Webb
Bonnie Webber
Eric Wehrli
Kilian Weinberger
David Weir
Michael White
Michael Wick
Jan Wiebe
Yorick Wilks
Theresa Wilson
Shuly Wintner
Andreas Witt
Peter Wittek
Magda Wolska
Chung-Hsien Wu
Dan Wu
Dekai Wu
Hua Wu
Xiaoyun Wu
Yunfang Wu
Fei Xia
Rui Xia
Xinyan Xiao
Deyi Xiong
Peng Xu
Ruifeng Xu
Bert Xue
Yongxin Yan
Muyun Yang
Qiang Yang
Ainur Yessenalina
Bei Yu
Jianxing Yu
Kun Yu
Mo Yu
Yisong Yue
Zdenek Zabokrtsky
Taras Zagibalov
Omar Zaidan
Roberto Zanoli
Fabio Zanzotto
Alessandra Zarcone
Richard Zens
Torsten Zesch
Luke Zettlemoyer
Hao Zhang
Hui Zhang
Jiajun Zhang
Min Zhang
Ruiqiang Zhang
Yujie Zhang
Ziqi Zhang
Hai Zhao
Jun Zhao
Bowen Zhou
Guodong Zhou
Yiping Zhou
Conghui Zhu
Jerry Zhu
Tao Zhuang
Heike Zinsmeister
Imed Zitouni
Andreas Zollmann
Chengqing Zong
Ingrid Zukerman

Table of Contents

Testing SDRT's Right Frontier
Stergos Afantenos and Nicholas Asher 1

Identifying Multi-word Expressions by Leveraging Morphological and Syntactic Idiosyncrasy
Hassan Al-Haj and Shuly Wintner 10

Robust Measurement and Comparison of Context Similarity for Finding Translation Pairs
Daniel Andrade, Tetsuya Nasukawa and Junichi Tsujii 19

Multilingual Subjectivity: Are More Languages Better?
Carmen Banea, Rada Mihalcea and Janyce Wiebe 28

Plagiarism Detection across Distant Language Pairs
Alberto Barrón-Cedeño, Paolo Rosso, Eneko Agirre and Gorka Labaka 37

Automatic Detection of Non-deverbal Event Nouns for Quick Lexicon Production
Nuria Bel, Maria Coll and Gabriela Resnik 46

Fast and Accurate Arc Filtering for Dependency Parsing
Shane Bergsma and Colin Cherry 53

A Hierarchical Classifier Applied to Multi-way Sentiment Detection
Adrian Bickerstaffe and Ingrid Zukerman 62

Fluency Constraints for Minimum Bayes-Risk Decoding of Statistical Machine Translation Lattices
Graeme Blackwood, Adrià de Gispert and William Byrne 71

Self-Annotation for fine-grained geospatial relation extraction
Andre Blessing and Hinrich Schütze 80

Top Accuracy and Fast Dependency Parsing is not a Contradiction
Bernd Bohnet 89

Broad Coverage Multilingual Deep Sentence Generation with a Stochastic Multi-Level Realizer
Bernd Bohnet, Leo Wanner, Simon Mill and Alicia Burga 98

Towards an optimal weighting of context words based on distance
Bernard Brosseau-Villeneuve, Jian-Yun Nie and Noriko Kando 107

Measuring the Non-compositionality of Multiword Expressions
Fan Bu, Xiaoyan Zhu and Ming Li 116

A Utility-Driven Approach to Question Ranking in Social QA
Razvan Bunescu and Yunfeng Huang 125

Simultaneous Ranking and Clustering of Sentences: A Reinforcement Approach to Multi-Document Summarization
Xiaoyan Cai, Wenjie Li, You Ouyang and Hong Yan 134

End-to-End Coreference Resolution via Hypergraph Partitioning
Jie Cai and Michael Strube 143

Exploiting Background Knowledge for Relation Extraction
Yee Seng Chan and Dan Roth 152

Jointly Modeling WSD and SRL with Markov Logic
Wanxiang Che and Ting Liu 161

Bipolar Person Name Identification of Topic Documents Using Principal Component Analysis
Chien Chin Chen and Chen-Yuan Wu 170

Emotion Cause Detection with Linguistic Constructions
Ying Chen, Sophia Yat Mei Lee, Shoushan Li and Chu-Ren Huang 179

A Twin-Candidate Based Approach for Event Pronoun Resolution using Composite Kernel
Bin Chen, Jian Su and Chew Lim Tan 188

Unsupervised Synthesis of Multilingual Wikipedia Articles
Yuncong Chen and Pascale Fung 197

Simplicity is Better: Revisiting Single Kernel PPI Extraction
Sung-Pil Choi and Sung-Hyon Myaeng 206

An ontology-driven system for detecting global health events
Nigel Collier, Reiko Matsuda Goodwin, John McCrae, Son Doan, Ai Kawazoe, Mike Conway, Asanee Kawtrakul, Koichi Takeuchi and Dinh Dien 215

Constraining robust constructions for broad-coverage parsing with precision grammars
Bart Cramer and Yi Zhang 223

Local lexical adaptation in Machine Translation through triangulation: SMT helping SMT
Josep Maria Crego, Aurélien Max and François Yvon 232

Automated Translation of Semantic Relationships
Dmitry Davidov and Ari Rappoport 241

Comparison of different algebras for inducing the temporal structure of texts
Pascal Denis and Philippe Muller 250

Generating Learner-Like Morphological Errors in Russian
Markus Dickinson 259

Resolving Object and Attribute Coreference in Opinion Mining
Xiaowen Ding and Bing Liu 268

Entity Disambiguation for Knowledge Base Population
Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber and Tim Finin 277

A Discriminative Latent Variable-Based "DE" Classifier for Chinese-English SMT
Jinhua Du and Andy Way 286

An Empirical Study on Learning to Rank of Tweets
Yajuan Duan, Long Jiang, Tao Qin, Ming Zhou and Heung-Yeung Shum 295

Translation Model Generalization using Probability Averaging for Machine Translation
Nan Duan, Hong Sun and Ming Zhou 304

Mixture Model-based Minimum Bayes Risk Decoding using Multiple Machine Translation Systems
Nan Duan, Mu Li, Dongdong Zhang and Ming Zhou 313

Multi-Sentence Compression: Finding Shortest Paths in Word Graphs
Katja Filippova 322

Enriching Dictionaries with Images from the Internet - Targeting Wikipedia and a Japanese Semantic Lexicon: Lexeed -
Sanae Fujita and Masaaki Nagata 331

Opinosis: A Graph Based Approach to Abstractive Summarization of Highly Redundant Opinions
Kavita Ganesan, ChengXiang Zhai and Jiawei Han 340

EMDC: A Semi-supervised Approach for Word Alignment
Qin Gao, Francisco Guzman and Stephan Vogel 349

A Large Scale Ranker-Based System for Search Query Spelling Correction
Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk and Xu Sun 358

RTG based surface realisation for TAG
Claire Gardent and Laura Perez-Beltrachini 367

Automatically Learning Source-side Reordering Rules for Large Scale Machine Translation
Dmitriy Genzel 376

Comparing Language Similarity across Genetic and Typologically-Based Groupings
Ryan Georgi, Fei Xia and William Lewis 385

Better Arabic Parsing: Baselines, Evaluations, and Analysis
Spence Green and Christopher D. Manning 394

Paraphrase Alignment for Synonym Evidence Discovery
Gintarè Grigonytè, João Paulo Cordeiro, Gaël Dias, Rumen Moraliyski and Pavel Brazdil . . . 403

Finding the Storyteller: Automatic Spoiler Tagging using Linguistic Cues
Sheng Guo and Naren Ramakrishnan 412

Detection of Simple Plagiarism in Computer Science Papers
Yaakov HaCohen-Kerner, Aharon Tayeb and Natan Ben-Dror 421

A Structured Vector Space Model for Hidden Attribute Meaning in Adjective-Noun Phrases
Matthias Hartung and Anette Frank 430

Hierarchical Phrase-based Machine Translation with Word-based Reordering Model
Katsuhiko Hayashi, Hajime Tsukada, Katsuhito Sudoh, Kevin Duh and Seiichi Yamamoto . . . 439

A Novel Reordering Model Based on Multi-layer Phrase for Statistical Machine Translation
Yanqing He, Yu Zhou, Chengqing Zong and Huilin Wang 447

Standardizing Wordnets in the ISO Standard LMF: Wordnet-LMF for GermaNet
Verena Henrich and Erhard Hinrichs 456

Normal-form parsing for Combinatory Categorial Grammars with generalized composition and type-raising
Julia Hockenmaier and Yonatan Bisk 465

An Empirical Study on Web Mining of Parallel Data
Gumwon Hong, Chi-Ho Li, Ming Zhou and Hae-Chang Rim 474

Enhancing Cross Document Coreference of Web Documents with Context Similarity and Very Large Scale Text Categorization
Jian Huang, Pucktada Treeratpituk, Sarah Taylor and C. Lee Giles 483

Feature-Rich Discriminative Phrase Rescoring for SMT
Fei Huang and Bing Xiang 492

FactRank: Random Walks on a Web of Facts
Alpa Jain and Patrick Pantel 501

Open Entity Extraction from Web Search Query Logs
Alpa Jain and Marco Pennacchiotti 510

Reranking Models in Fine-grained Opinion Analysis
Richard Johansson and Alessandro Moschitti 519

Unsupervised phonemic Chinese word segmentation using Adaptor Grammars
Mark Johnson and Katherine Demuth 528

Data-Driven Parsing with Probabilistic Linear Context-Free Rewriting Systems
Laura Kallmeyer and Wolfgang Maier 537

Learning to Predict Readability using Diverse Linguistic Features
Rohit Kate, Xiaoqiang Luo, Siddharth Patwardhan, Martin Franz, Radu Florian, Raymond Mooney, Salim Roukos and Chris Welty 546

Value for Money: Balancing Annotation Effort, Lexicon Building and Accuracy for Multilingual WSD
Mitesh Khapra, Saurabh Sohoney, Anup Kulkarni and Pushpak Bhattacharyya 555

A Cross-lingual Annotation Projection Approach for Relation Detection
Seokhwan Kim, Minwoo Jeong, Jonghoon Lee and Gary Geunbae Lee 564

Evaluating N-gram based Evaluation Metrics for Automatic Keyphrase Extraction
Su Nam Kim, Timothy Baldwin and Min-Yen Kan 572

Improving the Quality of Text Understanding by Delaying Ambiguity Resolution
Doo Soon Kim, Ken Barker and Bruce Porter 581

Automatic generation of inter-passage links based on semantic similarity
Petr Knoth, Jakub Novotny and Zdenek Zdrahal 590

Dependency-driven Anaphoricity Determination for Coreference Resolution
Fang Kong, Guodong Zhou, Longhua Qian and Qiaoming Zhu 599

Phrase Clustering for Smoothing TM Probabilities - or, How to Extract Paraphrases from Phrase Tables
Roland Kuhn, Boxing Chen, George Foster and Evan Stratford 608

Revisiting Context-based Projection Methods for Term-Translation Spotting in Comparable Corpora
Audrey Laroche and Philippe Langlais 617

Constituent Reordering and Syntax Models for English-to-Japanese Statistical Machine Translation
Young-Suk Lee, Bing Zhao and Xiaoqian Luo 626

Sentiment Classification and Polarity Shifting
Shoushan Li, Sophia Y. M. Lee, Ying Chen, Chu-Ren Huang and Guodong Zhou 635

Improving Corpus Comparability for Bilingual Lexicon Extraction from Comparable Corpora
Bo Li and Eric Gaussier 644

Structure-Aware Review Mining and Summarization
Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang and Hao Yu . . . 653

Adaptive Development Data Selection for Log-linear Model in Statistical Machine Translation
Mu Li, Yinggong Zhao, Dongdong Zhang and Ming Zhou 662

Learning the Scope of Negation via Shallow Semantic Parsing
Junhui Li, Guodong Zhou, Hongling Wang and Qiaoming Zhu 671

Filtered Ranking for Bootstrapping in Event Extraction
Shasha Liao and Ralph Grishman 680

Exploring variation across biomedical subdomains
Tom Lippincott, Diarmuid Ó Séaghdha, Lin Sun and Anna Korhonen 689

Semantic Role Labeling for News Tweets
Xiaohua Liu, Kuan Li, Bo Han, Ming Zhou, Long Jiang, Zhongyang Xiong and Changning Huang
698

Joint Parsing and Translation
Yang Liu and Qun Liu 707

Semantic Role Features for Machine Translation
Ding Liu and Daniel Gildea 716

TimeML Events Recognition and Classification: Learning CRF Models with Semantic Roles
Hector Llorens, Estela Saquete and Borja Navarro-Colorado 725

Exploiting Structured Ontology to Organize Scattered Online Opinions
Yue Lu, Huizhong Duan, Hongning Wang and ChengXiang Zhai 734

Enhancing Morphological Alignment for Translating Highly Inflected Languages
Minh-Thang Luong and Min-Yen Kan 743

Automatic analysis of semantic similarity in comparable text through syntactic tree matching
Erwin Marsi and Emiel Krahmer 752

Controlling Listening-oriented Dialogue using Partially Observable Markov Decision Processes
Toyomi Meguro, Ryuichiro Higashinaka, Yasuhiro Minami and Kohji Dohsaka 761

Recognising Entailment within Discourse
Shachar Mirkin, Jonathan Berant, Ido Dagan and Eyal Shnarch 770

Evaluating Dependency Representations for Event Extraction
Makoto Miwa, Sampo Pyysalo, Tadayoshi Hara and Jun'ichi Tsujii 779

Entity-Focused Sentence Simplification for Relation Extraction
Makoto Miwa, Rune Sætre, Yusuke Miyao and Jun'ichi Tsujii 788

Using Cross-Lingual Projections to Generate Semantic Role Labeled Annotated Corpus for Urdu - A Resource Poor Language
Smruthi Mukund, Debanjan Ghosh and Rohini Srihari 797

Recognition of Affect, Judgment, and Appreciation in Text
Alena Neviarouskaya, Helmut Prendinger and Mitsuru Ishizuka 806

Nonparametric Word Segmentation for Machine Translation
ThuyLinh Nguyen, Stephan Vogel and Noah A. Smith 815

Automatic Discovery of Feature Sets for Dependency Parsing
Peter Nilsson and Pierre Nugues 824

Evaluation of Dependency Parsers on Unbounded Dependencies
Joakim Nivre, Laura Rimell, Ryan McDonald and Carlos Gómez Rodríguez 833

Co-STAR: A Co-training Style Algorithm for Hyponymy Relation Acquisition from Structured and Unstructured Text
Jong-Hoon Oh, Ichiro Yamada, Kentaro Torisawa and Stijn De Saeger 842

Simple and Efficient Algorithm for Approximate Dictionary Matching
Naoaki Okazaki and Jun'ichi Tsujii 851

Latent Mixture of Discriminative Experts for Multimodal Prediction Modeling
Derya Ozkan, Kenji Sagae and Louis-Philippe Morency 860

Text Summarization of Turkish Texts using Latent Semantic Analysis
Makbule Ozsoy, Ilyas Cicekli and Ferda Alpaslan 869

Knowing What to Believe (when you already know something)
Jeff Pasternack and Dan Roth 877

Using Web-scale N-grams to Improve Base NP Parsing Performance
Emily Pitler, Shane Bergsma, Dekang Lin and Kenneth Church 886

Citation Summarization Through Keyphrase Extraction
Vahed Qazvinian, Dragomir R. Radev and Arzucan Ozgur 895

2D Trie for Fast Parsing
Xian Qian, Qi Zhang, Xuangjing Huang and Lide Wu 904

The Bag-of-Opinions Method for Review Rating Prediction from Sparse Text Patterns
Lizhen Qu, Georgiana Ifrim and Gerhard Weikum 913

An Exploration of Features for Recognizing Word Emotion
Changqin Quan and Fuji Ren 922

Inducing Fine-Grained Semantic Classes via Hierarchical and Collective Classification
Altaf Rahman and Vincent Ng 931

Fast, Greedy Model Minimization for Unsupervised Tagging
Sujith Ravi, Ashish Vaswani, Kevin Knight and David Chiang 940

Bringing Active Learning to Life
Ines Rehbein, Josef Ruppenhofer and Alexis Palmer 949

Computing EM-based Alignments of Routes and Route Directions as a Basis for Natural Language Generation
Michael Roth and Anette Frank 958

A Multiple-Domain Ontology Builder
Sara Salem and Samir AbdelRahman 967

Weakly Supervised Morphology Learning for Agglutinating Languages Using Small Training Sets.
Ksenia Shalonova and Bruno Golenia 976

Multi-Document Summarization via the Minimum Dominating Set
Chao Shen and Tao Li 984

Corpus-based Semantic Class Mining: Distributional vs. Pattern-Based Approaches
Shuming Shi, Huibin Zhang, Xiaojie Yuan and Ji-Rong Wen 993

Metaphor Identification Using Verb and Noun Clustering
Ekaterina Shutova, Lin Sun and Anna Korhonen 1002

Explore the Structure of Social Tags by Subsumption Relations
Xiance Si, Zhiyuan Liu and Maosong Sun 1011

Ukwabelana - An open-source morphological Zulu corpus
Sebastian Spiegler, Andrew van der Spuy and Peter A. Flach 1020

EMMA: A novel Evaluation Metric for Morphological Analysis
Sebastian Spiegler and Christian Monson 1029

Modeling Socio-Cultural Phenomena in Discourse
Tomek Strzalkowski, George Aaron Broadwell, Jennifer Stromer-Galley, Samira Shaikh, Sarah Taylor and Nick Webb 1038

Discriminative Induction of Sub-Tree Alignment using Limited Labeled Data
Jun Sun, Min Zhang and Chew Lim Tan 1047

Investigating the cross-linguistic potential of VerbNet-style classification
Lin Sun, Thierry Poibeau, Anna Korhonen and Cedric Messiant 1056

Semi-supervised dependency parsing using generalized tri-training
Anders Søgaard and Christian Rishøj 1065

SemanticRank: Ranking Keywords and Sentences Using Semantic Graphs
George Tsatsaronis, Iraklis Varlamis and Kjetil Nørvåg 1074

Chinese CCGbank: extracting CCG derivations from the Penn Chinese Treebank
Daniel Tse and James R. Curran 1083

Dependency Forest for Statistical Machine Translation
Zhaopeng Tu, Yang Liu, Young-Sook Hwang, Qun Liu and Shouxun Lin 1092

Large Scale Parallel Document Mining for Machine Translation
Jakob Uszkoreit, Jay Ponte, Ashok Popat and Moshe Dubiner 1101

Hungarian Corpus of Light Verb Constructions
Veronika Vincze and János Csirik 1110

Syntax Based Reordering with Automatically Derived Rules for Improved Statistical Machine Translation
Karthik Visweswariah, Jiri Navratil, Jeffrey Sorensen, Vijil Chenthamarakshan and Nandakishore Kambhatla 1119

Efficient Statement Identification for Automatic Market Forecasting
Henning Wachsmuth, Peter Prettenhofer and Benno Stein 1128

Towards a Unified Approach to Simultaneous Single-Document and Multi-Document Summarizations
Xiaojun Wan 1137

”Got You!”: Automatic Vandalism Detection in Wikipedia with Web-based Shallow Syntactic-Semantic Modeling
William Yang Wang and Kathleen McKeown 1146

Exploiting Salient Patterns for Question Detection and Question Retrieval in Community-based Question Answering
Kai Wang and Tat-Seng Chua 1155

Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering
Mengqiu Wang and Christopher Manning 1164

A Character-Based Joint Model for Chinese Word Segmentation
Kun Wang, Chengqing Zong and Keh-Yih Su 1173

Near-synonym Lexical Choice in Latent Semantic Space
Tong Wang and Graeme Hirst 1182

Disambiguating Dynamic Sentiment Ambiguous Adjectives
Yunfang Wu and Miaomiao Wen 1191

Joint Tokenization and Translation
Xinyan Xiao, Yang Liu, YoungSook Hwang, Qun Liu and Shouxun Lin 1200

Build Chinese Emotion Lexicons Using A Graph-based Algorithm and Multiple Resources
Ge Xu, Xinfan Meng and Houfeng Wang 1209

A Methodology for Automatic Identification of Nocuous Ambiguity
Hui Yang, Anne De Roeck, Alistair Willis and Bashar Nuseibeh 1218

Contextual Modeling for Meeting Translation Using Unsupervised Word Sense Disambiguation
Mei Yang and Katrin Kirchhoff 1227

A Working Report on Statistically Modeling Dative Variation in Mandarin Chinese
Yao Yao and Feng-hsi Liu 1236

Kernel Slicing: Scalable Online Training with Conjunctive Features
Naoki Yoshinaga and Masaru Kitsuregawa 1245

Discriminative Training for Near-Synonym Substitution
Liang-Chih Yu, Hsiu-Min Shih, Yu-Ling Lai, Jui-Feng Yeh and Chung-Hsien Wu 1254

Estimating Linear Models for Compositional Distributional Semantics
Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi and Suresh Manandhar 1263

Grouping Product Features Using Semi-Supervised Learning with Soft-Constraints
Zhongwu Zhai, Bing Liu, Hua Xu and Peifa Jia 1272

Forest-guided Supertagger Training
Yao-zhong Zhang, Takuya Matsuzaki and Jun'ichi Tsujii 1281

Entity Linking Leveraging Automatically Generated Annotation
Wei Zhang, Jian Su, Chew Lim Tan and Wen Ting Wang 1290

A Rhetorical Syntax-Driven Model for Speech Summarization
Jian Zhang and Pascale Fung 1299

Maximum Metric Score Training for Coreference Resolution
Shanheng Zhao and Hwee Tou Ng 1308

Paraphrasing with Search Engine Query Logs
Shiqi Zhao, Haifeng Wang and Ting Liu 1317

Leveraging Multiple MT Engines for Paraphrase Generation
Shiqi Zhao, Haifeng Wang, Xiang Lan and Ting Liu 1326

Resolving Surface Forms to Wikipedia Topics
Yiping Zhou, Lan Nie, Omid Rouhani-Kalleh, Flavian Vasile and Scott Gaffney 1335

Heterogeneous Parsing via Collaborative Decoding
Muhua Zhu, Jingbo Zhu and Tong Xiao 1344

A Monolingual Tree-based Translation Model for Sentence Simplification
Zhemin Zhu, Delphine Bernhard and Iryna Gurevych 1353

A Minimum Error Weighting Combination Strategy for Chinese Semantic Role Labeling
Tao Zhuang and Chengqing Zong 1362

Detecting Speech Repairs Incrementally Using a Noisy Channel Approach
Simon Zwarts, Mark Johnson and Robert Dale 1371

Syntactic Scope Resolution in Uncertainty Analysis
Lilja Øvrelid, Erik Velldal and Stephan Oepen 1379

Author Index

AbdelRahman, Samir, 967
Afantenos, Stergos, 1
Agirre, Eneko, 37
Al-Haj, Hassan, 10
Alpaslan, Ferda, 869
Andrade, Daniel, 19
Asher, Nicholas, 1

Baldwin, Timothy, 572
Banea, Carmen, 28
Barker, Ken, 581
Barrón-Cedeño, Alberto, 37
Bel, Nuria, 46
Ben-Dror, Natan, 421
Berant, Jonathan, 770
Bergsma, Shane, 53, 886
Bernhard, Delphine, 1353
Bhattacharyya, Pushpak, 555
Bickerstaffe, Adrian, 62
Bisk, Yonatan, 465
Blackwood, Graeme, 71
Blessing, Andre, 80
Bohnet, Bernd, 89, 98
Brazdil, Pavel, 403
Broadwell, George Aaron, 1038
Brosseau-Villeneuve, Bernard, 107
Bu, Fan, 116
Bunescu, Razvan, 125
Burga, Alicia, 98
Byrne, William, 71

Cai, Jie, 143
Cai, Xiaoyan, 134
Chan, Yee Seng, 152
Che, Wanxiang, 161
Chen, Bin, 188
Chen, Boxing, 608
Chen, Chien Chin, 170
Chen, Ying, 179, 635
Chen, Yuncong, 197
Chenthamarakshan, Vijil, 1119
Cherry, Colin, 53
Chiang, David, 940
Choi, Sung-Pil, 206
Chua, Tat-Seng, 1155
Church, Kenneth, 886
Cicekli, Ilyas, 869
Coll, Maria, 46
Collier, Nigel, 215
Conway, Mike, 215
Cordeiro, João Paulo, 403
Cramer, Bart, 223
Crego, Josep Maria, 232
Csirik, János, 1110
Curran, James R., 1083

Dagan, Ido, 770
Dale, Robert, 1371
Davidov, Dmitry, 241
de Gispert, Adrià, 71
De Roeck, Anne, 1218
De Saeger, Stijn, 842
Demuth, Katherine, 528
Denis, Pascal, 250
Dias, Gaël, 403
Dickinson, Markus, 259
Dien, Dinh, 215
Ding, Xiaowen, 268
Doan, Son, 215
Dohsaka, Kohji, 761
Dredze, Mark, 277
Du, Jinhua, 286
Duan, Huizhong, 734
Duan, Nan, 304, 313
Duan, Yajuan, 295
Dubiner, Moshe, 1101
Duh, Kevin, 439

Fallucchi, Francesca, 1263
Filippova, Katja, 322

Finin, Tim, 277
Flach, Peter A., 1020
Florian, Radu, 546
Foster, George, 608
Frank, Anette, 430, 958
Franz, Martin, 546
Fujita, Sanae, 331
Fung, Pascale, 197, 1299

Gaffney, Scott, 1335
Ganesan, Kavita, 340
Gao, Jianfeng, 358
Gao, Qin, 349
Gardent, Claire, 367
Gaussier, Eric, 644
Genzel, Dmitriy, 376
Georgi, Ryan, 385
Gerber, Adam, 277
Ghosh, Debanjan, 797
Gildea, Daniel, 716
Giles, C. Lee, 483
Golenia, Bruno, 976
Gómez Rodríguez, Carlos, 833
Green, Spence, 394
Grigonytè, Gintarè, 403
Grishman, Ralph, 680
Guo, Sheng, 412
Gurevych, Iryna, 1353
Guzman, Francisco, 349

HaCohen-Kerner, Yaakov, 421
Han, Bo, 698
Han, Chao, 653
Han, Jiawei, 340
Hara, Tadayoshi, 779
Hartung, Matthias, 430
Hayashi, Katsuhiko, 439
He, Yanqing, 447
Henrich, Verena, 456
Higashinaka, Ryuichiro, 761
Hinrichs, Erhard, 456
Hirst, Graeme, 1182
Hockenmaier, Julia, 465
Hong, Gumwon, 474
Huang, Changning, 698
Huang, Chu-Ren, 179, 635
Huang, Fei, 492
Huang, Jian, 483
Huang, Minlie, 653
Huang, Xuangjing, 904
Huang, Yunfeng, 125
Hwang, Young-Sook, 1092
Hwang, YoungSook, 1200

Ifrim, Georgiana, 913
Ishizuka, Mitsuru, 806

Jain, Alpa, 501, 510
Jeong, Minwoo, 564
Jia, Peifa, 1272
Jiang, Long, 295, 698
Johansson, Richard, 519
Johnson, Mark, 528, 1371

Kallmeyer, Laura, 537
Kambhatla, Nandakishore, 1119
Kan, Min-Yen, 572, 743
Kando, Noriko, 107
Kate, Rohit, 546
Kawazoe, Ai, 215
Kawtrakul, Asanee, 215
Khapra, Mitesh, 555
Kim, Doo Soon, 581
Kim, Seokhwan, 564
Kim, Su Nam, 572
Kirchhoff, Katrin, 1227
Kitsuregawa, Masaru, 1245
Knight, Kevin, 940
Knoth, Petr, 590
Kong, Fang, 599
Korhonen, Anna, 689, 1002, 1056
Korkontzelos, Ioannis, 1263
Krahmer, Emiel, 752
Kuhn, Roland, 608
Kulkarni, Anup, 555

Labaka, Gorka, 37
Lai, Yu-Ling, 1254
Lan, Xiang, 1326
Langlais, Philippe, 617
Laroche, Audrey, 617
Lee, Gary Geunbae, 564
Lee, Jonghoon, 564
Lee, Sophia Y. M., 635
Lee, Sophia Yat Mei, 179
Lee, Young-Suk, 626
Lewis, William, 385

Li, Bo, 644
Li, Chi-Ho, 474
Li, Fangtao, 653
Li, Junhui, 671
Li, Kuan, 698
Li, Ming, 116
Li, Mu, 313, 662
Li, Shoushan, 179, 635
Li, Tao, 984
Li, Wenjie, 134
Li, Xiaolong, 358
Liao, Shasha, 680
Lin, Dekang, 886
Lin, Shouxun, 1092, 1200
Lippincott, Tom, 689
Liu, Bing, 268, 1272
Liu, Ding, 716
Liu, Feng-hsi, 1236
Liu, Qun, 707, 1092, 1200
Liu, Ting, 161, 1317, 1326
Liu, Xiaohua, 698
Liu, Yang, 707, 1092, 1200
Liu, Zhiyuan, 1011
Llorens, Hector, 725
Lu, Yue, 734
Luo, Xiaoqian, 626
Luo, Xiaoqiang, 546
Luong, Minh-Thang, 743

Maier, Wolfgang, 537
Manandhar, Suresh, 1263
Manning, Christopher, 1164
Manning, Christopher D., 394
Marsi, Erwin, 752
Matsuda Goodwin, Reiko, 215
Matsuzaki, Takuya, 1281
Max, Aurélien, 232
McCrae, John, 215
McDonald, Ryan, 833
McKeown, Kathleen, 1146
McNamee, Paul, 277
Meguro, Toyomi, 761
Meng, Xinfan, 1209
Messiant, Cedric, 1056
Micol, Daniel, 358
Mihalcea, Rada, 28
Mill, Simon, 98
Minami, Yasuhiro, 761
Mirkin, Shachar, 770
Miwa, Makoto, 779, 788
Miyao, Yusuke, 788
Monson, Christian, 1029
Mooney, Raymond, 546
Moraliyski, Rumen, 403
Morency, Louis-Philippe, 860
Moschitti, Alessandro, 519
Mukund, Smruthi, 797
Muller, Philippe, 250
Myaeng, Sung-Hyon, 206

Nagata, Masaaki, 331
Nasukawa, Tetsuya, 19
Navarro-Colorado, Borja, 725
Navratil, Jiri, 1119
Neviarouskaya, Alena, 806
Ng, Hwee Tou, 1308
Ng, Vincent, 931
Nguyen, ThuyLinh, 815
Nie, Jian-Yun, 107
Nie, Lan, 1335
Nilsson, Peter, 824
Nivre, Joakim, 833
Novotny, Jakub, 590
Nørvåg, Kjetil, 1074
Nugues, Pierre, 824
Nuseibeh, Bashar, 1218

Ó Séaghdha, Diarmuid, 689
Oepen, Stephan, 1379
Oh, Jong-Hoon, 842
Okazaki, Naoaki, 851
Ouyang, You, 134
Ozgur, Arzucan, 895
Ozkan, Derya, 860
Ozsoy, Makbule, 869

Palmer, Alexis, 949
Pantel, Patrick, 501
Pasternack, Jeff, 877
Patwardhan, Siddharth, 546
Pennacchiotti, Marco, 510
Perez-Beltrachini, Laura, 367
Pitler, Emily, 886
Poibeau, Thierry, 1056
Ponte, Jay, 1101
Popat, Ashok, 1101

Porter, Bruce, 581
Prendinger, Helmut, 806
Prettenhofer, Peter, 1128
Pyysalo, Sampo, 779

Qazvinian, Vahed, 895
Qian, Longhua, 599
Qian, Xian, 904
Qin, Tao, 295
Qu, Lizhen, 913
Quan, Changqin, 922
Quirk, Chris, 358

Radev, Dragomir R., 895
Rahman, Altaf, 931
Ramakrishnan, Naren, 412
Rao, Delip, 277
Rappoport, Ari, 241
Ravi, Sujith, 940
Rehbein, Ines, 949
Ren, Fuji, 922
Resnik, Gabriela, 46
Rim, Hae-Chang, 474
Rimell, Laura, 833
Rishøj, Christian, 1065
Rosso, Paolo, 37
Roth, Dan, 152, 877
Roth, Michael, 958
Rouhani-Kalleh, Omid, 1335
Roukos, Salim, 546
Ruppenhofer, Josef, 949

Sagae, Kenji, 860
Salem, Sara, 967
Saquete, Estela, 725
Schütze, Hinrich, 80
Sætre, Rune, 788
Søgaard, Anders, 1065
Shaikh, Samira, 1038
Shalonova, Ksenia, 976
Shen, Chao, 984
Shi, Shuming, 993
Shih, Hsiu-Min, 1254
Shnarch, Eyal, 770
Shum, Heung-Yeung, 295
Shutova, Ekaterina, 1002
Si, Xiance, 1011
Smith, Noah A., 815

Sohoney, Saurabh, 555
Sorensen, Jeffrey, 1119
Spiegler, Sebastian, 1020, 1029
Srihari, Rohini, 797
Stein, Benno, 1128
Stratford, Evan, 608
Stromer-Galley, Jennifer, 1038
Strube, Michael, 143
Strzalkowski, Tomek, 1038
Su, Jian, 188, 1290
Su, Keh-Yih, 1173
Sudoh, Katsuhito, 439
Sun, Hong, 304
Sun, Jun, 1047
Sun, Lin, 689, 1002, 1056
Sun, Maosong, 1011
Sun, Xu, 358

Takeuchi, Koichi, 215
Tan, Chew Lim, 188, 1047, 1290
Tayeb, Aharon, 421
Taylor, Sarah, 483, 1038
Torisawa, Kentaro, 842
Treeratpituk, Pucktada, 483
Tsatsaronis, George, 1074
Tse, Daniel, 1083
Tsujii, Jun'ichi, 779, 788, 851, 1281
Tsujii, Junichi, 19
Tsukada, Hajime, 439
Tu, Zhaopeng, 1092

Uszkoreit, Jakob, 1101

van der Spuy, Andrew, 1020
Varlamis, Iraklis, 1074
Vasile, Flavian, 1335
Vaswani, Ashish, 940
Velldal, Erik, 1379
Vincze, Veronika, 1110
Visweswariah, Karthik, 1119
Vogel, Stephan, 349, 815
Øvrelid, Lilja, 1379

Wachsmuth, Henning, 1128
Wan, Xiaojun, 1137
Wang, Haifeng, 1317, 1326
Wang, Hongling, 671
Wang, Hongning, 734
Wang, Houfeng, 1209

Wang, Huilin, 447
Wang, Kai, 1155
Wang, Kun, 1173
Wang, Mengqiu, 1164
Wang, Tong, 1182
Wang, Wen Ting, 1290
Wang, William Yang, 1146
Wanner, Leo, 98
Way, Andy, 286
Webb, Nick, 1038
Weikum, Gerhard, 913
Welty, Chris, 546
Wen, Ji-Rong, 993
Wen, Miaomiao, 1191
Wiebe, Janyce, 28
Willis, Alistair, 1218
Wintner, Shuly, 10
Wu, Chen-Yuan, 170
Wu, Chung-Hsien, 1254
Wu, Lide, 904
Wu, Yunfang, 1191

Xia, Fei, 385
Xia, Ying-Ju, 653
Xiang, Bing, 492
Xiao, Tong, 1344
Xiao, Xinyan, 1200
Xiong, Zhongyang, 698
Xu, Ge, 1209
Xu, Hua, 1272

Yamada, Ichiro, 842
Yamamoto, Seiichi, 439
Yan, Hong, 134
Yang, Hui, 1218
Yang, Mei, 1227
Yao, Yao, 1236
Yeh, Jui-Feng, 1254
Yoshinaga, Naoki, 1245
Yu, Hao, 653
Yu, Liang-Chih, 1254
Yuan, Xiaojie, 993
Yvon, François, 232

Zanzotto, Fabio Massimo, 1263
Zdrahal, Zdenek, 590
Zhai, ChengXiang, 340, 734
Zhai, Zhongwu, 1272
Zhang, Dongdong, 313, 662
Zhang, Huibin, 993
Zhang, Jian, 1299
Zhang, Min, 1047
Zhang, Qi, 904
Zhang, Shu, 653
Zhang, Wei, 1290
Zhang, Yao-zhong, 1281
Zhang, Yi, 223
Zhao, Bing, 626
Zhao, Shanheng, 1308
Zhao, Shiqi, 1317, 1326
Zhao, Yinggong, 662
Zhou, Guodong, 599, 635, 671
Zhou, Ming, 295, 304, 313, 474, 662, 698
Zhou, Yiping, 1335
Zhou, Yu, 447
Zhu, Jingbo, 1344
Zhu, Muhua, 1344
Zhu, Qiaoming, 599, 671
Zhu, Xiaoyan, 116, 653
Zhu, Zhemin, 1353
Zhuang, Tao, 1362
Zong, Chengqing, 447, 1173, 1362
Zukerman, Ingrid, 62
Zwarts, Simon, 1371

Testing SDRT's Right Frontier

Stergos D. Afantenos and **Nicholas Asher**
Institut de recherche en informatique de Toulouse (IRIT),
CNRS, Université Paul Sabatier
{stergos.afantenos, nicholas.asher}@irit.fr

Abstract

The Right Frontier Constraint (RFC), as a constraint on the attachment of new constituents to an existing discourse structure, has important implications for the interpretation of anaphoric elements in discourse and for Machine Learning (ML) approaches to learning discourse structures. In this paper we provide strong empirical support for SDRT's version of RFC. The analysis of about 100 doubly annotated documents by five different naive annotators shows that SDRT's RFC is respected about 95% of the time. The qualitative analysis of presumed violations that we have performed shows that they are either click-errors or structural misconceptions.

1 Introduction

A cognitively plausible way to view the construction of a discourse structure for a text is an incremental one. Interpreters integrate discourse constituent n into the antecedently constructed discourse structure D for constituents 1 to $n-1$ by linking n to some constituent in D with a discourse relation. SDRT's Right Frontier Constraint (RFC) (Asher, 1993; Asher and Lascarides, 2003) says that a new constituent n cannot attach to an arbitrary node in D. Instead it must attach to either the last node entered into the graph or one of the nodes that dominate this last node. Assuming that the last node is usually found on the right of the structure, this means that the nodes available for attachment occur on the *right frontier* (RF) of the discourse *graph* or SDRS.

Researchers working in different theoretical paradigms have adopted some form of this constraint. Polanyi (1985; 1988) originally proposed the RFC as a constraint on antecedents to anaphoric pronouns. SDRT generalizes this to a condition on all anaphoric elements. As the attachment of new information to a contextually given discourse graph in SDRT involves the resolution of an anaphoric dependency, RFC furnishes a constraint on the attachment problem. (Webber, 1988; Mann and Thompson, 1987; 1988) have also adopted versions of this constraint. But there are important differences. While SDRT and RST both take RFC as a constraint on all discourse attachments (in DLTAG, in contrast, anaphoric discourse particles are not limited to finding an antecedent on the RF), SDRT's notion of RF is substantially different from that of RST's or Polanyi's, because SDRT's notion of a RF depends on a 2-dimensional discourse graph built from *coordinating* and *subordinating* discourse relations. Defining RFC with respect to SDRT's 2-dimensional graphs allows the RF to contain discourse constituents that do not include the last constituent entered into the graph (in contrast to RST). SDRT also allows for multiple attachments of a constituent to the RFC.

SDRT's RFC has important implications for the interpretation of various types of anaphoric elements: tense (Lascarides and Asher, 1993), ellipsis (Hardt et al., 2001; Hardt and Romero, 2004; Asher, 2007), as well as pronouns referring to individuals and abstract entities (Asher, 1993; Asher and Lascarides, 2003). The RFC, we believe, will also benefit ML approaches to learning discourse structures, as a constraint limiting the search space for possible discourse attachments. Despite its importance, SDRT's RFC has never been empirically validated, however. We present evidence in this paper providing strong empirical support for SDRT's version of the constraint. We have chosen to study SDRT's notion of a RF, because of SDRT's greater expressive power over RST (Danlos, 2008), the greater generality of SDRT's defi-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1–9,
Beijing, August 2010

nition of RFC, and because of SDRT's greater theoretical reliance on the constraint for making semantic predictions. SDRT also makes theoretically clear why the RFC should apply to discourse relation attachment, since it treats discourse structure construction as a dynamic process in which all discourse relations are essentially anaphors. The analysis of about 100 doubly annotated documents by five different naive annotators shows that this constraint, as defined in SDRT, is respected about 95% of the time. The qualitative analysis of the presumed violations that we have performed shows that they are either click-errors or structural misconceptions by the annotators.

Below, we give a formal definition of SDRT's RFC; section 3 explains our annotation procedure. Details of the statistical analysis we have performed are given in section 4, and a qualitative analysis is provided in section 5. Finally, section 6 presents the implications of the empirical study for ML techniques for the extraction of discourse structures while sections 7 and 8 present the related work and conclusions.

2 The Right Frontier Constraint in SDRT

In SDRT, a discourse structure or SDRS (Segmented Discourse Representation Structure) is a tuple $< A, \mathcal{F}, \text{LAST} >$, where A is the set of labels representing the discourse constituents of the structure, $\text{LAST} \in A$ the last introduced label and $\mathcal{F}$ a function which assigns each member of A a well-formed formula of the SDRS language (defined (Asher and Lascarides, 2003, p 138)). SDRSs correspond to λ expressions with a continuation style semantics. SDRT distinguishes coordinating and subordinating discourse relations using a variety of linguistic tests (Asher and Vieu, 2005),[1] and isolates structural relations (Parallel and Contrast) based on their semantics.

The RF is the set of available attachment points to which a new utterance can be attached. What this set includes depends on the discourse relation used to make the attachment. Here is the definition from (Asher and Lascarides, 2003, p 148).

> Suppose that a constituent β is to be attached to a constituent in the SDRS with a discourse relation other than Parallel or Contrast. Then the available attachment points for β are:
>
> 1. The label $\alpha = \text{LAST}$;
> 2. Any label γ such that:
>   (a) *i-outscopes*(γ, α) (*i.e.* $R(\delta, \alpha)$ or $R(\alpha, \delta)$ is a conjunct in $\mathcal{F}(\gamma)$ for some R and some δ); or
>   (b) $R(\gamma, \alpha)$ is a conjunct in $\mathcal{F}(\lambda)$ for some label λ, where R is a subordinating discourse relation.
>
>   We gloss this as $\alpha < \gamma$.
> 3. Transitive Closure:
>   Any label γ that dominates α through a sequence of labels $\gamma_1, \gamma_2, \dots \gamma_n$ such that $\alpha < \gamma_1 < \gamma_2 < \dots \gamma_n < \gamma$

We can represent an SDRS as a graph $\mathcal{G}$, whose nodes are the labels of the SDRSs constituents and whose typed arcs represent the relations between them. The nodes available for attachment of a new element β in $\mathcal{G}$ are the last introduced node LAST and any other node dominating LAST, where the notion of domination should be understood as the transitive closure over the arrows given by *subordinating* relations or those holding between a complex segment and its parts. Subordinating relations like *Elaboration* extend the vertical dimension of the graph, whereas coordinating relations like *Narration* expand the structure horizontally. The graph of every SDRS has a unique top label for the whole structure or formula; however, there may be multiple $<$ paths defined within a given SDRS, allowing for multiple parents, in the terminology of (Wolf and Gibson, 2006). Furthermore, SDRT allows for multiple arcs between constituents and attachments to multiple constituents on the RFC, making for a very rich structure.

SDRT's RFC is restricted to non-structural relations, because structural relations postulate a partial isomorphism from the discourse structure of the second constituent to the discourse structure of the first, which provides its own attachment possibilities for subconstituents of the two related structures (Asher, 1993). Sometimes such parallelism or contrast, also known as *discourse subordination* (Asher, 1993), can be enforced in a long

[1] The subordinating relations of SDRT are currently: Elaboration (a relation defined in terms of the main eventualities of the related constituents), Entity-Elaboration (E-Elab(a,b) iff b says more about an entity mentioned in a that is not the main eventuality of a) Comment, Flashback (the reverse of Narration), Background, Goal (intentional explanation), Explanation, and Attribution. The coordinating relations are: Narration, Contrast, Result, Parallel, Continuation, Alternation, and Conditional, all defined in Asher and Lascarides (2003).

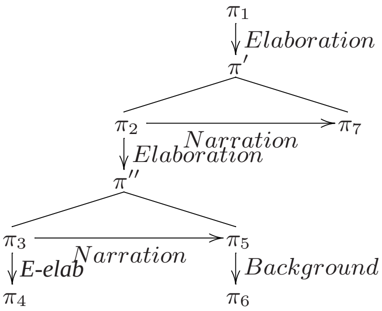
distance way by repeating the same wording in the two constituents.

RFC has the name it does because the segments that belong on this set (the γs in the above definition) are typically nodes on a discourse graph which are geometrically placed at the RF of the graph. Consider the following example embellished from Asher and Lascarides (2003):

(1) (π_1) John had a great evening last night. (π_2) He first had a great meal at Michel Sarran. (π_3) He ate profiterolles de foie gras, (π_4) which is a specialty of the chef. (π_5) He had the lobster, (π_6) which he had been dreaming about for weeks. (π_7) He then went out to a several swank bars.

The graph of the SDRS for 1 looks like this:

(2)

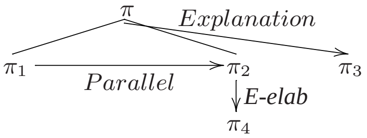


where π' and π'' represent complex segments. Given that the last introduced utterance is represented by the node π_7, the set of nodes that are on the RF are π_7 (LAST), π' (the complex segment that includes π_7) and π_1 (connected via a subordinating relation to π'). All those nodes are geometrically placed at the RF of the graph.

SDRT's notion of a RF is more general than RST's or DLTAG's. First, SDRSs can have complex constituents with multiple elements linked by coordinate relations that serve as arguments to other relations, thus permitting instances of *shared structure* that are difficult to capture in a pure tree notation (Lee et al., 2008). In addition, in RST the RF picks out the *adjacent* constituents, LAST and complex segments including LAST. Contrary to RST, SDRT, as it uses 2-dimensional graphs, predicts that an available attachment point for π_7 is the non local and non adjacent π_2, which is distinct from the complex constituent consisting of π_2 to π_6.[2] This difference is crucial to the interpretation of the Narration: Narration claims a sequence of two events; making the complex constituent (essentially a sub-SDRS) an argument of Narration, as RST does, makes it difficult to recover such an interpretation. Danlos's (2008) interpretation of the Nuclearity Principle provides an interpretation of the Narration([2-4],5) that is equivalent to the SDRS graph above.[3] But even an optional Nuclearlity Principle interpretation won't help with discourse structures like (2) where the backgrounding material in π_4 and the commentary in π_6 do not and cannot figure as part of the Elaboration for semantic reasons. In our corpus described below, over 20% of the attachments were non adjacent; *i.e.* the attachment point for the new material did not include LAST.

A further difference between SDRT and other theories is that, as SDRT's RFC is applied recursively over complex segments within a given SDRS, many more attachment points are available in SDRT. E.g., consider the SDRS for this example, adapted from (Wolf and Gibson, 2006):

(3) (π_1) Mary wanted garlic and thyme. (π_2) She also needed basil. (π_3) The recipe called for them. (π_4) The basil would be hard to come by this time of year.



Because π is the complex segment consisting of π_1 and π_2, attachment to π with a subordinating discourse relation permits attachment π's open constituents as well.[4]

3 Annotated Corpus

Our corpus comes from the discourse structure annotation project ANNODIS[5] which represents an on going effort to build a discourse graph bank for French texts with the two-fold goal of testing various theoretical proposals about discourse

[2] The 2-dimensionality of SDRSs also allows us to represent many examples with Elaboration that involve crossing dependencies in Wolf and Gibson's (2006) representation without violation of the RFC.

[3] Baldridge et al. (2007), however, show that the Nuclearity Principle does not always hold.

[4] This part of the RFC was not used in (Asher and Lascarides, 2003).

[5] `http://w3.erss.univ-tlse2.fr/annodis`

structure and providing a seed corpus for learning discourse structures using ML techniques. ANNODIS's annotation manual provides detailed instructions about the segmentation of a text into Elementary Discourse Units (EDUs). EDUs correspond often to clauses but are also introduced by frame adverbials,[6] appositive elements, correlative constructions (*[the more you work,] [the more you earn]*), interjections and discourse markers within coordinated VPs *[John denied the charges] [but then later admitted his guilt]*. Appositive elements often introduce *embedded* EDUs; e.g., *[Jim Powers, [President of the University of Texas at Austin], resigned today.]*, which makes our segmentation more fine-grained than Wolf and Gibson's (2006) or annotation schemes for RST or the PDTB.

The manual also details the meaning of discourse relations but says nothing about the structural postulates of SDRT. For example, there is no mention of the RFC in the manual and very little about hierarchical structure. Subjects were told to put whatever discourse relations from our list above between constituents they felt were appropriate. They were also told that they could group constituents together whenever they felt that as a whole they jointly formed the term of a discourse relation. We purposely avoided making the manual too restrictive, because one of our goals was to examine how well SDRT predicts the discourse structure of subjects who have little knowledge of discourse theories.

In total 5 subjects with little to no knowledge of discourse theories that use RFC participated in the annotation campaign. Three were undergraduate linguistics students and two were graduate linguistics students studying different areas. The 3 undergraduates benefitted from a completed and revised annotation manual. The two graduate students did their annotations while the annotation manual was undergoing revisions. All in all, our annotators doubly annotated about 100 French newspaper texts and *Wikipedia* articles. Subjects first segmented each text into EDUs, and then they were paired off and compared their segmentations, resolving conflicts on their own or via a supervisor. The annotation of the discourse relations was performed by each subject working in isolation. ANNODIS provided a new state of the art tool, GLOZZ, for discourse annotation for the three undergraduates. With GLOZZ annotators could isolate sections of text corresponding to several EDUs, and insert relations between selected constituents using the mouse. Though it did portray relations selected as lines between parts of the text, GLOZZ did not provide a discourse graph or SDRS as part of its graphical interface. The representation often yielded a dense number of lines between segments that annotators and evaluators found hard to read. The inadequate interline spacing in GLOZZ also contributed to certain number of click errors that we detail below in the paper. The statistics on the number of documents, EDUs and relations provided by each annotator are in table 1.

annotator	*# Docs*	*# EDUs*	*# Relations*
undergrad 1	27	1342	1216
undergrad 2	31	1378	1302
undergrad 3	31	1376	1173
grad 1	47	1387	1390
grad 2	48	1314	1321

Table 1: Statistics on documents, EDUs and Relations.

4 Experiments and Results

Using ANNODIS's annotated corpus, we checked for all EDUs π, whether π was attached to a constituent in the SDRS built from the previous EDUs in a way that violated the RFC. Given a discourse as a series of EDUS $\pi_1, \pi_2, \ldots, \pi_n$, we constructed for each π_i the corresponding sub-graph and calculated the set of nodes on the RF of this sub-graph. We then checked whether the EDU π_{i+1} was attached to a node that was found in this set. We also checked whether any newly created complex segment was attached to a node on the RF of this sub-graph.

4.1 Calculating the Nodes at the RF

To calculate the nodes on the RF, we slightly extended the annotated graphs, in order to add im-

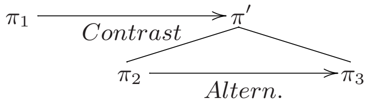
[6] Frame adverbials are sentence initial adverbial phrases that can either be temporal, spatial or "topical" (*in Chemistry*).

plied relations left out by the annotators.[7]

Disconnected Graphs While checking the RFC for the attachment of a node n, the SDRS graph at this point might consist of 2 or more disjoint subgraphs which get connected together at a later point. Because we did not want to decide which way these graphs should be connected, we defined a right frontier for each one using its own LAST. We then calculated the RF for each one of them and set the set of available nodes to be those in the union of the RFs of the disjoint subgraphs. If the subgraphs were not connected at the end of the incremental process in a way that conformed to RFC, we counted this as a violation. Annotators did not always provide us with a connected graph.

Postponed Decisions SDRT allows for the attachment not only of EDUs but also of subgraphs to an available node in the contextually given SDRS. For instance, in the following example, the intended meaning is given by the graph in which the Contrast is between the first label and the complex constituent composed of the disjunction of π_2 and π_3.

(π_1) Bill doesn't like sports. (π_2) But Sam does. (π_3) Or John does.



Naive annotators attached subgraphs instead of EDUs to the RF with some regularity (around 2%). This means that an EDU π_{i+1} could be attached to a node that was not present in the subgraph produced by $\pi_1, \ldots, \pi_i$. There were two main reasons for this: (1) π_{i+1} came from a syntactically fronted clause, a parenthetical or apposition in a sentence whose main clause produced π_{i+2} and π_{i+1} was attached to π_{i+2}; (2) π_{i+1} was attached to a complex segment $[\ldots, \pi_{i+1}, \ldots, \pi_{i+k}, \ldots]$ which was not yet introduced in the subgraph.

Since the nodes to which π_{i+1} is attached in such cases are not present in the graph, *by definition* they are not in the RF and they could be counted as violations. Nonetheless, if the nodes which connect nodes like π_{i+1} eventually link up to the incrementally built SDRS in the right way, π_{i+1} might eventually end up linked to something on the RF. For this reason, we postponed the decision on nodes like π_{i+1} until the nodes to which they are attached were explicitly introduced in the SDRS.

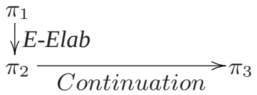
The Coherence of Complex Segments In an SDRS, several EDUs may combine to form a complex segment α that serves as a term for a discourse relation R. The interpretation of the SDRS implies that all of α's constituents contribute to the rhetorical function specified by R. This implies that the coordinating relation *Continuation* holds between the EDUs inside α, unless there is some other relation between them that is incompatible with Continuation (like a subordinating relation). Continuations are often used in SDRT (Asher, 1993; Asher and Lascarides, 2003). During the annotation procedure, our subjects did not always explicitly link the EDUs within a complex segment. In order to enforce the coherence of those complex segments we added *Continuation* relations between the constituents of a complex segment *unless* there was already another path between those constituents.

Expanding Continuations Consider the following discourse:

(4) [John, [who owns a chain of restaurants]$_{\pi_2}$, [and is a director of a local charity organization,]$_{\pi_3}$ wanted to sell his yacht.]$_{\pi_1}$ [He couldn't afford it anymore.]$_{\pi_4}$

Annotators sometimes produced the following SDRT graph for the first three EDUs of this discourse:

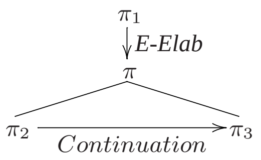
(5)



In this case the only open node is π_3 due to the coordinating relation *Continuation*. Nonetheless, π_4 should be attached to π_1, without violating the RFC. Indeed, SDRT's definition of the Continuation relation enforces that if we have $R(\pi_1, \pi_2)$ and Continuation(π_2, π_3) then we actually have the complex segment $[\pi_2, \pi_3]$ with $R(\pi_1, [\pi_2, \pi_3])$. So there is in fact a missing complex segment in (5). The proper SDRS graph of (4) is:

[7] In similar work on TimeML annotations, Setzer et al. (2003; Muller and Raymonet (2005) add implied relations to annotated, temporal graphs.

(6)



which makes π_1 an available attachment site for π_4. Such implied constituents have been added to the SDRS graphs.

Factoring Related to the operation of Expansion, SDRT's definition of Continuation and various subordinating relations also requires that if we have $R(a, [\pi_1, \pi_2, \ldots, \pi_n])$ where $[\pi_1, \pi_2, \ldots, \pi_n]$ is a complex segment with $\pi_1, \ldots \pi_n$ linked by Continuation and R is Elaboration, Entity-Elaboration, Frame, Attribution, or Commentary, then we also have $R(a, \pi_i)$ for each i. We added these relations when they were missing.

4.2 Results

With the operations just described, we added several inferred relations to the graph. We then calculated statistics concerning the percentage of attachments for which the RFC is respected using the following formula:

$$RFC_{\text{EDU}} = \frac{\texttt{\# EDUs attached to the RF}}{\texttt{\# EDUs in total}}$$

As we explained, an EDU can be attached to an SDRT graph directly by itself or indirectly as part of a bigger complex segment. In order to calculate the nominator we determine first whether an EDU directly attaches to the graph's RF, and if that fails we determine whether it is part of a larger complex segment which is attached to the graph's RF. The results obtained are shown in the first two columns of table 2. The RFC is respected by at least some attachment decision 95% of the time—i.e., 95% of the EDUs get attached to another node that is found on the RF. The breakdown across our annotators is given in table 2.

SDRT allows for multiple attachments of an EDU to various nodes in an SDRS; *e.g.* while an EDU may be attached via one relation to a node on the RF, it may be attached to another node off the RF. To take account of all the attachments for a given EDU, we need another way of measuring the percentage of attachments that respects the RFC. So we counted the ways each EDU is related to a node in the SDRS for the previous text and then divided the number of attachment decisions that respect the RFC by the total number of attachment decisions—*i.e.* :

$$RFC_r = \frac{\texttt{\# RF attachment decisions}}{\texttt{\# Total attachment decisions}}$$

.

annotator	RFC_{EDU}	RFC_r
undergrad 1	98.57%	91.28%
undergrad 2	98.12%	94.39%
undergrad 3	91.93%	89.17%
grad 1	94.38%	86.54%
grad 2	92.68%	83.57%
Mean for all annotators	95.24%	88.91%
Mean for 3 undergrad	96.17%	91.71%

Table 2: The % with which each annotator has respected SDRT's RFC using the EDU and attachment decision measures.

The third column of table 2 shows that having a stable annotation manual and GLOZZ improved the results across our two annotator populations, even though the annotation manual did not say anything about RFC or about the structure of the discourse graphs. Moreover, the distribution of violations of the RFC follows a power law and only 4.56% of the documents contained more than 5 violations. This is strong evidence that there is little propagation of violations.

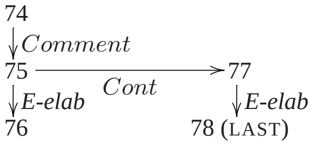
5 Analysis of Presumed Violations

Although 95% of EDUs attach to nodes on the RF of an SDRT graph, 5% of EDUs don't. SDRT experts performed a qualitative analysis of some of these presumed violations. In many cases, the experts judged that the presumed violations were due to click-errors: sometimes the annotators simply clicked on something that did not translate into a segment. Sometimes, the experts judged that the annotators picked the wrong segment to attach a new segment or the wrong type of relation during the construction of the SDRT graph. For example, in the graph that follows the relation between segments 74 and 75 is not a *Comment* but an *Entity-Elaboration*.

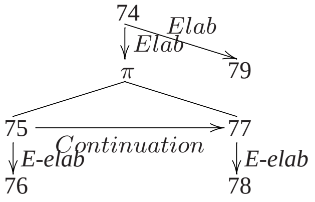
As expected, there were also *"structural"* errors, arising from a lack or a misuse of complex segments. Here is a typical example (translated from the original French):

> [Around her,]_74 [we should mention Joseph Racaille]_75 [responsible for the magnificent arrangements,]_76 [Christophe Dupouy]_77 [regular associate of Jean-Louis Murat responsible for mixing,]_78 [without forgetting her two guardian angels:]_79 [her agent Olivier Gluzman]_80 [who signed after a love at first sight,]_81 [and her husband Mokhtar]_82 [who has taken care of the family]_83

Here is the annotated structure up to EDU 78:

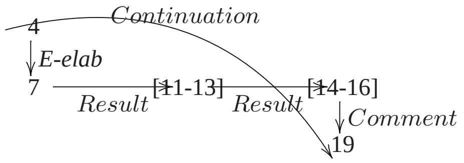


Note that the attachment of 77 to 75 is non-local and non-adjacent. The annotator then attaches EDU 79 to 75 which is blocked from the RF due to the $Continuation$ coordinating relation. By not having created a complex segment due the enumeration that includes EDUs 75 to 78, the annotator had no option but to violate the RF. Here is the proper SDRT graph for segments 74 to 79 (where the attachment of 79 to 74 is also both non-local and non-adjacent):



In this case, before the introduction of EDU 79, EDU 78 is LAST and by consequence $77, \pi$ and 74 are on the RF. Attaching 79 to 74 is thus legitimate.

We also found more interesting examples of right frontier violations. One annotator produced a graph for a story which is about the attacks of 9/11/2001 and is too long to quote here. A simplified graph of the first part of the story is shown below. EDU 4 elaborates on the main event of the story but it is not on the RF for 19. However, 19 is the first recurrence of the complex definite description *le 11 septembre 2001* since the title and the term's definition in EDU 4.



This reuse of the full definite description could be considered a case of SDRT's discourse subordination.

6 RFC and distances of attachment

Our empirical study vindicates SDRT's RFC, but it also has computational implications. Using the RFC dramatically diminishes the number of attachment possibilities and thus greatly reduces the search space for any incremental discourse parsing algorithm.[8] The mean of nodes that are open on the RF at any given moment on our ANNODIS data is 16.43% of all the nodes in the graph.

Our data also allowed us to calculate the distance of attachment sites from LAST, which could be an important constraint on machine learning algorithms for constructing discourse structures. Given a pair of constituents (π_i, π_j) distance is calculated either *textually* (the number of intervening EDUs between π_i and π_j) or *topologically* (the length the shortest path between π_i and π_j). Topological distance, however, does not take into account the fact that a textually further segment is cognitively less salient. Moreover, this measure can give the same distance to nodes that are textually far away between them due to long distance pop-ups (Asher and Lascarides, 2003). A purely textual distance, on the other hand, gives the same distance to an EDU π_i and a complex segment $[\pi_1, \ldots, \pi_i]$ even if π_1 and π_i are textually distant (since both have the same span end). We used a measure combining both. The distance scheme that we used assigns to each EDU its textual distance from LAST in the graph under consideration, while a complex segment of rank 1 gets a distance which is computed from the highest distance of their constituent EDUs plus 1. For a constituent σ of rank n we have:

$$Dist = Max\{\text{dist}(x) \colon x \text{ in } \sigma\} + n$$

[8] An analogous approach for search space reduction is followed by duVerle and Prendinger (2009) who use the "Principle of Sequentiality" (Marcu, 2000), though they do not say how much the search space is reduced.

The distribution of attachment follows a power law with 40% of attachments performed non-locally, that is on segments of distance 2 or more (figure 1). This implies that the distance between candidate attachment sites that are on the RF is an important feature for an ML algorithm. It is important to note at this point that following the baseline approach of always attaching on the LAST misses 40% of attachments. We also have 20.38% of the non-local, non-adjacent attachments in our annotations. So an RST parser using Marcu's (2000) adjacency constraint as do duVerle and Prendinger (2009) would miss these.

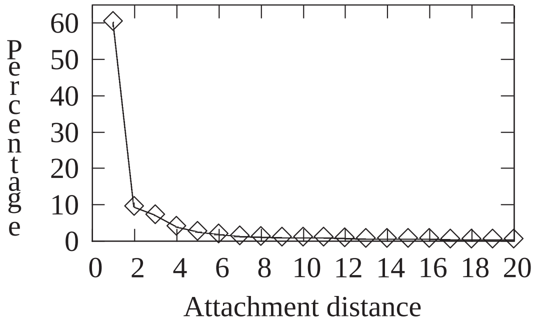


Figure 1: Distribution of attachment distance

7 Related Work

Several studies have shown that the RFC may be violated as an anaphoric constraint when there are other clues, content or linguistic features, that determine the antecedent. (Poesio and di Eugenio, 2001; Holler and Irmen, 2007; Asher, 2008; Prévot and Vieu, 2008), for example, show that anaphors such as definite descriptions and complex demonstratives, which often provide enough content on their own to isolate their antecedents, or pronouns in languages like German which must obey gender agreement, might remain felicitous although the discourse relations between them and their antecedents might violate the RFC. Usually there are few linguistic clues that help find the appropriate antecedent to a discourse relation, in contrast to the anaphoric expressions mentioned above. Exceptions involve stylistic devices like direct quotation that license discourse subordination. Thus, SDRT predicts that RFC violations for discourse attachments should be much more rare than those for the resolution of anaphors that provide linguistic clues about their antecedents.

As regards other empirical validation of various versions of the RFC for the attachment of discourse constituents, Wolf and Gibson (2006) show an RST-like RFC is not supported in their corpus GraphBank. Our study concurs in that some 20% of the attachments in our corpus cannot be formulated in RST.[9] On the other hand, we note that because of the 2 dimensional nature of SDRT graphs and because of the caveats introduced by structural relations and discourse subordination, the counterexamples from GraphBank against, say, RST representations do not carry over straightforwardly to SDRSs. In fact, once these factors are taken into account, the RFC violations in our corpus and in GraphBank are roughly about the same.

8 Conclusions

We have shown that SDRT's RFC has strong empirical support: the attachments of our 3 completely naive annotators fully comply with RFC 91.7% of the time and partially comply with it 96% of the time. As a constraint on discourse parsing SDRT's RFC, we have argued, is both empirically and computationally motivated. We have also shown that non-local attachments occur about 40% of the time, which implies that attaching directly on the LAST will not yield good results. Further, many of the non local attachments do not respect RST's adjacency constraint. We need SDRT's RFC to get the right attachment points for our corpus. We believe that empirical studies of the kind we have given here are essential to finding robust and useful features that will vastly improve discourse parsers.

[9] One other study we are aware of is Sassen and Kühnlein (2005), who show that in chat conversations, the RFC does not always hold unconditionally. Since this genre of discourse is not always coherent, it is expected that the RFC will not always hold here.

References

Asher, N. and A. Lascarides. 2003. *Logics of Conversation*. Studies in Natural Language Processing. Cambridge University Press, Cambridge, UK.

Asher, N. and L. Vieu. 2005. Subordinating and coordinating discourse relations. *Lingua*, 115(4):591–610.

Asher, N. 1993. *Reference to Abstract Objects in Discourse*. Kluwer Academic Publishers.

Asher, N. 2007. A large view of semantic content. *Pragmatics and Cognition*, 15(1):17–39.

Asher, N. 2008. Troubles on the right frontier. In Benz, A. and P. Kühnlein, editors, *Constraints in Discourse*, Pragmatics and Beyond New Series, chapter 2, pages 29–52. John Benjamins Publishing Company.

Baldridge, J., N. Asher, and J. Hunter. 2007. Annotation for and robust parsing of discourse structure on unrestricted texts. *Zeitschrift fur Sprachwissenschaft*, 26:213–239.

Danlos, L. 2008. Strong generative capacity of rst, sdrt and discourse dependency dags. In Benz, A. and P. Kühnlein, editors, *Constraints in Discourse*, Pragmatics and Beyond New Series, pages 69–95. John Benjamins Publishing Company.

duVerle, D. and H. Prendinger. 2009. A novel discourse parser based on support vector machine classification. In *Proceedings of ACL*, pages 665–673, Suntec, Singapore, August.

Hardt, D. and M. Romero. 2004. Ellipsis and the structure of discourse. *Journal of Semantics*, 21:375–414, November.

Hardt, D., N. Asher, and J. Busquets. 2001. Discourse parallelism, scope and ellipsis. *Journal of Semantics*, 18:1–16.

Holler, A. and L. Irmen. 2007. Empirically assessing effects of the right frontier constraint. In *Anaphora: Analysis, Algorithms and Applications*, pages 15–27. Springer, Berlin/Heidelberg.

Lascarides, A. and N. Asher. 1993. Temporal interpretation, discourse relations and commonsense entailment. *Linguistics and Philosophy*, 16(5):437–493.

Lee, A., R. Prasad, A. Joshi, and B. Webber. 2008. Departures from tree structures in discourse: Shared arguments in the penn discourse treebank. In *Constraints in Discourse (CID '08)*, pages 61–68.

Mann, W. and S. Thompson. 1987. Rhetorical structure theory: A framework for the analysis of texts. Technical Report ISI/RS-87-185, Information Sciences Institute, Marina del Rey, California.

Mann, W. and S. Thompson. 1988. Rhetorical structure theory: Towards a functional theory of text organization. *Text*, 8(3):243–281.

Marcu, D. 2000. *The Theory and Practice of Discourse Parsing and Summarization*. The MIT Press.

Muller, P. and A. Raymonet. 2005. Using inference for evaluating models of temporal discourse. In *12th International Symposium on Temporal Representation and Reasoning*, pages 11–19. IEEE Computer Society Press.

Poesio, M. and B. di Eugenio. 2001. Discourse structure and anaphoric accessibility. In *Proc. of the ESSLLI Workshop on Discourse Structure and Information Structure*, August.

Polanyi, L. 1985. A theory of discourse structure and discourse coherence. In Kroeber, P. D., W. H. Eilfort, and K. L. Peterson, editors, *Papers from the General Session at the 21st Regional Meeting of the Chicago Linguistics Society*.

Polanyi, L. 1988. A formal model of the structure of discourse. *Journal of Pragmatics*, 12:601–638.

Prévot, L. and L. Vieu. 2008. The moving right frontier. In Benz, A. and P. Kühnlein, editors, *Constraints in Discourse*, Pragmatics and Beyond New Series, chapter 3, pages 53–66. John Benjamins Publishing Company.

Sassen, C. and P. Kühnlein. 2005. The right frontier constraint as conditional. In *Computational Linguistics and Intelligent Text Processing*, Lecture Notes in Computer Science (LNCS), pages 222–225.

Setzer, A., R. Gaizauskas, and M. Hepple. 2003. Using semantic inferences for temporal annotation comparison. In *Proceedings of the Fourth International Workshop on Inference in Computational Semantics (ICoS-4)*.

Webber, B. 1988. Title discourse deixis and discourse processing. Technical Report MS-CIS-88-75, University of Pennsylvania, Department of Computer and Information Science, September.

Wolf, F. and E. Gibson. 2006. *Coherence in Natural Language: Data Stuctures and Applications*. The MIT Press.

Identifying Multi-word Expressions by Leveraging Morphological and Syntactic Idiosyncrasy

Hassan Al-Haj
Language Technologies Institute
Carnegie Mellon University
hhaj@cs.cmu.edu

Shuly Wintner
Department of Computer Science
University of Haifa
shuly@cs.haifa.ac.il

Abstract

Multi-word expressions constitute a significant portion of the lexicon of every natural language, and handling them correctly is mandatory for various NLP applications. Yet such entities are notoriously hard to define, and are consequently missing from standard lexicons and dictionaries. Multi-word expressions exhibit idiosyncratic behavior on various levels: orthographic, morphological, syntactic and semantic. In this work we take advantage of the morphological and syntactic idiosyncrasy of Hebrew noun compounds and employ it to extract such expressions from text corpora. We show that relying on linguistic information dramatically improves the accuracy of compound extraction, reducing over one third of the errors compared with the best baseline.

1 Introduction

Multi-word expressions (MWEs) are notoriously hard to define. They span a range of constructions, from completely frozen, semantically opaque idiomatic expressions, to frequent but morphologically productive and semantically compositional collocations. Various linguistic processes (orthographic, morphological, syntactic, semantic, and cognitive) apply to MWEs in idiosyncratic ways. Notably, MWEs blur the distinction between the lexicon and the grammar, since they often have some properties of words and some of phrases.

In this work we define MWEs as expressions whose linguistic properties (morphological, syntactic or semantic) are not directly derived from the properties of their word constituents. This is a functional definition, driven by a practical motivation: any natural language processing (NLP) application that cares about morphology, syntax or semantics must consequently store MWEs in the lexicon.

MWEs are numerous and constitute a significant portion of the lexicon of any natural language. They are a heterogeneous class of constructions with diverse sets of characteristics. Morphologically, some MWEs allow some of their constituents to freely inflect while restricting (or even preventing) the inflection of other constituents. MWEs may allow constituents to undergo non-standard morphological inflections that they would not undergo in isolation. Some MWEs contain words that never occur outside the context of the MWE. Syntactically, some MWEs appear in one rigid pattern (and a fixed order), while others permit various syntactic transformations. Semantically, the compositionality of MWEs (i.e., the degree to which the meaning of the whole expression results from combining the meanings of its individual words when they occur in isolation) is gradual.

These morphological, syntactic and semantic idiosyncrasies make MWEs a challenge for NLP applications (Sag et al., 2002). They are even more challenging in languages with complex morphology, because of the unique interaction of morphological and orthographic processes with the lexical specification of MWEs (Oflazer et al., 2004; Alegria et al., 2004).

Because the idiosyncratic features of MWEs cannot be predicted on the basis of their component words, they must be stored in the lexicon of NLP applications. Handling MWEs correctly is beneficial for a variety of applications, including information retrieval, building ontologies, text alignment, and machine translation. Automatic identification and corpus-based extraction of MWEs is thus crucial for such (and several other) applications.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 10–18,
Beijing, August 2010

In this work we describe an approach that leverages the morphological and syntactic idiosyncrasy of a certain class of Hebrew[1] MWEs, namely noun compounds, to help identify such expressions in texts. While the main contribution of this work is a system that can distinguish between MWE and non-MWE instances of a particular construction in Hebrew, thereby facilitating faster and more accurate integration of MWEs in a large-coverage lexicon of the language, we believe that it carries added value to anyone interested in MWEs. The technique that we propose here should be applicable in principle to any language in which MWEs exhibit linguistically idiosyncratic behavior.

We describe the properties of Hebrew noun-noun constructions in Section 2, and specify the irregularities exhibited by compounds. Section 3 presents the experimental setup and the main results. Compared with the best (collocation-based) baseline, our approach reduces over 30% of the errors, yielding accuracy of over 80%. We discuss related work in Section 4 and conclude with suggestions for future research.

2 Hebrew noun-noun constructions

We focus on Hebrew noun-noun constructions; these are extremely frequent constructions, and while many of them are fully compositional, others, called *noun compounds* (or just *compounds*) here, are clearly MWEs. We first discuss the general construction and then describe the peculiar, idiosyncratic properties of compounds.

2.1 The general case

Hebrew nouns inflect for number (singular and plural) and, when the noun denotes an animate entity, for gender (masculine and feminine). In addition, nouns come in three *states*: indefinite, definite and a *construct* state that is used in genitive constructions. Table 1 demonstrates the paradigm.

State	M/Sg	F/Sg	M/Pl	F/Pl
indefinite	*ild*	*ildh*	*ildim*	*ildwt*
definite	*hild*	*hildh*	*hildim*	*hildwt*
construct	*ild*	*ildt*	*ildi*	*ildwt*

Table 1: The noun paradigm, demonstrated on *ild* "child"

A noun-noun construction (henceforth NNC) consists of a construct-state noun, called *head* here, followed by a noun phrase, the *modifier* (Borer, 1988; Borer, 1996; Glinert, 1989). The semantic relation between the two is usually, but not always, related to possession (Levi, 1976). Construct-state nouns only occur in the context of NNC, and can never occur in isolation. When a NNC is definite, the definite article is expressed on its modifier (Wintner, 2000).

In the examples below, we explicitly indicate construct-state nouns by the morpheme '.CONST' in the gloss; and definite nouns are indicated by the morpheme 'the-'. We provide both a literal and a non-literal meaning of the MWE examples. Expressions that have a literal, but not the expected MWE meaning, are preceded by '#'.

Example 1 (Noun-noun constructions)

hxlTt *hw‘dh*
*decision.*CONST *the-committee*
"the committee decision"

‘wrk *h‘itwn*
*editor.*CONST *the-journal*
"the journal editor"

‘wrk *din*
*editor.*CONST *law*
"law editor" ⟹ *lawyer*

bti *xwlim*
*houses.*CONST *patients*
"patient houses" ⟹ *hospitals*

2.2 Noun compounds: Linguistic properties

While many of the NNCs are free, compositional combinations of words, some are not; we use the term *noun compounds* for the latter group. Compounds typically (but not necessarily) have non-compositional meaning; presumably due to their opaque, more lexical meaning, they also differ from other NNCs in their morphological and syntactic behavior. Some of these distinctive properties are listed below, to motivate the methodology that we propose in Section 3 to distinguish between compounds and non-MWE NNCs.

[1] To facilitate readability we use a transliteration of Hebrew using Roman characters; the letters used, in Hebrew lexicographic order, are *abgdhwzxTiklmns‘pcqršt*.

2.2.1 Limited inflection

When a NNC consists of two nouns, the second can typically occur in either singular or plural form. Compounds often limit the possibilities to only one of those.

Example 2 (No plural form of the modifier)

‘wrki	*h‘itwnim*
*editors-.*CONST	*the-journals*

“the journals’ editors”

‘wrki	*hdin*
*editors.*CONST	*the-law*

“the law editors” ⟹ the lawyers

#wrki	*hdinim*
*editors.*CONST	*the-laws*

Example 3 (No singular form of the modifier)

kiwwn	*hrwx*
*direction.*CONST	*the-wind*

“the wind’s direction”

kiwwn	*hrwxwt*
*direction.*CONST	*the-winds*

“the winds’ direction”

šwšnt	*h-rwxwt*
*lily.*CONST	*the-winds*

“lily of the winds” ⟹ compass rose

#šwšnt	*h-rwx*
*lily.*CONST	*the-wind*

2.2.2 Limited syntactic variation

Since NNCs typically denote genitive (possessive) constructions, they can be paraphrased by a construction that uses the genitive preposition *šl* “of” (or, in some cases, other prepositions). These syntactic variants are often restricted in the case of compounds.

Example 4 (Limited paraphrasing)

h‘wrk	*šl*	*h‘itwn*
the-editor	*of*	*the-journal*

“the journal editor”

#h‘wrk	*šl*	*hdin*
the-editor	*of*	*the-law*

Example 5 (Limited paraphrasing)

m‘il	*cmr*
*coat.*CONST	*wool*

“wool coat”

m‘il	*mcmr*
coat	*from-wool*

“wool coat”

cmr	*pldh*
*wool.*CONST	*steel*

“steel wool” ⟹ steel wool

#cmr	*mpldh*
wool	*from-steel*

2.2.3 Limited syntactic modification

NNCs typically allow adjectival modification of either of their constituents. Since compounds tend to be more semantically opaque, it is often only possible to modify the entire compound, but not any of the constituents. In the following example, note that *‘wrkt* “editor” is feminine, whereas *‘itwn* “journal” is masculine; adjectives must agree on gender with the noun they modify.

Example 6 (Limited adjectival modification)

’wrkt	*h’itwn*
*editor-f.*CONST	*the-journal-m*

“the journal editor”

‘wrkt	*h‘itwn*	*hxdšh*
*editor-f.*CONST	*the-journal-m*	*the-new-f*

“the new editor of the journal”

‘wrkt	*h‘itwn*	*hxdš*
*editor-f.*CONST	*the-journal-m*	*the-new-m*

“the editor of the new journal”

‘wrkt	*hdin*	*hxdšh*
*editor-f.*CONST	*the-law-m*	*the-new-f*

“the new law editor” ⟹ the new lawyer

#‘wrkt	*hdin*	*hxdš*
*editor-f.*CONST	*the-law-m*	*the-new-m*

2.2.4 Limited coordination

Two NNCs that share a common head can be conjoined using the coordinating conjunction *w* “and”. This possibility is often blocked in the case of compounds.

Example 7 (Limited coordination)

mwsdwt	*xinwk*	*wbriawt*
*institutions.*CONST	*education*	*and-health*

“education and health institutions”

bti	*spr*
*houses.*CONST	*book*

“book houses” ⟹ schools

bti	*xwlim*
*houses.*CONST	*patients*

"patient houses" ⟹ *hospitals*

#bti	*spr*	*wxwlim*
*houses.*CONST	*book*	*and-patients*

3 Identification of noun compounds

In this section we describe a system that identifies noun compounds in Hebrew text, and extracts them in order to extend the lexicon. We capitalize on the morphological and syntactic irregularities of noun compounds described in Section 2.2.

Given a large monolingual corpus, the text is first morphologically analyzed and disambiguated. Then, all NNCs (candidate noun compounds) are extracted from the morphologically disambiguated text. For each candidate noun compound we define a set of features (Section 3.3) based on the idiosyncratic morphological and syntactic properties defined in Section 2.2. These features inform a support vector machine classifier which is then used to identify the noun compounds in the set of NNCs with high accuracy (Section 3.5).

3.1 Resources

We use (a subset of) the Corpus of Contemporary Hebrew (Itai and Wintner, 2008) which consists of four sub-corpora: The *Knesset* corpus contains the Israeli parliament proceedings from 2004-2005; the *Haaretz* corpus contains articles from the Haaretz newspaper from 1991; *TheMarker* corpus contains financial articles from the TheMarker newspaper from 2002; and the *Arutz 7* corpus contains newswire articles from 2001-2006. Corpora sizes are listed in Table 2.

Corpus	Number of tokens
Knesset	12,742,879
Harretz	463,085
The Marker	684,801
Arutz 7	7,714,309
Total	21,605,074

Table 2: Corpus data

The entire corpus was morphologically analyzed (Yona and Wintner, 2008; Itai and Wintner, 2008) and POS-tagged (Bar-haim et al., 2008); note that no syntactic parser is available for Hebrew. From the morphologically disambiguated corpus, we extract all bi-grams in which the first token is a noun in the construct state and the second token is a noun that is not in the construct state, i.e., all two-word NNC *candidates*.

3.2 Annotation

For training and evaluation, we select the NNCs that occur at least 100 times in the corpus, yielding 1060 NNCs. These NNCs were annotated by three annotators, who were asked to classify them to the following four groups: compounds (+); non-compounds (–); unsure (0); and errors of the morphological processor (i.e., the candidate is not a NNC at all). Table 3 lists the number of candidates in each class.

Annotator	+	–	0	err
1	314	332	238	176
2	335	403	179	143
3	400	630	16	14

Table 3: NNC classification by annotator

We adopt a conservative approach in combining the three annotations. First, we eliminate 204 NNCs that were tagged as errors by at least one annotator. For the remaining NNCs, a candidate is considered a compound or a non-compound only if all three annotators agree on its classification. This reduces the annotated data to 463 instances, of which 205 are compounds and 258 are clear cases of non-compound NNCs.[2]

3.3 Linguistically-motivated features

We define a set of features based on the idiosyncratic properties of noun compounds defined in Section 2.2. For each candidate NNC, we compute counts which reflect the likelihood of it exhibiting one of the linguistic properties.

Refer back to Section 2.2. We focus on the property of limited inflection (Section 2.2.1), and define features 1–8 to reflect it. To reflect limited syntactic variation (Section 2.2.2) we define features 9–10. Feature 11 addresses the phenomenon

[2] This annotated corpus is freely available for download.

of limited coordination (Section 2.2.4). To reflect limited syntactic modification (Section 2.2.3) we define feature 12. .

For each NNC candidate N_1 N_2, the following features are defined:

1. The number of occurrences of the NNC in which both constituents are in singular.
2. The number of occurrences of the NNC in which N_1 is in singular and N_2 is in plural.
3. The number of occurrences of the NNC in which N_1 is in plural and N_2 is in singular.
4. The number of occurrences of the NNC in which both constituents are in plural.
5. The number of occurrences of N_1 in plural outside the expression.
6. The number of occurrences of N_1 in singular outside the expression.
7. The number of occurrences of N_2 in plural outside the expression.
8. The number of occurrences of N_2 in singular outside the expression.
9. The number of occurrences of N_1 *šl* N_2 "N_1 of N_2" in the corpus.
10. The number of occurrences of N_1 *m* N_2 "N_1 from N_2" in the corpus.
11. The number of occurrences of N_1 N_2 *w* N_3 "N_1 N_2 and N_3" in the corpus, where N_3 is an indefinite, non-construct-state noun.
12. The number of occurrences of N_1 N_2 *Adj* in the corpus, where the adjective *Adj* agrees with N_2 on both gender and number, while disagreeing with N_1 on at least one of these attributes.

We also define four features that represent known collocation measures (Evert and Krenn, 2001): Point-wise mutual information (PMI); T-Score; log-likelihood; and the raw frequency of N_1 N_2 in the corpus.[3]

3.4 Training and evaluation

For each NNC in the annotated set of Section 3.2 we create a vector of the 16 features described in Section 3.3 (12 linguistically-motivated features plus four collocation measures). We obtain a list of 463 instances, of which 205 are positive examples (noun compounds) and 258 are negative. We use this set for training and evaluation of a two class soft margin SVM classifier (Chang and Lin, 2001) with a radial basis function kernel. We experiment below with different combinations of features, where for each combination we use 10-fold cross-validation over the 463 NNCs to evaluate the classifier. We report Precision, Recall, F-score and Accuracy (averaged over the 10 folds).

3.5 Results

The results of the different classifiers that we trained are given in Table 4. The first four rows of the table show the performance of classifiers trained using each of the four different collocation measure features alone. Both PMI and Log-likelihood outperform the other collocation measures, with an F-score of 60, which we consider our baseline. We also report the performance of two combinations of collocation measures, which yield small improvement. The best combinations provide accuracy of about 70% and F-score of 63.

The remaining rows report results using the linguistically-motivated features (LMF) of Section 3.3. These features alone yield accuracy of 77.75% and an F-score of 76. Adding also Log-likelihood improves F-score by 1.16 and accuracy by 1.29%. Finally, using Log-likelihood with a subset of the LMF consisting of features 1-2, 4-6, 9-10 and 12 (see below) yields the best results, namely accuracy of over 80% and F-score of 78.85, reflecting a reduction of over one third in classification error rate compared with the baseline.

3.6 Optimizing feature combination

We search for the combination of linguistically-motivated features that would yield the best performance. Training a classifier on all possible feature combinations is clearly infeasible. Instead, we follow a more efficient greedy approach, whereby we start with the best collocation mea-

[3] A detailed description of these measures is given by Manning and Schütze (1999, Chapter 5); see also `http://www.collocations.de/`, where several other association measures are discussed as well.

Features	Accuracy	Precision	Recall	F-score
PMI	67.17	64.97	**56.09**	**60.20**
Frequency	60.47	60.00	32.19	41.90
T-Score	61.98	59.86	42.92	50.00
Log-likelihood	**69.33**	**71.42**	51.21	59.65
T-score+Log-likelihood	**70.62**	**71.42**	56.09	62.84
PMI+Log-likelihood	69.97	68.96	**58.53**	**63.32**
LMF	77.75	71.98	81.46	76.43
LMF+PMI	77.32	71.18	**81.95**	76.19
LMF+Log-likelihood	79.04	73.68	**81.95**	77.59
Log-likelihood+LMF[1-2,4-6,9-10,12]	**80.77**	**76.85**	80.97	**78.85**

Table 4: Results: 10-Fold accuracy, precision, recall, and F-score for classifiers trained using different combinations of features. *LMF* stands for linguistically-motivated features

sure, Log-likelihood, and add other features one at a time, in the order in which they are listed in Section 3.3. After adding each feature the classifier is retrained; the feature is retained in the feature set only if adding it improves the 10-fold F-score of the current feature set.

Table 5 lists the results of this experiment. For each feature set the difference in the 10-fold F-score compared to the previous feature set is listed in parentheses. The results show that the best feature combination improves the F-score by 1.26, compared with using all features. This experiments shows that features 3, 7, 8 and 11 turn out not to be useful, and the classifier is more accurate without them. We also tried this approach with PMI as the starting feature, with very similar results.

Feature set	F-score
Log-likelihood	59.65
Log-likelihood,1	60.34 (+0.68)
Log-likelihood,1-2	65.42 (+5.08)
Log-likelihood,1-3	64.87 (-0.54)
Log-likelihood,1-2,4	66.66 (+1.78)
Log-likelihood,1-2,4-5	70.00 (+3.33)
Log-likelihood,1-2,4-6	74.37 (+4.37)
Log-likelihood,1-2,4-7	73.78 (–0.58)
Log-likelihood,1-2,4-6,8	73.58 (–0.79)
Log-likelihood,1-2,4-6,9	78.72 (+4.35)
Log-likelihood,1-2,4-6,9-10	78.83 (+0.10)
Log-likelihood,1-2,4-6,9-11	77.37 (–1.46)
Log-likelihood,1-2,4-6,9-10,12	78.85 (+0.02)

Table 5: Optimizing the set of linguistically-motivated features

4 Related work

There has been a growing awareness in the research community of the problems that MWEs pose, both in linguistics and in NLP (Villavicencio et al., 2005). Recent works address the definition, lexical representation and computational processing of MWEs, as well as algorithms for extracting them from data.

Focusing on acquisition of MWEs, early approaches concentrated on their collocational behavior (Church and Hanks, 1989). Pecina (2008) compares 55 different association measures in ranking German Adj-N and PP-Verb collocation candidates. This work shows that combining different collocation measures using standard statistical-classification methods (such as Linear Logistic Regression and Neural Networks) gives a significant improvement over using a single collocation measure. Our results show that this is indeed the case, but the contribution of collocation methods is limited, and more information is needed in order to distinguish frequent collocations from bona fide MWEs.

Other works show that adding linguistic information to collocation measures can improve identification accuracy. Several approaches rely on the semantic opacity of MWEs; but very few semantic resources are available for Hebrew (the Hebrew WordNet (Ordan and Wintner, 2007), the only lexical semantic resource for this language, is small and too limited). Instead, we capitalize on the morphological and syntactic irregularities that MWEs exhibit, using computational resources that are more readily-available.

Ramisch et al. (2008) evaluate a number of association measures on the task of identifying English Verb-Particle Constructions and German Adjective-Noun pairs. They show that adding linguistic information (mostly POS and POS-sequence patterns) to the association measure yields a significant improvement in performance over using pure frequency. We follow this line of research by defining a number of syntactic patterns as a source of linguistic information. In addition, our linguistic features are much more specific to the phenomenon we are interested in, and the syntactic patterns are enriched by morphological information pertaining to the idiosyncrasy of MWEs; we believe that this explains the improved performance compared to the baseline.

Several works address the *lexical fixedness* or *syntactic fixedness* of (certain types of) MWEs in order to extract them from texts. An expression is considered lexically fixed if replacing any of its constituents by a semantically (and syntactically) similar word generally results in an invalid or literal expression. Syntactically fixed expressions prohibit (or restrict) syntactic variation.

For example, Van de Cruys and Villada Moirón (2007) use lexical fixedness to extract Dutch Verb-Noun idiomatic combinations (VNICs). Bannard (2007) uses syntactic fixedness to identify English VNICs. Another work uses both the syntactic and the lexical fixedness of VNICs in order to distinguish them from non-idiomatic ones, and eventually to extract them from corpora (Fazly and Stevenson, 2006). While these approaches are in line with ours, they require lexical semantic resources (e.g., a database that determines semantic similarity among words) and syntactic resources (parsers) that are unavailable for Hebrew (and many other languages). Our approach only requires morphological processing, which is more readily-available for several languages.

Another unique feature of our work is that it computationally addresses Hebrew (and, more generally, Semitic) MWEs for the first time. Berman and Ravid (1986) define the *dictionary degree* of noun compounds in Hebrew as their closeness to a single word from a grammatical point of view, as judged by the manner in which they are grasped by language speakers. A group of 120 Hebrew speakers were asked to assign a dictionary degree (from 1 to 5) to a list of 30 noun compounds. An analysis of the questionnaire results revealed that language speaker share a common dictionary, where the highest degree of agreement was achieved on the ends of the dictionary degree spectrum. Another conclusion is that both the pragmatic uses of the noun compound and the semantic relation between its constituents define the dictionary degree of the compound. Not having access to semantic and pragmatic knowledge, we are trying to approximate it using morphology.

Attia (2005) proposes methods to process fixed, semi-fixed, and syntactically-flexible *Arabic* MWEs (adopting the classification and the terminology of Sag et al. (2002)). Fabri (2009) provides an overview of the different types of compounds (14 in total) in present-day Maltese, focusing on one type of compounds consisting of an adjective followed by a noun. He also provides morphological, syntactic, and semantic properties of this group which distinguishes them from other non-compound constructions. Automatic identification of MWEs is not addressed in either of these works.

5 Conclusions and future work

We described a system that can identify Hebrew noun compounds with high accuracy, distinguishing them from non-idiomatic noun-noun constructions. The methodology we advocate is based on careful examination of the linguistic peculiarities of the construction, followed by corpus-based approximation of these properties via a general machine learning algorithm that is fed with features based on the linguistic properties. While our application is limited to a particular construction in a particular language, we are confident that it can be equally well applied to other constructions and other languages, as long as the targeted MWEs exhibit a consistent set of irregular features (especially in the morphology).

This work can be extended in various directions. Addressing other constructions is relatively easy, and requires only a theoretical linguistic investigation of the construction. We are currently interested in extending the system to cope also with Adjective-Noun, Noun-Adjective and Verb-Preposition constructions in Hebrew.

The accuracy of MWE acquisition systems can be further improved by combining our morphological and syntactic features with semantically informed features such as translational entropy computed from a parallel corpus (Villada Moirón and Tiedemann, 2006), or features that can capture the local linguistic context of the expression using latent semantic analysis (Katz and Giesbrecht, 2006). We are currently working on the former direction (Tsvetkov and Wintner, 2010b), utilizing a small Hebrew-English parallel corpus (Tsvetkov and Wintner, 2010a).

Finally, we are interested in evaluating the methodology proposed in this paper to other languages with complex morphology, in particular to Arabic. We leave this direction to future research.

Acknowledgments

This research was supported by THE ISRAEL SCIENCE FOUNDATION (grants No. 137/06, 1269/07). We are grateful to Alon Itai for his continuous help and advice throughout the course of this project, and to Bracha Nir for very useful comments. We also wish to thank Yulia Tsvetkov and Gily Chen for their annotation work.

References

Alegria, Iñaki, Olatz Ansa, Xabier Artola, Nerea Ezeiza, Koldo Gojenola, and Ruben Urizar. 2004. Representation and treatment of multiword expressions in Basque. In Tanaka, Takaaki, Aline Villavicencio, Francis Bond, and Anna Korhonen, editors, *Second ACL Workshop on Multiword Expressions: Integrating Processing*, pages 48–55, Barcelona, Spain, July. Association for Computational Linguistics.

Attia, Mohammed A. 2005. Accommodating multiword expressions in an lfg grammar. The ParGram Meeting, Japan September 2005, September. Mohammed A. Attia The University of Manchester School of Informatics mohammed.attia@postgrad.manchester.ac.uk.

Bannard, Colin. 2007. A measure of syntactic flexibility for automatically identifying multiword expressions in corpora. In *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions*, pages 1–8. Association for Computational Linguistics.

Bar-haim, Roy, Khalil Sima'an, and Yoad Winter. 2008. Part-of-speech tagging of Modern Hebrew text. *Natural Language Engineering*, 14(2):223–251.

Berman, Ruth A. and Dorit Ravid. 1986. Lexicalization of noun compounds. *Hebrew Linguistics*, 24:5–22. In Hebrew.

Borer, Hagit. 1988. On the morphological parallelism between compounds and constructs. In Booij, Geert and Jaap van Marle, editors, *Yearbook of Morphology 1*, pages 45–65. Foris publications, Dordrecht, Holland.

Borer, Hagit. 1996. The construct in review. In Lecarme, Jacqueline, Jean Lowenstamm, and Ur Shlonsky, editors, *Studies in Afroasiatic Grammar*, pages 30–61. Holland Academic Graphics, The Hague.

Chang, Chih-Chung and Chih-Jen Lin, 2001. *LIBSVM: a library for support vector machines.* Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Church, Kenneth. W. and Patrick Hanks. 1989. Word association norms, mutual information and lexicography (rev). *Computational Linguistics*, 19(1):22–29.

Evert, Stefan and Brigitte Krenn. 2001. Methods for the qualitative evaluation of lexical association measures. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 188–195, Morristown, NJ, USA. Association for Computational Linguistics.

Fabri, Ray. 2009. Compounding and adjective-noun compounds in Maltese. In Comrie, Bernard, Ray Fabri, Elizabeth Hume, Manwel Mifsud, Thomas Stolz, and Martine Vanhove, editors, *Introducing Maltese Linguistics*, volume 113 of *Studies in Language Companion Series*. John Benjamins.

Fazly, Afsaneh and Suzanne Stevenson. 2006. Automatically constructing a lexicon of verb phrase idiomatic combinations. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 337–344.

Glinert, Lewis. 1989. *The Grammar of Modern Hebrew*. Cambridge University Press, Cambridge.

Itai, Alon and Shuly Wintner. 2008. Language resources for Hebrew. *Language Resources and Evaluation*, 42:75–98, March.

Katz, Graham and Eugenie Giesbrecht. 2006. Automatic identification of non-compositional multiword expressions using latent semantic analysis. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 12–19, Sydney, Australia, July. Association for Computational Linguistics.

Levi, Judith N. 1976. A semantic analysis of Hebrew compound nominals. In Cole, Peter, editor, *Studies in Modern Hebrew Syntax and Semantics*, number 32 in North-Holland Linguistic Series, pages 9–55. North-Holland, Amsterdam.

Manning, Christopher D. and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. The MIT Press, Cambridge, Mass.

Oflazer, Kemal, Özlem Çetinoğlu, and Bilge Say. 2004. Integrating morphology with multi-word expression processing in Turkish. In Tanaka, Takaaki, Aline Villavicencio, Francis Bond, and Anna Korhonen, editors, *Second ACL Workshop on Multiword Expressions: Integrating Processing*, pages 64–71, Barcelona, Spain, July. Association for Computational Linguistics.

Ordan, Noam and Shuly Wintner. 2007. Hebrew WordNet: a test case of aligning lexical databases across languages. *International Journal of Translation, special issue on Lexical Resources for Machine Translation*, 19(1).

Pecina, Pavel. 2008. A machine learning approach to multiword expression extraction. In *Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions*.

Ramisch, Carlos, Paulo Schreiner, Marco Idiart, and Alline Villavicencio. 2008. An evaluation of methods for the extraction of multiword expressions. In *Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions*.

Sag, Ivan, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics (CICLING 2002)*, pages 1–15, Mexico City, Mexico.

Tsvetkov, Yulia and Shuly Wintner. 2010a. Automatic acquisition of parallel corpora from websites with dynamic content. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, pages 3389–3392. European Language Resources Association (ELRA), May.

Tsvetkov, Yulia and Shuly Wintner. 2010b. Extraction of multi-word expressions from small parallel corpora. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, August.

Van de Cruys, Tim and Begoña Villada Moirón. 2007. Semantics-based multiword expression extraction. In *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions*, pages 25–32, Prague, Czech Republic, June. Association for Computational Linguistics.

Villada Moirón, Begoña and Jörg Tiedemann. 2006. Identifying idiomatic expressions using automatic word alignment. In *Proceedings of the EACL 2006 Workshop on Multi-word-expressions in a multilingual context*. Association for Computational Linguistics.

Villavicencio, Aline, Francis Bond, Anna Korhonen, and Diana McCarthy. 2005. Introduction to the special issue on multiword expressions: Having a crack at a hard nut. *Computer Speech & Language*, 19(4):365–377.

Wintner, Shuly. 2000. Definiteness in the Hebrew noun phrase. *Journal of Linguistics*, 36:319–363.

Yona, Shlomo and Shuly Wintner. 2008. A finite-state morphological grammar of Hebrew. *Natural Language Engineering*, 14(2):173–190, April.

Robust Measurement and Comparison of Context Similarity for Finding Translation Pairs

Daniel Andrade[†], **Tetsuya Nasukawa**[‡], **Jun'ichi Tsujii**[†]
[†]Department of Computer Science, University of Tokyo
{daniel.andrade, tsujii}@is.s.u-tokyo.ac.jp
[‡]IBM Research - Tokyo
nasukawa@jp.ibm.com

Abstract

In cross-language information retrieval it is often important to align words that are similar in meaning in two corpora written in different languages. Previous research shows that using context similarity to align words is helpful when no dictionary entry is available. We suggest a new method which selects a subset of words (pivot words) associated with a query and then matches these words across languages. To detect word associations, we demonstrate that a new Bayesian method for estimating Point-wise Mutual Information provides improved accuracy. In the second step, matching is done in a novel way that calculates the chance of an accidental overlap of pivot words using the hypergeometric distribution. We implemented a wide variety of previously suggested methods. Testing in two conditions, a small comparable corpora pair and a large but unrelated corpora pair, both written in disparate languages, we show that our approach consistently outperforms the other systems.

1 Introduction

Translating domain-specific, technical terms from one language to another can be challenging because they are often not listed in a general dictionary. The problem is exemplified in cross-lingual information retrieval (Chiao and Zweigenbaum, 2002) restricted to a certain domain. In this case, the user might enter only a few technical terms. However, jargons that appear frequently in the data set but not in general dictionaries, impair the usefulness of such systems. Therefore, various means to extract translation pairs automatically have been proposed. They use different clues, mainly

- Spelling distance or transliterations, which are useful to identify loan words (Koehn and Knight, 2002).
- Context similarity, helpful since two words with identical meaning are often used in similar contexts across languages (Rapp, 1999).

The first type of information is quite specific; it can only be helpful in a few cases, and can thereby engender high-precision systems with low recall, as described for example in (Koehn and Knight, 2002). The latter is more general. It holds for most words including loan words. Usually the context of a word is defined by the words which occur around it (bag-of-words model).

Let us briefly recall the main idea for using context similarity to find translation pairs. First, the degree of association between the query word and all content words is measured with respect to the corpus at hand. The same is done for every possible translation candidate in the target corpus. This way, we can create a feature vector for the query and all its possible translation candidates. We can assume that, for some content words, we have valid translations in a general dictionary, which enables us to compare the vectors across languages. We will designate these content words as pivot words. The query and its translation candidates are then compared using their feature vectors, where each dimension in the feature vector contains the degree of association to

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 19–27,
Beijing, August 2010

one pivot word. We define the degree of association, as a measurement for finding words that co-occur, or which do not co-occur, more often than we would expect by pure chance.[1]

We argue that common ways for comparing similarity vectors across different corpora perform worse because they assume that degree of associations are very similar across languages and can be compared without much preprocessing. We therefore suggest a new robust method including two steps. Given a query word, in the first step we determine the set of pivots that are all positively associated with statistical significance. In the second step, we compare this set of pivots with the set of pivots extracted for a possible translation candidate. For extracting positively associated pivots, we suggest using a new Bayesian method for estimating the critical Pointwise Mutual Information (PMI) value. In the second step, we use a novel measure to compare the sets of extracted pivot words which is based on an estimation of the probability that pivot words overlap by pure chance. Our approach engenders statistically significant improved accuracy for aligning translation pairs, when compared to a variety of previously suggested methods. We confirmed our findings using two very different pairs of comparable corpora for Japanese and English.

In the next section, we review previous related work. In Section 3 we explain our method in detail, and argue that it overcomes subtle weaknesses of several previous efforts. In Section 4, we show with a series of cross-lingual experiments that our method, in some settings, can lead to considerable improvement in accuracy. Subsequently in Section 4.2, we analyze our method in contrast to the baseline by giving two examples. We summarize our findings in Section 5.

2 Related Work

Extracting context similarity for nouns and then matching them across languages to find translation pairs was pioneered in (Rapp, 1999) and (Fung, 1998). The work in (Chiao and Zweigenbaum, 2002), which can be regarded as a variation of (Fung, 1998), uses tf.idf, but suggests to normalize the term frequency by the maximum number of co-occurrences of two words in the corpus. All this work is closely related to our work because they solely consider context similarity, whereas context is defined using a word window. The work in (Rapp, 1999; Fung, 1998; Chiao and Zweigenbaum, 2002) will form the baselines for our experiments in Section 4.[2] This baseline is also similar to the baseline in (Gaussier et al., 2004), which showed that it can be difficult to beat such a feature vector approach.

In principle our method is not restricted to how context is defined; we could also use, for example, modifiers and head words, as in (Garera et al., 2009). Although, we found in a preliminary experiment that using a dependency parser to differentiate between modifiers and head words like in (Garera et al., 2009), instead of a bag-of-words model, in our setting, actually decreased accuracy due to the narrow dependency window. However, our method could be combined with a back-translation step, which is expected to improve translation quality as in (Haghighi et al., 2008), which performs indirectly a back-translation by matching *all* nouns mutually exclusive across corpora. Notably, there also exist promising approaches which use both types of information, spelling distance, and context similarity in a joint framework, see (Haghighi et al., 2008), or (Déjean et al., 2002) which include knowledge of a thesaurus. In our work here, we concentrate on the use of degrees of association as an effective means to extract word translations.

In this application, to measure association robustly, often the Log-Likelihood Ratio (LLR) measurement is suggested (Rapp, 1999; Morin et al., 2007; Chiao and Zweigenbaum, 2002). The occurrence of a word in a document is modeled as a binary random variable. The LLR measurement measures stochastic dependency between

[1] For example ”car” and ”tire” are expected to have a high (positive) degree of association, and ”car” and ”apple” is expected to have a high (negative) degree of association.

[2] Notable differences are that we neglected word order, in contrast to (Rapp, 1999), as it is little useful to compare it between Japanese and English. Furthermore in contrast to (Fung, 1998) we use only one translation in the dictionary, which we select by comparing the relative frequencies. We also made a second run of the experiments where we manually selected the correct translations for the first half of the most frequent pivots – Results did not change significantly.

two such random variables (Dunning, 1993), and is known to be equal to Mutual Information that is linearly scaled by the size of the corpus (Moore, 2004). This means it is a measure for how much the occurrence of word A makes the occurrence of word B more likely, which we term positive association, and how much the absence of word A makes the occurrence of word B more likely, which we term negative association. However, our experiments show that only positive association is beneficial for aligning words cross-lingually. In fact, LLR can still be used for extracting positive associations by filtering in a pre-processing step words with possibly negative associations (Moore, 2005). Nevertheless a problem which cannot be easily remedied is that confidence estimates using LLR are unreliable for small sample sizes (Moore, 2004). We suggest a more principled approach that measures from the start only how much the occurrence of word A makes the occurrence of word B more likely, which is designated as Robust PMI.

Another point that is common to (Rapp, 1999; Morin et al., 2007; Chiao and Zweigenbaum, 2002; Garera et al., 2009; Gaussier et al., 2004) is that word association is compared in a fine-grained way, i.e. they compare the degree of association[3] with every pivot word, even when it is low or exceptionally high. They suggest as a comparison measurement Jaccard similarity, Cosine similarity, and the L1 (Manhattan) distance.

3 Our Approach

We presume that rather than similarity between *degree (strength of)* of associations, the *existence* of common word associations is a more reliable measure for word similarity because the degrees of association are difficult to compare for the following reasons:

- **Small differences in the degree of association are not statistically significant**
  Taking, for example, two sample sets from the same corpus, we will in general measure different degrees of association.

- **Differences in sub-domains / sub-topics**
  Corpora sharing the same topic can still differ in sub-topics.

- **Differences in style or language**
  Differences in word usage. [4]

Other information that is used in vector approaches such as that in (Rapp, 1999) is negative association, although negative association is less informative than positive. Therefore, if it is used at all, it should be assigned a much smaller weight.

Our approach caters to these points, by first deciding whether a pivot word is positively associated (with statistical significance) or whether it is not, and then uses solely this information for finding translation pairs in comparable corpora. It is divisible into two steps. In the first, we use a Bayesian estimated Pointwise Mutual Information (PMI) measurement to find the pivots that are positively associated with a certain word with high confidence. In the second step, we compare two words using their associated pivots as features. The similarity of feature sets is calculated using pointwise entropy. The words for which feature sets have high similarity are assumed to be related in meaning.

3.1 Extracting positively associated words – Feature Sets

To measure the degree of positive association between two words x and y, we suggest the use of information about how much the occurrence of word x makes the occurrence of word y more likely. We express this using Pointwise Mutual Information (PMI), which is defined as follows:

$$PMI(x,y) = \log \frac{p(x,y)}{p(x) \cdot p(y)} = \log \frac{p(x|y)}{p(x)} \,.$$

Therein, $p(x)$ is the probability that word x occurs in a document; $p(y)$ is defined analogously. Furthermore, $p(x,y)$ is the probability that both

[3] To clarify terminology, where possible, we will try to distinguish between *association* and *degree of association*. For example word “car” has the *association* “tire”, whereas the *degree of association* with “tire” is a continuous number, like 5.6.

[4] For example, “stop” is not the only word to describe the fact that a car halted.

words occur in the same document. A positive association is given if $p(x|y) > p(x)$. In related works that use the PMI (Morin et al., 2007), these probabilities are simply estimated using relative frequencies, as

$$PMI(x, y) = \log \frac{\frac{f(x,y)}{n}}{\frac{f(x)}{n}\frac{f(y)}{n}},$$

where $f(x)$, $f(y)$ is the document frequency of word x and word y, and $f(x, y)$ is the co-occurrence frequency; n is the number of documents. However, using relative frequencies to estimate these probabilities can, for low-frequency words, produce unreliable estimates for PMI (Manning and Schütze, 2002). It is therefore necessary to determine the uncertainty of PMI estimates. The idea of defining confidence intervals over PMI values is not new (Johnson, 2001); however, the problem is that exact calculation is very computationally expensive if the number of documents is large, in which case one can approximate the binomial approximation for example with a Gaussian, which is, however only justified if n is large *and* p, the probability of an occurrence, is not close to zero (Wilcox, 2009). We suggest to define a beta distribution over each probability of the binary events that word x occurs, i.e. $[x]$, and analogously $[x|y]$. It was shown in (Ross, 2003) that a Bayesian estimate for Bernoulli trials using the beta distribution delivers good credibility intervals[5], importantly, when sample sizes are small, or when occurrence probabilities are close to 0. Therefore, we assume that

$$p(x|y) \sim beta(\alpha'_{x|y}, \beta'_{x|y}), p(x) \sim beta(\alpha'_x, \beta'_x)$$

where the parameters for the two beta distributions are set to

$$\alpha'_{x|y} = f(x, y) + \alpha_{x|y}\,,$$
$$\beta'_{x|y} = f(y) - f(x, y) + \beta_{x|y}\,, \text{and}$$
$$\alpha'_x = f(x) + \alpha_x,\ \beta'_x = n - f(x) + \beta_x\,.$$

Prior information related to $p(x)$ and the conditional probability $p(x|y)$ can be incorporated by setting the hyper-parameters of the beta-distribtutions.[6] These can, for example, be learned from another unrelated corpora pair and then weighted appropriately by setting $\alpha + \beta$. For our experiments, we use no information beyond the given corpora pair; the conditional priors are therefore set equal to the prior for $p(x)$. Even if we do not know which word x is, we have a notion about $p(x)$ because Zipf's law indicates to us that we should expect it to be small. A crude estimation is therefore the mean word occurrence probability in our corpus as

$$\gamma = \frac{1}{|\text{all words}|} \sum_{x \in \{\text{all words}\}} \frac{f(x)}{n}\,.$$

We give this estimate a total weight of one observation. That is, we set

$$\alpha = \gamma\,, \beta = 1 - \gamma\,.$$

From a practical perspective, this can be interpreted as a smoothing when sample sizes are small, which is often the case for $p(x|y)$. Because we assume that $p(x|y)$ and $p(x)$ are random variables, PMI is consequently also a random variable that is distributed according to a beta distribution ratio.[7] For our experiments, we apply a general sampling strategy. We sample $p(x|y)$ and $p(x)$ independently and then calculate the ratio of times $PMI > 0$ to determine $P(PMI > 0)$.[8] We will refer to this method as Robust PMI (RPMI).

Finally we can calculate, for any word x, the set of pivot words which have most likely a positive association with word x. We require that this set be statistically significant: the probability of one or more words being not a positive association is smaller than a certain p-value.[9]

[5] In the Bayesian notation we refer here to credibility intervals instead of confidence intervals.

[6] The hyper-parameters α and β, can be intuitively interpreted in terms of document frequency. For example α_x is the number of times we belief the word x occurs, and β_x the number of times we belief that x does not occur in a corpus. Analogously $\alpha_{x|y}$ and $\beta_{x|y}$ can be interpreted with respect to the subset of the corpus where the word y occurs, instead of the whole corpus. Note however, that α and β do not necessarily have to be integers.

[7] The resulting distribution for the general case of a beta distribution ratio was derived in (Pham-Gia, 2000). Unfortunately, it involves the calculation of a Gauss hyper-geometric function that is computationally expensive for large n.

[8] For experiments, we used $100,000$ samples for each estimate of $P(PMI > 0)$.

[9] We set, for all of our experiments, the p-value to 0.01.

As an alternative for determining the probability of a positive association using $P(PMI > 0)$, we calculate LLR and assume that approximately $LLR \sim \chi^2$ with one degree of freedom (Dunning, 1993). Furthermore, to ensure that only positive association counts, we set the probability to zero if $p(x, y) < p(x) \cdot p(y)$, where the probabilities are estimated using relative frequencies (Moore, 2005). We refer to this as LLR(P); lacking this correction, it is LLR.

3.2 Comparing Word Feature Sets Across Corpora

So far, we have explained a robust means to extract the pivot words that have a positive association with the query. The next task is to find a sensible way to use these pivots to compare the query with candidates from the target corpus. A simple means to match a candidate with a query is to see how many pivots they have in common, i.e. using the matching coefficient (Manning and Schütze, 2002) to score candidates. This similarity measure produces a reasonable result, as we will show in the experiment section; however, in our error analysis, we found out that this gives a bias to candidates with higher frequencies, which is explainable as follows. Assuming that a word A has a fixed number of pivots that are positively associated, then depending on the sample size—the document frequency in the corpus—not all of these are statistically significant. Therefore, not all true positive associations are included in the feature set to avoid possible noise. If the document frequency increases, then we can extract more statistically significant positive associations and the cardinality of the feature set increases. This consequently increases the likelihood of having more pivots that overlap with pivots from the query's feature set. For example, imagine two candidate words A and B, for which feature sets of both include the feature set of the query, i.e. a complete match, however A's feature set is much larger than B's feature set. In this case, the information conveyed by having a complete match with the query word‘s feature set is lower in the case of A's feature set than in case of B's feature set. Therefore, we suggest its use as a basis of our similarity measure, the degree of pointwise entropy of having an estimate of m matches, as

$$\text{Information}(m, q, c) = -\log(P(matches = m)).$$

Therein, $P(matches = m)$ is the likelihood that a candidate word with c pivots has m matches with the query word, which has q pivots. Letting w be the total number of pivot words, we can then calculate that the probability that the candidate with c pivots was selected by chance

$$P(matches = m) = \frac{\binom{q}{m} \cdot \binom{w-q}{c-m}}{\binom{w}{c}}.$$

Note that this probability equals a hypergeometric distribution.[10] The smaller $P(matches = m)$ is, the less likely it is that we obtain m matches by pure chance. In other words, if $P(matches = m)$ is very small, m matches are more than we would expect to occur by pure chance.[11]

Alternatively, in our experiments, we also consider standard similarity measurements (Manning and Schütze, 2002) such as the Tanimoto coefficient, which also lowers the score of candidates that have larger feature sets.

4 Experiments

In our experiments, we specifically examine translating nouns, mostly technical terms, which occur in complaints about cars collected by the Japanese Ministry of Land, Infrastructure, Transport and Tourism (MLIT)[12], and in complaints about cars collected by the USA National Highway Traffic Safety Administration (NHTSA)[13]. We create for each data collection a corpus for which a document corresponds to one car customer reporting a certain problem in free text. The complaints are, in general, only a few sentences long.

[10] $\binom{q}{m}$ is the number of possible combinations of pivots which the candidate has in common with the query. Therefore, $\binom{q}{m} \cdot \binom{w-q}{c-m}$ is the number of possible different feature sets that the candidate can have such that it shares m common pivots with the query. Furthermore, $\binom{w}{c}$ is the total number of possible feature sets the candidate can have.

[11] The discussion is simplified here. It can also be that $P(matches = m)$ is very small, if there are less occurrences of m that we would expect to occur by pure chance. However, this case can be easily identified by looking at the gradient of $P(matches = m)$.

[12] http://www.mlit.go.jp/jidosha/carinf/rcl/defects.html

[13] http://www-odi.nhtsa.dot.gov/downloads/index.cfm

To verify whether our results can be generalized over other pairs of comparable corpora, we additionally made experiments using two corpora extracted from articles of Mainichi Shinbun, a Japanese newspaper, in 1995 and English articles from Reuters in 1997. There are two notable differences between those two pairs of corpora: the content is much less comparable, Mainichi reports more national news than world news, and secondly, Mainichi and Reuters corpora are much larger than MLIT/NHTSA.[14]

For both corpora pairs, we extracted a gold-standard semi-automatically by looking at Japanese nouns and their translations with document frequency of at least 50 for MLIT/NHTSA, and 100 for Mainichi/Reuters. As a dictionary we used the Japanese-English dictionary JMDic[15]. In general, we preferred domain-specific terms over very general terms, i.e. for example for MLIT/NHTSA the noun 噴射 "injection" was preferred over 取り付け "installation". We extracted 100 noun pairs for MLIT/NHTSA and Mainichi/Reuters, each. Each Japanese noun which is listed in the gold-standard forms a query which is input into our system. The resulting ranking of the translation candidates is automatically evaluated using the gold-standard. Therefore, synonyms that are not listed in the gold standard are not recognized, engendering a conservative estimation of the translation accuracy. Because all methods return a ranked list of translation candidates, the accuracy is measured using the rank of the translation listed in the gold-standard.[16] The Japanese corpora are preprocessed with MeCab (Kudo et al., 2004); the English corpora with Stepp Tagger (Tsuruoka et al., 2005) and Lemmatizer (Okazaki et al., 2008). As a dictionary we use the Japanese-English dictionary JMDic[17]. In line with related work (Gaussier et al., 2004), we remove a word pair (Japanese noun s, English noun t) from the dictionary, if s occurs in the gold-standard. Afterwards we define the pivot words by consulting the remaining dictionary.

4.1 Crosslingual Experiment

We compare our approach used for extracting cross-lingual translation pairs against several baselines. We compare to LLR + Manhattan (Rapp, 1999) and our variation LLR(P) + Manhattan. Additionally, we compare TFIDF(MSO) + Cosine, which is the TFIDF measure, whereas the Term Frequency is normalized using the maximal word frequency and the cosine similarity for comparison suggested in (Fung, 1998). Furthermore, we implemented two variations of this, TFIDF(MPO) + Cosine and TFIDF(MPO) + Jaccard coefficient, which were suggested in (Chiao and Zweigenbaum, 2002). In fact, TFIDF(MPO) is the TFIDF measure, whereas the Term Frequency is normalized using the maximal word pair frequency. The results are displayed in Figure 1. Our approach clearly outperforms all baselines; notably it has Top 1 accuracy of 0.14 and Top 20 accuracy of 0.55, which is much better than that for the best baseline, which is 0.11 and 0.44, respectively.

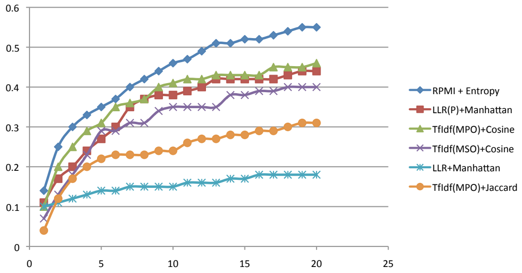


Figure 1: Crosslingual Experiment MLIT/NHTSA – Percentile Ranking of RPMI + Entropy Against Various Previous Suggested Methods.

We next leave the proposed framework constant, but change the mode of estimating positive associations and the way to match feature sets. As alternatives for estimating the probability that there is a positive association, we test LLR(P) and LLR. As alternatives for comparing feature sets, we investigate the matching coefficient (match), cosine similarity (cosine), Tanimoto coefficient (tani), and overlap coefficient

[14] MLIT/MLIT has each 20,000 documents. Mainichi/Reuters corpora 75,935 and 148,043 documents, respectively.

[15] http://www.csse.monash.edu.au/ jwb/edict_doc.html

[16] In cases for which there are several translations listed for one word, the rank of the first is used.

[17] http://www.csse.monash.edu.au/ jwb/edict_doc.html

(over) (Manning and Schütze, 2002). The result of every combination is displayed concisely in Table 1 using the median rank[18]. The cases in which the median ranks are close to RPMI + Entropy are magnified in Table 2. We can see there that RPMI + Entropy, and LLR(P) + Entropy perform nearly equally. All other combinations perform worse, especially in Top 1 accuracy. Finally, LLR(P) presents a clear edge over LLR, which suggests that indeed only positive associations seem to matter in a cross-lingual setting.

	Entropy	Match	Cosine	Tani	Over
RPMI	13.0	17.0	24.0	37.5	36.0
LLR(P)	16.0	15.0	22.5	34.0	25.5
LLR	23.5	22.0	27.5	50.5	50.0

Table 1: Crosslingual experiment MLIT/NHTSA – Evaluation matrix showing the median ranks of several combinations of association and similarity measures.

	Top 1	Top 10	Top 20
RPMI + Entropy	0.14	0.46	0.55
RPMI + Matching	0.08	0.41	0.57
LLR(P) + Entropy	0.14	0.46	0.55
LLR(P) + Matching	0.08	0.44	0.55

Table 2: Accuracies for crosslingual experiment MLIT/NHTSA.

Finally we conduct an another experiment using the corpora pair Mainichi/Reuters which is quite different from MLIT/NHTSA. When comparing to the best baselines in Table 3 we see that our approach again performs best. Furthermore, the experiments displayed in Table 4 suggest that Robust PMI and pointwise entropy are better choices for positive association measurement and similarity measurement, respectively. We can see that the overall best baseline turns out to be LLR(P) + Manhattan. Comparing the rank from each word from the gold-standard pairwise, we see that our approach, RPMI + Entropy, is significantly better than this baseline in MLIT/NHTSA as well as in Mainichi/Reuters.[19]

	Top 1	Top 10	Top 20
RPMI + Entropy	0.15	0.38	0.46
LLR(P) + Manhattan	0.10	0.26	0.33
TFIDF(MPO) + Cos	0.05	0.12	0.18

Table 3: Accuracies for crosslingual experiment Mainichi/Reuters – Comparison to best baselines.

	Top 1	Top 10	Top 20
RPMI + Entropy	0.15	0.38	0.46
RPMI + Matching	0.08	0.30	0.35
LLR(P) + Entropy	0.13	0.36	0.47
LLR(P) + Matching	0.08	0.29	0.37

Table 4: Accuracies for crosslingual experiment Mainichi/Reuters – Comparison to alternatives.

4.2 Analysis

In this section, we provide two representative examples extracted from the previous experiments which sheds light into a weakness of the standard feature vector approach which was used as a baseline before. The two example queries and the corresponding responses of LLR(P) + Manhattan and our approach are listed in Table 5. Furthermore in Table 6 we list the pivot words with the highest degree of association (here LLR values) for the query and its correct translation. We can see that a query and its translation shares some pivots which are associated with statistical significance[20]. However it also illustrates that the actual LLR value is less insightful and can hardly be compared across these two corpora.

Let us analyze the two examples in more detail. In Table 6, we see that the first query ギア "gear"[21] is highly associated with 入れる "shift". However, on the English side we see that gear is most highly associated with the pivot word gear. Note that here the word gear is also a pivot word corresponding to the Japanese pivot word 歯車 "gear (wheel)".[22] Since in English the word gear (shift) and gear (wheel) is polysemous, the surface forms are the same leading to a high LLR value of

[18] A median rank of i, means that 50% of the correct translations have a rank higher than i.

[19] Using pairwise test with p-value 0.05.

[20] Note that for example, an LLR value bigger than 11.0 means the chances that there is no association is smaller than 0.001 using that $LLR \sim \chi^2$.

[21] For a Japanese word, we write the English translation which is *appropriate in our context*, immediately after it.

[22] In other words, we have the entry (歯車, gear) in our dictionary but not the entry (ギア, gear). The first pair is used as a pivot, the latter word pair is what we try to find.

gear. Finally, the second example query ペダル "pedal" shows that words which, not necessarily always, but very often co-occur, can cause relatively high LLR values. The Japanese verb 踏む "to press" is associated with ペダル with a high LLR value – 4 times higher than 戻る "return" – which is not reflected on the English side. In summary, we can see that in both cases the degree of associations are rather different, and cannot be compared without preprocessing. However, it is also apparent that in both examples a simple L1 normalization of the degree of associations does *not* lead to more similarity, since the relative differences remain.

ギア "gear"		
Method	**Top 3 candidates**	**Rank**
baseline	jolt, lever, design	284
filtering	reverse, gear, lever	2

ペダル "pedal"		
Method	**Top 3 candidates**	**Rank**
baseline	mj, toyota, action	176
filtering	pedal, situation, occasion	1

Table 5: List of translation suggestions using LLR(P) + Manhattan (baseline) and our method (filtering). The third column shows the rank of the correct translation.

ギア		**gear**	
Pivots	**LLR(P)**	**Pivots**	**LLR(P)**
入る "shift"	154	gear	7064
入れる "shift"	144	shift	1270
抜ける "come out"	116	reverse	314

ペダル		**pedal**	
Pivots	**LLR(P)**	**Pivots**	**LLR(P)**
踏む "press"	628	floor	1150
戻る "return"	175	stop	573
足 "foot"	127	press	235

Table 6: Shows the three pivot words which have the highest degree of association with the query (left side) and the correct translation (right side).

5 Conclusions

We introduced a new method to compare context similarity across comparable corpora using a Bayesian estimate for PMI (Robust PMI) to extract positive associations and a similarity measurement based on the hypergeometric distribution (measuring pointwise entropy). Our experiments show that, for finding cross-lingual translations, the assumption that words with similar meaning share positive associations with the same words is more appropriate than the assumption that the degree of association is similar. Our approach increases Top 1 and Top 20 accuracy of up to 50% and 39% respectively, when compared to several previous methods. We also analyzed the two components of our method separately. In general, Robust PMI yields slightly better performance than the popular LLR, and, in contrast to LLR, allows to extract positive associations as well as to include prior information in a principled way. Pointwise entropy for comparing feature sets cross-lingually improved the translation accuracy clearly when compared with standard similarity measurements.

Acknowledgment

We thank Dr. Naoaki Okazaki and the anonymous reviewers for their helpful comments. Furthermore we thank Daisuke Takuma, IBM Research - Tokyo, for mentioning previous work on statistical corrections for PMI. This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan). The first author is supported by the MEXT Scholarship and by an IBM PhD Scholarship Award.

References

Chiao, Y.C. and P. Zweigenbaum. 2002. Looking for candidate translational equivalents in specialized, comparable corpora. In *Proceedings of the International Conference on Computational Linguistics*, pages 1–5. International Committee on Computational Linguistics.

Déjean, H., É. Gaussier, and F. Sadat. 2002. An approach based on multilingual thesauri and model combination for bilingual lexicon extraction. In *Proceedings of the International Conference on Computational Linguistics*, pages 1–7. International Committee on Computational Linguistics.

Dunning, T. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Fung, P. 1998. A statistical view on bilingual lexicon extraction: from parallel corpora to non-parallel corpora. *Lecture Notes in Computer Science*, 1529:1–17.

Garera, N., C. Callison-Burch, and D. Yarowsky. 2009. Improving translation lexicon induction from monolingual corpora via dependency contexts and part-of-speech equivalences. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 129–137. Association for Computational Linguistics.

Gaussier, E., J.M. Renders, I. Matveeva, C. Goutte, and H. Dejean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 526–533. Association for Computational Linguistics.

Haghighi, A., P. Liang, T. Berg-Kirkpatrick, and D. Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 771–779. Association for Computational Linguistics.

Johnson, M. 2001. Trading recall for precision with confidence-sets. Technical report, Brown University.

Koehn, P. and K. Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of ACL Workshop on Unsupervised Lexical Acquisition*, volume 34, pages 9–16. Association for Computational Linguistics.

Kudo, T., K. Yamamoto, and Y. Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 230–237. Association for Computational Linguistics.

Manning, C.D. and H. Schütze. 2002. *Foundations of Statistical Natural Language Processing*. MIT Press.

Moore, R.C. 2004. On log-likelihood-ratios and the significance of rare events. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 333–340. Association for Computational Linguistics.

Moore, R.C. 2005. A discriminative framework for bilingual word alignment. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 81–88. Association for Computational Linguistics.

Morin, E., B. Daille, K. Takeuchi, and K. Kageura. 2007. Bilingual terminology mining-using brain, not brawn comparable corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, volume 45, pages 664–671. Association for Computational Linguistics.

Okazaki, N., Y. Tsuruoka, S. Ananiadou, and J. Tsujii. 2008. A discriminative candidate generator for string transformations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 447–456. Association for Computational Linguistics.

Pham-Gia, T. 2000. Distributions of the ratios of independent beta variables and applications. *Communications in Statistics. Theory and Methods*, 29(12):2693–2715.

Rapp, R. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 519–526. Association for Computational Linguistics.

Ross, T.D. 2003. Accurate confidence intervals for binomial proportion and Poisson rate estimation. *Computers in Biology and Medicine*, 33(6):509–531.

Tsuruoka, Y., Y. Tateishi, J. Kim, T. Ohta, J. McNaught, S. Ananiadou, and J. Tsujii. 2005. Developing a robust part-of-speech tagger for biomedical text. *Lecture Notes in Computer Science*, 3746:382–392.

Wilcox, R.R. 2009. *Basic Statistics: Understanding Conventional Methods and Modern Insights*. Oxford University Press.

Multilingual Subjectivity: Are More Languages Better?

Carmen Banea, Rada Mihalcea
Department of Computer Science
University of North Texas
carmenbanea@my.unt.edu
rada@cs.unt.edu

Janyce Wiebe
Department of Computer Science
University of Pittsburgh
wiebe@cs.pitt.edu

Abstract

While subjectivity related research in other languages has increased, most of the work focuses on single languages. This paper explores the integration of features originating from multiple languages into a machine learning approach to subjectivity analysis, and aims to show that this enriched feature set provides for more effective modeling for the source as well as the target languages. We show not only that we are able to achieve over 75% macro accuracy in all of the six languages we experiment with, but also that by using features drawn from multiple languages we can construct high-precision meta-classifiers with a precision of over 83%.

1 Introduction

Following the terminology proposed by (Wiebe et al., 2005), subjectivity and sentiment analysis focuses on the automatic identification of private states, such as opinions, emotions, sentiments, evaluations, beliefs, and speculations in natural language. While subjectivity classification labels text as either subjective or objective, sentiment or polarity classification adds an additional level of granularity, by further classifying subjective text as either positive, negative or neutral.

To date, a large number of text processing applications have used techniques for automatic sentiment and subjectivity analysis, including automatic expressive text-to-speech synthesis (Alm et al., 1990), tracking sentiment timelines in on-line forums and news (Balog et al., 2006; Lloyd et al., 2005), and mining opinions from product reviews (Hu and Liu, 2004). In many natural language processing tasks, subjectivity and sentiment classification has been used as a first phase filtering to generate more viable data. Research that benefited from this additional layering ranges from question answering (Yu and Hatzivassiloglou, 2003), to conversation summarization (Carenini et al., 2008), and text semantic analysis (Wiebe and Mihalcea, 2006; Esuli and Sebastiani, 2006a).

Although subjectivity tends to be preserved across languages – see the manual study in (Mihalcea et al., 2007), (Banea et al., 2008) hypothesize that subjectivity is expressed differently in various languages due to lexicalization, formal versus informal markers, etc. Based on this observation, our research seeks to answer the following questions. First, can we *reliably* predict sentence-level subjectivity in languages other than English, by leveraging on a manually annotated English dataset? Second, can we improve the English subjectivity classification by expanding the feature space through the use of multilingual data? Similarly, can we also improve the classifiers in the other target languages? Finally, third, can we benefit from the multilingual subjectivity space and build a high-precision subjectivity classifier that could be used to generate subjectivity datasets in the target languages?

The paper is organized as follows. We introduce the datasets and the general framework in Section 2. Sections 3, 4, and 5 address in turn each of the three research questions mentioned above. Section 6 describes related literature in the area of multilingual subjectivity. Finally, we draw our conclusions in Section 7.

2 Multilingual Datasets

Corpora that are manually annotated for subjectivity, polarity, or emotion, are available in only select languages, since they require a considerable amount of human effort. Due to this impediment, the focus of this paper is to create a method for extrapolating subjectivity data devel-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 28–36,
Beijing, August 2010

SubjP	SubjR	SubjF	ObjP	ObjR	ObjF	AllP	AllR	AllF
90.4%	34.2%	46.6%	82.4%	30.7%	44.7%	86.7%	32.6%	47.4%

Table 1: Results obtained with a rule-based subjectivity classifier on the MPQA corpus (Wiebe and Riloff, 2005)

oped in a source language and to transfer it to other languages. Multilingual feature spaces are generated to create even better subjectivity classifiers, outperforming those trained on the individual languages alone.

We use the Multi-Perspective Question Answering (MPQA) corpus, consisting of 535 English-language news articles from a variety of sources, manually annotated for subjectivity (Wiebe et al., 2005). Although the corpus is annotated at the clause and phrase levels, we use the sentence-level annotations associated with the dataset in (Wiebe and Riloff, 2005). A sentence is labeled as subjective if it has at least one private state of strength medium or higher. Otherwise the sentence is labeled as objective. From the approximately 9700 sentences in this corpus, 55% of them are labeled as subjective, while the rest are objective. Therefore, 55% represents the majority baseline on this corpus. (Wiebe and Riloff, 2005) apply both a subjective and an objective rule-based classifier to the MPQA corpus data and obtain the results presented in Table 1.[1]

In order to generate parallel corpora to MPQA in other languages, we rely on the method we proposed in (Banea et al., 2008). We experiment with five languages other than English (*En*), namely Arabic (*Ar*), French (*Fr*), German (*De*), Romanian (*Ro*) and Spanish (*Es*). Our choice of languages is motivated by several reasons. First, we wanted languages that are highly lexicalized and have clear word delimitations. Second, we were interested to cover languages that are similar to English as well as languages with a completely different etymology. Consideration was given to include Asian languages, such as Chinese or Japanese, but the fact that their script without word-segmentation preprocessing does not directly map to words was a deterrent. Finally, another limitation on our choice of languages is the need for a publicly available machine translation system between the source language and each of the target languages.

We construct a subjectivity annotated corpus for each of the five languages by using machine translation to transfer the source language data into the target language. We then project the original sentence level English subjectivity labeling onto the target data. For all languages, other than Romanian, we use the Google Translate service,[2] a publicly available machine translation engine based on statistical models. The reason Romanian is not included in this group is that, at the time when we performed the first experiments, Google Translate did not provide a translation service for this language. Instead, we used an alternative statistical translation system called LanguageWeaver,[3] which was commercially available, and which the company kindly allowed us to use for research purposes.

The raw corpora in the five target languages are available for download at http://lit.csci.unt.edu/index.php/Downloads, while the English MPQA corpus can be obtained from http://www.cs.pitt.edu/mpqa.

Given the specifics of each language, we employ several preprocessing techniques. For Romanian, French, English, German and Spanish, we remove all the diacritics, numbers and punctuation marks except - and ’. The exceptions are motivated by the fact that they may mark contractions, such as En: *it’s* or Ro: *s-ar* (*may be*), and the component words may not be resolved to the correct forms. For Arabic, although it has a different encoding, we wanted to make sure to treat it in a way similar to the languages with a Roman

[1] For the purpose of this paper we follow this abbreviation style: *Subj* stands for subjective, *Obj* stands for objective, and *All* represents overall macro measures, computed over the subjective and objective classes; *P*, *R*, *F*, and *MAcc* correspond to precision, recall, F-measure, and macro-accuracy, respectively.

[2] http://www.google.com/translate_t

[3] http://www.languageweaver.com/

alphabet. We therefore use a library[4] that maps Arabic script to a space of Roman-alphabet letters supplemented with punctuation marks so that they can allow for additional dimensionality.

Once the corpora are preprocessed, each sentence is defined by six views: one in the original source language (English), and five obtained through automatic translation in each of the target languages. Multiple datasets that cover all possible combinations of six languages taken one through six (a total of 63 combinations) are generated. These datasets feature a vector for each sentence present in MPQA (approximately 9700). The vector contains only unigram features in one language for a monolingual dataset. For a multilingual dataset, the vector represents a cumulation of monolingual unigram features extracted from each view of the sentence. For example, one of the combinations of six taken three is Arabic-German-English. For this combination, the vector is composed of unigram features extracted from each of the Arabic, German and English translations of the sentence.

We perform ten-fold cross validation and train Naïve Bayes classifiers with feature selection on each dataset combination. The top 20% of the features present in the training data are retained. For datasets resulting from combinations of all languages taken one, the classifiers are monolingual classifiers. All other classifiers are multilingual, and their feature space increases with each additional language added. Expanding the feature set by encompassing a group of languages enables us to provide an answer to two problems that can appear due to data sparseness. First, enough training data may not be available in the monolingual corpus alone in order to correctly infer labeling based on statistical measures. Second, features appearing in the monolingual test set may not be present in the training set and therefore their information cannot be used to generate a correct classification.

Both of these problems are further explained through the examples below, where we make the simplifying assumption that the words in italics are the only potential carriers of subjective content, and that, without them, their surrounding contexts would be objective. Therefore, their association with an either objective or subjective meaning imparts to the entire segment the same labeling upon classification.

To explore the first sparseness problem, let us consider the following two examples extracted from the English version of the MPQA dataset, followed by their machine translations in German:

> "En 1: rights group Amnesty International said it was *concerned* about the high risk of violence in the aftermath"
> "En 2: official said that US diplomats to countries *concerned* are authorized to explain to these countries"
> "De 1: Amnesty International sagte, es sei *besorgt* über das hohe Risiko von Gewalt in der Folgezeit"
> "De 2: Beamte sagte, dass US-Diplomaten *betroffenen* Länder berechtigt sind, diese Länder zu erklären"

We focus our discussion on the word *concerned*, which in the first example is used in its subjective sense, while in the second it carries an objective meaning (as it refers to a group of countries exhibiting a particular feature defined earlier on in the context). The words in italics in the German contexts represent the translations of *concerned* into German, which are functionally different as they are shaped by their surrounding context. By training a classifier on the English examples alone, under the data sparseness paradigm, the machine learning model may not differentiate between the word's objective and subjective uses when predicting a label for the entire sentence. However, appending the German translation to the examples generates additional dimensions for this model and allows the classifier to potentially distinguish between the senses and provide the correct sentence label.

For the second problem, let us consider two other examples from the English MPQA and their respective translations into Romanian:

> "En 3: could secure concessions on Taiwan in return for *supporting* Bush on issues such as anti-terrorism and"

[4] Lingua::AR::Word PERL library.

Lang	SubjP	SubjR	SubjF	ObjP	ObjR	ObjF	AllP	AllR	AllF	MAcc
En	74.01%	**83.64%**	**78.53%**	**75.89%**	63.68%	**69.25%**	**74.95%**	**73.66%**	**73.89%**	**74.72%**
Ro	73.50%	82.06%	77.54%	74.08%	63.40%	68.33%	73.79%	72.73%	72.94%	73.72%
Es	**74.02%**	82.84%	78.19%	75.11%	**64.05%**	69.14%	74.57%	73.44%	73.66%	74.44%
Fr	73.83%	83.03%	78.16%	75.19%	63.61%	68.92%	74.51%	73.32%	73.54%	74.35%
De	73.26%	83.49%	78.04%	75.32%	62.30%	68.19%	74.29%	72.90%	73.12%	74.02%
Ar	71.98%	81.47%	76.43%	72.62%	60.78%	66.17%	72.30%	71.13%	71.30%	72.22%

Table 2: Naïve Bayes learners trained on six individual languages

> "En 4: to the potential for change from within America. *Supporting* our schools and community centres is a good"
>
> "Ro 3: ar putea asigura concesii cu privire la Taiwan, în schimb pentru *susţinerea* lui Bush pe probleme cum ar fi anti-terorismului şi"
>
> "Ro 4: la potenţialul de schimbare din interiorul Americii. *Sprijinirea* şcolile noastre şi centre de comunitate este un bun"

In this case, *supporting* is used in both English examples in senses that are both subjective; the word is, however, translated into Romanian through two synonyms, namely *susţinerea* and *sprijinirea*. Let us assume that sufficient training examples are available to strengthen a link between *supporting* and *susţinerea*, and the classifier is presented with a context containing *sprijinirea*, unseen in the training data. A multilingual classifier may be able to predict a label for the context using the co-occurrence metrics based on *supporting* and extrapolate a label when the context contains both the English word and its translation into Romanian as *sprijinirea*. For a monolingual classifier, such an inference is not possible, and the feature is discarded. Therefore a multi-lingual classifier model may gain additional strength from co-occurring words across languages.

3 Question 1

Can we reliably predict sentence-level subjectivity in languages other than English, by leveraging on a manually annotated English dataset?

In (Banea et al., 2008), we explored several methods for porting subjectivity annotated data from a source language (English) to a target language (Romanian and Spanish). Here, we focus on the transfer of manually annotated corpora through the usage of machine translation by projecting the original sentence level annotations onto the generated parallel text in the target language. Our aim is not to improve on that method, but rather to verify that the results are reliable across a number of languages. Therefore, we conduct this experiment in several additional languages, namely French, German and Arabic, and compare the results with those obtained for Spanish and Romanian.

Table 2 shows the results obtained using Naïve Bayes classifiers trained in each language individually, with a macro accuracy ranging from 71.30% (for Arabic) to 73.89% (for English).[5] As expected, the English machine learner outperforms those trained on other languages, as the original language of the annotations is English. However, it is worth noting that all measures do not deviate by more than 3.27%, implying that classifiers built using this technique exhibit a consistent behavior across languages.

4 Question 2

Can we improve the English subjectivity classification by expanding the feature space through the use of multilingual data? Similarly, can we also improve the classifiers in the other target languages?

We now turn towards investigating the impact on subjectivity classification of an expanded feature space through the inclusion of multilingual data. In order to methodically assess classifier behavior, we generate multiple datasets containing all pos-

[5] Note that the experiments conducted in (Banea et al., 2008) were made on a different test set, and thus the results are not directly comparable across the two papers.

No lang	SubjP	SubjR	SubjF	ObjP	ObjR	ObjF	AllP	AllR	AllF
1	73.43%	82.76%	77.82%	74.70%	62.97%	68.33%	74.07%	72.86%	73.08%
2	74.59%	83.14%	78.63%	75.70%	64.97%	69.92%	75.15%	74.05%	74.28%
3	75.04%	83.27%	78.94%	76.06%	65.75%	70.53%	75.55%	74.51%	74.74%
4	75.26%	83.36%	79.10%	76.26%	66.10%	70.82%	75.76%	74.73%	74.96%
5	75.38%	83.45%	79.21%	76.41%	66.29%	70.99%	75.90%	74.87%	75.10%
6	**75.43%**	**83.66%**	**79.33%**	**76.64%**	**66.30%**	**71.10%**	**76.04%**	**74.98%**	**75.21%**

Table 3: Average measures for a particular number of languages in a combination (from one through six) for Naïve Bayes classifiers using a multilingual space

sible combinations of one through six languages, as described in Section 2. We then train Naïve Bayes learners on the multilingual data and average our results per each group comprised of a particular number of languages. For example, for one language, we have the six individual classifiers described in Section 3; for the group of three languages, the average is calculated over 20 possible combinations; and so on.

Table 3 shows the results of this experiment. We can see that the overall F-measure increases from 73.08% – which is the average over one language – to 75.21% when all languages are taken into consideration (8.6% error reduction). We measured the statistical significance of these results by considering on one side the predictions made by the best performing classifier for one language (i.e., English), and on the other side the predictions made by the classifier trained on the multilingual space composed of all six languages. Using a paired t-test, the improvement was found to be significant at $p = 0.001$. It is worth mentioning that both the subjective and the objective precision measures increase to 75% when more than 3 languages are considered, while the overall recall level stays constant at 74%.

To verify that the improvement is due indeed to the addition of multilingual features, and it is not a characteristic of the classifier, we also tested two other classifiers, namely KNN and Rocchio. Figure 1 shows the average macro-accuracies obtained with these classifiers. For all the classifiers, the accuracies of the multilingual combinations exhibit an increasing trend, as a larger number of languages is used to predict the subjectivity annotations. The Naïve Bayes algorithm has the best performance, and a relative error rate reduction in accuracy of 8.25% for the grouping formed of six languages versus one, while KNN and Rocchio exhibit an error rate reduction of 5.82% and 9.45%, respectively. All of these reductions are statistically significant.

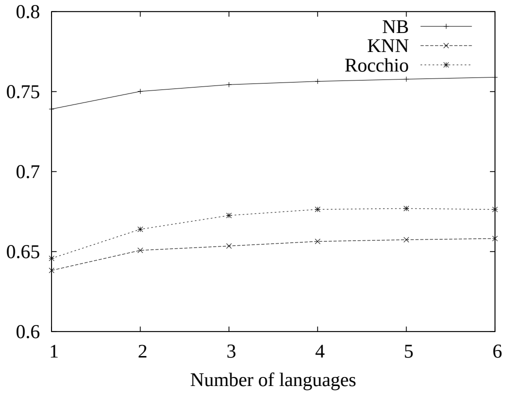


Figure 1: Average Macro-Accuracy per group of languages (combinations of 6 taken one through six)

In order to assess how the proposed multilingual expansion improves on the individual language classifiers, we select one language at a time to be the reference, and then compute the average accuracies of the Naïve Bayes learner across all the language groupings (from one through six) that contain the language. The results from this experiment are illustrated in Figure 2. The baseline in this case is represented by the accuracy obtained with a classifier trained on only one language (this corresponds to 1 on the X-axis). As more languages are added to the feature space, we notice a steady improvement in performance. When the language of reference is Arabic, we obtain an error reduction of 15.27%; 9.04% for Ro-

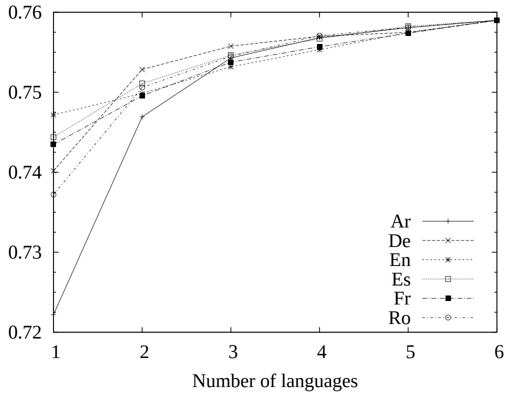


Figure 2: Average macro-accuracy progression relative to a given language

manian; 7.80% for German; 6.44% for French; 6.06% for Spanish; and 4.90 % for English. Even if the improvements seem minor, they are consistent, and the use of a multilingual feature set enables every language to reach a higher accuracy than individually attainable.

In terms of the best classifiers obtained for each grouping of one through six, English provides the best accuracy among individual classifiers (74.71%). When considering all possible combinations of six classifiers taken two, German and Spanish provide the best results, at 75.67%. Upon considering an additional language to the mix, the addition of Romanian to the German-Spanish classifier further improves the accuracy to 76.06%. Next, the addition of Arabic results in the best performing overall classifier, with an accuracy of 76.22%. Upon adding supplemental languages, such as English or French, no further improvements are obtained. We believe this is the case because German and Spanish are able to expand the dimensionality conferred by English alone, while at the same time generating a more orthogonal space. Incrementally, Romanian and Arabic are able to provide high quality features for the classification task. This behavior suggests that languages that are somewhat further apart are more useful for multilingual subjectivity classification than intermediary languages.

5 Question 3

Can we train a high precision classifier with a good recall level which could be used to generate subjectivity datasets in the target languages?

Since we showed that the inclusion of multilingual information improves the performance of subjectivity classifiers for all the languages involved, we further explore how the classifiers' predictions can be combined in order to generate high-precision subjectivity annotations. As shown in previous work, a high-precision classifier can be used to automatically generate subjectivity annotated data (Riloff and Wiebe, 2003). Additionally, the data annotated with a high-precision classifier can be used as a seed for bootstrapping methods, to further enrich each language individually.

We experiment with a majority vote meta-classifier, which combines the predictions of the *monolingual* Naïve Bayes classifiers described in Section 3. For a particular number of languages (one through six), all possible combinations of languages are considered. Each combination suggests a prediction only if its component classifiers agree, otherwise the system returns an "unknown" prediction. The averages are computed across all the combinations featuring the same number of languages, regardless of language identity.

The results are shown in Table 4. The macro precision and recall averaged across groups formed using a given number of languages are presented in Figure 3. If the average monolingual classifier has a precision of 74.07%, the precision increases as more languages are considered, with a maximum precision of 83.38% obtained when the predictions of all six languages are considered (56.02% error reduction). It is interesting to note that the highest precision meta-classifier for groups of two languages includes German, while for groups with more than three languages, both Arabic and German are always present in the top performing combinations. English only appears in the highest precision combination for one, five and six languages, indicating the fact that the predictions based on Arabic and German are more robust.

We further analyze the behavior of each language considering only those meta-classifiers that include the given language. As seen in Figure 4, all languages experience a boost in performance

No lang	SubjP	SubjR	SubjF	ObjP	ObjR	ObjF	AllP	AllR	AllF
1	73.43%	82.76%	77.82%	74.70%	62.97%	68.33%	74.07%	72.86%	73.08%
2	76.88%	76.39%	76.63%	80.17%	54.35%	64.76%	78.53%	65.37%	70.69%
3	78.56%	72.42%	75.36%	82.58%	49.69%	62.02%	80.57%	61.05%	68.69%
4	79.61%	69.50%	74.21%	84.07%	46.54%	59.89%	81.84%	58.02%	67.05%
5	80.36%	67.17%	73.17%	85.09%	44.19%	58.16%	82.73%	55.68%	65.67%
6	**80.94**%	65.20%	72.23%	**85.83**%	42.32%	56.69%	**83.38**%	53.76%	64.46%

Table 4: Average measures for a particular number of languages in a combination (from one through six) for meta-classifiers

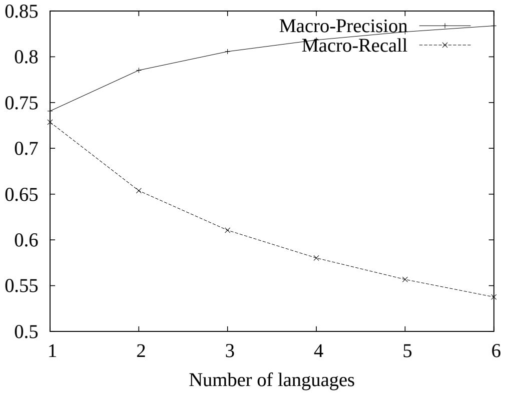


Figure 3: Average Macro-Precision and Recall across a given number of languages

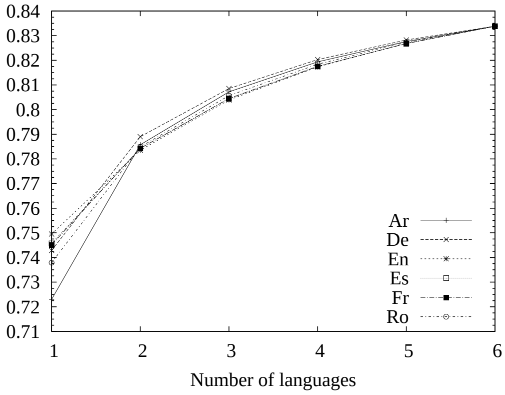


Figure 4: Average Macro-Precision relative to a given language

as a result of paired language reinforcement. Arabic gains an absolute 11.0% in average precision when considering votes from all languages, as compared to the 72.30% baseline consisting of the precision of the classifier using only monolingual features; this represents an error reduction in precision of 66.71%. The other languages experience a similar boost, including English which exhibits an error reduction of 50.75% compared to the baseline. Despite the fact that with each language that is added to the meta-classifier, the recall decreases, even when considering votes from all six languages, the recall is still reasonably high at 53.76%.

The results presented in table 4 are promising, as they are comparable to the ones obtained in previous work. Compared to (Wiebe et al., 2005), who used a high-precision rule-based classifier on the English MPQA corpus (see Table 1), our method has a precision smaller by 3.32%, but a recall larger by 21.16%. Additionally, unlike (Wiebe et al., 2005), which requires language-specific rules, making it applicable only to English, our method can be used to construct a high-precision classifier in any language that can be connected to English via machine translation.

6 Related Work

Recently, resources and tools for sentiment analysis developed for English have been used as a starting point to build resources in other languages, via cross-lingual projections or monolingual and multi-lingual bootstrapping. Several directions were followed, focused on leveraging annotation schemes, lexica, corpora and automated annotation systems. The English annotation scheme developed by (Wiebe et al., 2005) for opinionated text lays the groundwork for the research carried out by (Esuli et al., 2008) when annotating expressions of private state in the Italian Content Annotation Bank. Sentiment and subjectivity lexica such as the one included with the OpinionFinder distribution (Wiebe and Riloff, 2005), the General Inquirer (Stone et al., 1967), or the SentiWordNet (Esuli and Sebastiani, 2006b) were transfered into Chinese (Ku et al., 2006; Wu, 2008) and into Romanian (Mihalcea et al., 2007). English corpora manually annotated for subjectivity or sentiment such as MPQA (Wiebe et al., 2005), or the multi-domain sentiment classification corpus (Blitzer et al., 2007) were subjected to experiments in Spanish, Romanian, or Chinese upon automatic translation by (Banea et al., 2008; Wan, 2009). Furthermore, tools developed for English were used to determine sentiment or subjectivity labeling for a given target language by transferring the text to English and applying an English classifier on the resulting data. The labels were then transfered back into the target language (Bautin et al., 2008; Banea et al., 2008). These experiments are carried out in Arabic, Chinese, English, French, German, Italian, Japanese, Korean, Spanish, and Romanian.

The work closest to ours is the one proposed by (Wan, 2009), who constructs a polarity co-training system by using the multi-lingual views obtained through the automatic translation of product-reviews into Chinese and English. While this work proves that leveraging cross-lingual information improves sentiment analysis in Chinese over what could be achieved using monolingual resources alone, there are several major differences with respect to the approach we are proposing here. First, our training set is based solely on the automatic translation of the English corpus. We do not require an in-domain dataset available in the target language that would be needed for the co-training approach. Our method is therefore transferable to any language that has an English-to target language translation engine. Further, we focus on using multi-lingual data from six languages to show that the results are reliable and replicable across each language and that multiple languages aid not only in conducting subjectivity research in the target language, but also in improving the accuracy in the source language as well. Finally, while (Wan, 2009) research focuses on polarity detection based on reviews, our work seeks to determine sentence-level subjectivity from raw text.

7 Conclusion

Our results suggest that including multilingual information when modeling subjectivity can not only extrapolate current resources available for English into other languages, but can also improve subjectivity classification in the source language itself. We showed that we can improve an English classifier by using out-of-language features, thus achieving a 4.90% error reduction in accuracy with respect to using English alone. Moreover, we also showed that languages other than English can achieve an F-measure in subjectivity annotation of over 75%, without using any manually crafted resources for these languages. Furthermore, by combining the predictions made by monolingual classifiers using a majority vote learner, we are able to generate sentence-level subjectivity annotated data with a precision of 83% and a recall level above 50%. Such high-precision classifiers may be later used not only to create subjectivity-annotated data in the target language, but also to generate the seeds needed to sustain a language-specific bootstrapping.

To conclude and provide an answer to the question formulated in the title, more languages are better, as they are able to complement each other, and together they provide better classification results. When one language cannot provide sufficient information, another one can come to the rescue.

Acknowledgments

This material is based in part upon work supported by National Science Foundation awards #0917170 and #0916046. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Alm, Cecilia Ovesdotter, Dan Roth, and Richard Sproat. 1990. Emotions from text: machine learning for text-based emotion prediction. *Intelligence*.

Balog, Krisztian, Gilad Mishne, and Maarten De Rijke. 2006. Why Are They Excited? Identifying and Explaining Spikes in Blog Mood Levels. In *Proceedings of the*

11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-2006), Trento, Italy.

Banea, Carmen, Rada Mihalcea, Janyce Wiebe, and Samer Hassan. 2008. Multilingual Subjectivity Analysis Using Machine Translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-2008)*, pages 127–135, Honolulu.

Bautin, Mikhail, Lohit Vijayarenu, and Steven Skiena. 2008. International Sentiment Analysis for News and Blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM-2008)*, Seattle, Washington.

Blitzer, John, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational (ACL-2007)*, pages 440–447, Prague, Czech Republic. Association for Computational Linguistics.

Carenini, Giuseppe, Raymond T Ng, and Xiaodong Zhou. 2008. Summarizing Emails with Conversational Cohesion and Subjectivity. In *Proceedings of the Association for Computational Linguistics: Human Language Technologies (ACL- HLT 2008)*, pages 353–361, Columbus, Ohio.

Esuli, Andrea and Fabrizio Sebastiani. 2006a. Determining Term Subjectivity and Term Orientation for Opinion Mining. In *Proceedings of the 11th Meeting of the European Chapter of the Association for Computational Linguistics (EACL-2006)*, volume 2, pages 193–200, Trento, Italy.

Esuli, Andrea and Fabrizio Sebastiani. 2006b. SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. In *Proceedings of the 5th Conference on Language Resources and Evaluation*, pages 417–422.

Esuli, Andrea, Fabrizio Sebastiani, and Ilaria C Urciuoli. 2008. Annotating Expressions of Opinion and Emotion in the Italian Content Annotation Bank. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC-2008)*, Marrakech, Morocco.

Hu, Minqing and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of ACM Conference on Knowledge Discovery and Data Mining (ACM-SIGKDD-2004)*, pages 168–177, Seattle, Washington.

Ku, Lun-wei, Yu-ting Liang, and Hsin-hsi Chen. 2006. Opinion Extraction, Summarization and Tracking in News and Blog Corpora. In *Proceedings of AAAI-2006 Spring Symposium on Computational Approaches to Analyzing Weblogs*, number 2001, Boston, Massachusetts.

Lloyd, Levon, Dimitrios Kechagias, and Steven Skiena, 2005. *Lydia : A System for Large-Scale News Analysis (Extended Abstract) News Analysis with Lydia*, pages 161–166. Springer, Berlin / Heidelberg.

Mihalcea, Rada, Carmen Banea, and Janyce Wiebe. 2007. Learning Multilingual Subjective Language via Cross-Lingual Projections. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL-2007)*, pages 976–983, Prague, Czech Republic.

Riloff, Ellen and Janyce Wiebe. 2003. Learning Extraction Patterns for Subjective Expressions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2003)*, pages 105–112, Sapporo, Japan.

Stone, Philip J, Marshall S Smith, Daniel M Ogilivie, and Dexter C Dumphy. 1967. *The General Inquirer: A Computer Approach to Content Analysis. /.* The MIT Press, 1st edition.

Wan, Xiaojun. 2009. Co-Training for Cross-Lingual Sentiment Classification. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2009)*, Singapore.

Wiebe, Janyce and Rada Mihalcea. 2006. Word Sense and Subjectivity. In *Proceedings of the joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING-ACL-2006)*, Sydney, Australia.

Wiebe, Janyce and Ellen Riloff. 2005. Creating Subjective and Objective Sentence Classifiers from Unannotated Texts. In *Proceeding of CICLing-05, International Conference on Intelligent Text Processing and Computational Linguistics*, pages 486–497, Mexico City, Mexico.

Wiebe, Janyce, Theresa Wilson, and Claire Cardie. 2005. Annotating Expressions of Opinions and Emotions in Language. *Language Resources and Evaluation*, 39(2-3):165–210.

Wu, Yejun. 2008. Classifying attitude by topic aspect for English and Chinese document collections.

Yu, Hong and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentence. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2003)*, pages 129–136, Sapporo, Japan.

Plagiarism Detection across Distant Language Pairs

Alberto Barrón-Cedeño Paolo Rosso
Natural Language Engineering Lab. - ELiRF
Universidad Politécnica de Valencia
{lbarron, prosso}@dsic.upv.es

Eneko Agirre Gorka Labaka
IXA NLP Group
Basque Country University
{e.agirre, gorka.labaka}@ehu.es

Abstract

Plagiarism, the unacknowledged reuse of text, does not end at language boundaries. Cross-language plagiarism occurs if a text is translated from a fragment written in a different language and no proper citation is provided. Regardless of the change of language, the contents and, in particular, the ideas remain the same. Whereas different methods for the detection of monolingual plagiarism have been developed, less attention has been paid to the cross-language case.

In this paper we compare two recently proposed cross-language plagiarism detection methods (CL-CNG, based on character n-grams and CL-ASA, based on statistical translation), to a novel approach to this problem, based on machine translation and monolingual similarity analysis (T+MA). We explore the effectiveness of the three approaches for less related languages. CL-CNG shows not be appropriate for this kind of language pairs, whereas T+MA performs better than the previously proposed models.

1 Introduction

Plagiarism is a problem in many scientific and cultural fields. Text plagiarism may imply different operations: from a simple cut-and-paste, to the insertion, deletion and substitution of words, up to an entire process of paraphrasing. Different models approach the detection of monolingual plagiarism (Shivakumar and García-Molina, 1995; Hoad and Zobel, 2003; Maurer et al., 2006). Each of these models is appropriate only in those cases where all the implied documents are written in the same language.

Nevertheless, the problem does not end at language boundaries. Plagiarism is also committed if the reused text is translated from a fragment written in a different language and no citation is provided. When plagiarism is generated by a translation process, it is known as cross-language plagiarism (CLP).

Less attention has been paid to the detection of this kind of plagiarism due to its enhanced difficulty (Ceska et al., 2008; Barrón-Cedeño et al., 2008; Potthast et al., 2010). In fact, in the recently held 1st International Competition on Plagiarism Detection (Potthast et al., 2009), no participants tried to approach it.

In order to describe the prototypical process of automatic plagiarism detection, we establish the following notation. Let d_q be a plagiarism suspect document. Let D be a representative collection of reference documents. D presumably includes the source of the potentially plagiarised fragments in d_q. Stein et al., (2007) divide the process into three stages[1]:

1. *heuristic retrieval of potential source documents*: given d_q, retrieving an appropriate number of its potential source documents $D^* \in D$ such that $|D^*| \lll |D|$;
2. *exhaustive comparison of texts*: comparing the text from d_q and $d \in D^*$ in order to identify reused fragments and their potential

[1] This schema was formerly proposed for monolingual plagiarism detection. Nevertheless, it can be applied without further modifications to the cross-language case.

sources; and

3. *knowledge-based post-processing*: those detected fragments with proper citation are discarded as they are not plagiarised.

The result is offered to the human expert to take the final decision. In the case of cross-language plagiarism detection (CLPD), the texts are written in different languages: $d_q \in L$ and $d' \in L'$.

In this research we focus on step 2: *cross-language exhaustive comparison of texts*, approaching it as an Information Retrieval problem of cross-language text similarity. Step 1, *heuristic retrieval*, may be approached by different CLIR techniques, such as those proposed by Dumais et al. (1997) and Pouliquen et al. (2003).

Cross-language similarity between texts, $\varphi(d_q, d')$, has been previously estimated on the basis of different models: multilingual thesauri (Steinberger et al., 2002; Ceska et al., 2008), comparable corpora —CL-Explicit Semantic Analysis CL-ESA— (Potthast et al., 2008), machine translation techniques —CL-Alignment-based Similarity Analysis CL-ASA— (Barrón-Cedeño et al., 2008; Pinto et al., 2009) and n-grams comparison —CL-Character n-Grams CL-CNG— (Mcnamee and Mayfield, 2004).

A comparison of CL-ASA, CL-ESA, and CL-CNG was carried out recently by Potthast et al. (2010). The authors report that in general, despite its simplicity, CL-CNG outperformed the other two models. Additionally, CL-ESA showed good results in the cross-language retrieval of topic-related texts, whereas CL-ASA obtained better results in exact (human) translations.

However, most of the language pairs used in the reported experiments (English-$\{$German, Spanish, French, Dutch, Polish$\}$) are related, whether because they have common predecessors or because a large proportion of their vocabularies share common roots. In fact, the lower syntactical relation between the English-Polish pair caused a performance degradation for CL-CNG, and for CL-ASA to a lesser extent. In order to confirm whether the closeness among languages is an important factor, this paper works with more distant language pairs: English-Basque and Spanish-Basque.

The rest of the paper is structured as follows. Section 2 describes the motivation for working on this research topic, stressing the situation of cross-language plagiarism among writers in less resourced languages. A brief overview of the few works on CLPD is included. The three similarity estimation models compared in this research work are presented in Section 3. The experimental framework and the obtained results are included in Section 4. Finally, Section 5 draws conclusions and discusses further work.

2 Motivation

Cases of CLP are common nowadays because information in multiple languages is available on the Web, but people still write in their own language. This special kind of plagiarism occurs more often when the target language is a less resourced one[2], as is the case of Basque.

Basque is a pre-indoeuropean language with less than a million speakers in the world and no known relatives in the language families (Wikipedia, 2010a). Still, Basque shares a portion of its vocabulary with its contact languages (Spanish and French). Therefore, we decided to work with two language pairs: Basque with Spanish, its contact language, and with English, perhaps the language with major influence over the rest of languages in the world. Although the considered pairs share most of their alphabet, the vocabulary and language typologies are very different. For instance Basque is an agglutinative language.

In order to illustrate the relations among these languages, Fig. 1 includes extracts from the English (*en*), Spanish (*es*) and Basque (*eu*) versions of the same Wikipedia article. The fragments are a sample of the lexical and syntactic distance between Basque and the other two languages. In fact, these sentences are completely co-derived and the corresponding entire articles are a sample of the typical imbalance in text available in the different languages (around $2,000$, $1,300$, and only

[2] Less resourced language is that with a low degree of representation on the Web (Alegria et al., 2009). Whereas the available text for German, French or Spanish is less than for English, the difference is more dramatic with other languages such as Basque.

The Party of European Socialists (PES) is a European political party comprising thirty-two socialist, social democratic and labour parties from each European Union member state and Norway.

El Partido Socialista Europeo (PSE) es un partido político pan-europeo cuyos miembros son de partidos socialdemócratas, socialistas y laboristas de estados miembros de la Unión Europea, así como de Noruega.

Europako Alderdi Sozialista Europar Batasuneko herrialdeetako eta Norvegiako hogeita hamahiru alderdi sozialista, sozialdemokrata eta laborista biltzen dituen alderdia da.

Figure 1: First sentences from the Wikipedia articles “Party of European Socialists” (*en*),“Partido Socialista Europeo” (*es*), and “Europako Alderdi Sozialista” (*eu*) (Wikipedia, 2010b).

100 words are contained in the *en*, *es* and *eu* articles, respectively).

Of high relevance is that the two corpora used in this work were manually constructed by translating English and Spanish text into Basque. In the experiments carried out by Potthast et al. (2010), which inspired our work, texts from the JCR-Acquis corpus (Steinberger et al., 2006) and Wikipedia were used. The first one is a multilingual corpus with no clear definition of source and target languages, whereas in Wikipedia no specific relationship exists between the different languages in which a topic may be broached. In some cases (cf. Fig. 1) they are clearly co-derived, but in others they are completely independent.

CLPD has been investigated just recently, mainly by adapting models formerly proposed for cross-language information retrieval. This is the case of cross-language explicit semantic analysis (CL-ESA), proposed by Potthast et al. (2008). In this case the comparison between texts is not carried out directly. Instead, a comparable corpus $C_{L,L'}$ is required, containing documents on multiple topics in the two implied languages. One of the biggest corpora of this nature is Wikipedia. The similarity between $d_q \in L$ and every document $c \in C_L$ is computed based on the cosine measure. The same process is made for L'. This step generates two vectors $[cos(d_q, c_1), \ldots, cos(d_q, c_{|C_L|})]$ and $[cos(d', c'_1), \ldots, cos(d', c'_{|C_{L'}|})]$, where each dimension is comparable between the two vectors. Therefore, the cosine between such vectors can be estimated in order to —indirectly— estimate how similar d_q and d' are. The authors suggest that this model can be used for CLPD.

Another recent model is *MLPlag*, proposed by Ceska et al. (2008). It exploits the *EuroWordNet Thesaurus*[3], that includes sets of synonyms in multiple European languages, with common identifiers across languages. The authors report experiments over a subset of documents of the English and Czech sections of the JRC-Acquis corpus as well as a corpus of simplified vocabulary[4]. The main difficulty they faced was the amount of words in the documents not included in the thesaurus (approximately 50% of the vocabulary).

This is a very similar approach to that proposed by Pouliquen et al. (2003) for the identification of document translations. In fact, both approaches have something in common: translations are searched at document level. It is assumed that an entire document has been reused (translated). Nevertheless, a writer is free to plagiarise text fragments from different sources, and compose a mixture of original and reused text.

A third model is the cross-language alignment-based similarity analysis (CL-ASA), proposed by Barrón-Cedeño et al. (2008), which is based on statistical machine translation technology. This model was proposed to detect plagiarised text fragments (similar models have been proposed for extraction of parallel sentences from comparable corpora (Munteanu et al., 2004)). The authors report experiments over a short set of texts from which simulated plagiarism was created from English to Spanish. Human as well as automatic machine translations were included in the collection. Further descriptions of this model are included in Section 3, as it is one of those being assessed in this research work.

To the best of our knowledge, no work (including the three previously mentioned) has been done considering less resourced languages. In this research work we approach the not uncommon problem of CLPD in Basque, with source texts written in Spanish (the co-official language of the

[3] `http://www.illc.uva.nl/EuroWordNet/`

[4] The authors do not mention the origin of the documents.

	low	*tok*	*pd*	*bd*	*sd*	*lem*
T+MA	■	■				■
CL-ASA	■	■				■
CL-CNG	■		■	■	■	

Table 1: Text preprocessing operations required for the different models. low=lowercasing, tok=tokenization, pd=punctuation marks deletion, bd=blank space deletion, sd=symbols deletion, lem=lematization.

Basque Country) and English (the language with most available texts in the world).

We compare three cross-language similarity analysis methods: T+MA (translation followed by monolingual analysis), a novel method based on machine translation followed by a monolingual similarity estimation; CL-CNG, a character n-gram based comparison model; and CL-ASA a model that combines translation and similarity estimation in a single step. Neither MLPlag nor CL-ESA are included in the comparison. On the one hand, we are interested in plagiarism at sentence level, and MLPlag is designed to compare entire documents. On the other hand, in previous experiments over exact translations, CL-ASA has shown to outperform it on language pairs whose alphabet or syntax are unrelated (Potthast et al., 2010). This is precisely the case of *en-eu* and *es-eu* language pairs. Additionally, the amount of Wikipedia articles in Basque available for the construction of the required comparable corpus is insufficient for the CL-ESA data requirements.

3 Definition of Models

In this section, we describe the three cross-language similarity models we compare. For experimental purposes (cf. Section 4) we consider d_q to be a suspicious sentence written in L and D' to be a collection of potential source sentences written in L' ($L \neq L'$). The text pre-processing required by the different models is summarised in Table 1. Examples illustrating how the models work are included in Section 4.3.

3.1 Translation + Monolingual Analysis

$d_q \in L$ is translated into L' on the basis of the Giza++ (Och and Ney, 2003), Moses (Koehn et al., 2007) and SRILM (Stolcke, 2002) tools, generating d'_q. The translation system uses a log-linear combination of state-of-the-art features, such as translation probabilities and lexical translation models on both directions and a target language model. After translation, d'_q and d' are lexically related, making possible a monolingual comparison.

Multiple translations from d_q into d'_q are possible. Therefore, performing a monolingual similarity analysis based on "traditional" techniques, such as those based on word n-grams comparison (Broder, 1997) or hash collisions (Schleimer et al., 2003), is not an option. Instead, we take the approach of the bag-of-words, which has shown good results in the estimation of monolingual text similarity (Barrón-Cedeño et al., 2009). Words in d'_q and d' are weighted by the standard $tf\text{-}idf$, and the similarity between them is estimated by the cosine similarity measure.

3.2 CL-Alignment-based Similarity Analysis

In this model an estimation of how likely is that d' is a translation of d_q is performed. It is based on the adaptation of the Bayes rule for MT:

$$p(d' \mid d_q) = \frac{p(d')\ p(d_q \mid d')}{p(d_q)}. \quad (1)$$

As $p(d_q)$ does not depend on d', it is neglected. From an MT point of view, the conditional probability $p(d_q \mid d')$ is known as *translation model probability* and is computed on the basis of a statistical bilingual dictionary. $p(d')$ is known as *language model probability*; it describes the target language L' in order to obtain grammatically acceptable translations (Brown et al., 1993).

Translating d_q into L' is not the concern of this method, rather it focuses on retrieving texts written in L' which are potential translations of d_q. Therefore, Barrón-Cedeño et al. (2008) proposed replacing the language model (the one used in T+MA) by that known as *length model*. This model depends on text's character lengths instead of language structures.

Multiple translations from d into L' are possible, and it is uncommon to find a pair of translated texts d and d' such that $|d| = |d'|$. Nevertheless, the length of such translations is closely related to a translation length factor. In accordance with Pouliquen et al. (2003), the length model is defined as:

$$\varrho(d') = e^{-0.5\left(\frac{\frac{|d'|}{|d_q|}-\mu}{\sigma}\right)^2}, \tag{2}$$

where μ and σ are the mean and the standard deviation of the character lengths between translations of texts from L into L'. If the length of d' is not the expected given d_q, it receives a low qualification.

The translation model probability is defined as:

$$p(d \mid d') = \prod_{x \in d} \sum_{y \in d'} p(x, y), \tag{3}$$

where $p(x, y)$, a statistical bilingual dictionary, represents the likelihood that x is a valid translation of y. After estimating $p(x, y)$ from a parallel corpus, on the basis of the IBM statistical translation models (Brown et al., 1993), we consider, for each word x, only the k best translations y (those with the highest probabilities) up to a minimum probability mass of 0.4. This threshold was empirically selected as it eliminated noisy entries without discarding an important amount of relevant pairs.

The similarity estimation based on CL-ASA is finally computed as:

$$\varphi(d_q, d') = \varrho(d')\ p(d_q \mid d'). \tag{4}$$

3.3 CL-Character n-Gram Analysis

This model, the simplest of those compared in this research, has been used in (monolingual) Authorship Attribution (Keselj et al., 2003) as well as cross-language Information Retrieval (Mcnamee and Mayfield, 2004). The simplified alphabet considered is $\Sigma = \{a, \ldots, z, 0, \ldots, 9\}$; any other symbol is discarded (cf. Table 1). The resulting text strings are codified into character 3-grams, which are weighted by the standard tf-idf (considering this n has previously shown to produce the best results). The similarity between such representations of d_q and d' is estimated by the cosine similarity measure.

4 Experiments

The objective of our experiments is to compare the performance of the three similarity estimation models. Section 4.1 introduces the corpora we have exploited. The experimental framework is described in Section 4.2. Section 4.3 illustrates how the models work, and the obtained results are presented and discussed in Section 4.4.

4.1 Corpora

In other Information Retrieval tasks a plethora of corpora is available for experimental and comparison purposes. However, plagiarism implies an ethical infringement and, to the best of our knowledge, there is no corpora of actual cases available, other than some seminal efforts on creating corpora of text reuse (Clough et al., 2002), artificial plagiarism (Potthast et al., 2009), and simulated plagiarism (Clough and Stevenson, 2010). The problem is worse for cross-language plagiarism.

Therefore, in our experiments we use two parallel corpora: *Software*, an *en-eu* translation memory of software manuals generously supplied by Elhuyar Fundazioa[5]; and *Consumer*, a corpus extracted from a consumer oriented magazine that includes articles written in Spanish along with their Basque, Catalan, and Galician translations[6] (Alcázar, 2006). *Software* includes $288,000$ parallel sentences; 8.66 (6.83) words per sentence in the English (Basque) section. *Consumer* contains $58,202$ sentences; 19.77 (15.20) words per sentence in Spanish (Basque). These corpora also reflect the imbalance of text available in the different languages.

4.2 Experimental Framework

We consider D_q and D' to be two entire documents from which plagiarised sentences and their source are to be detected. We work at this level of granularity, and not entire documents, for two main reasons: (*i*) we are focused on the exhaustive comparison stage of the plagiarism detection process (cf. Section 1); and (*ii*) even a single sentence could be considered a case of plagiarism, as it transmits a complete idea. However, a plagiarised sentence is usually not enough to automatically negate the validity of an entire document. This decision is left to the human expert, which can examine the documents where several plagiarised sentences occur. Note that the task becomes computationally more expensive as, for every sentence, we are looking through thousands

[5] `http://www.elhuyar.org`

[6] `http://revista.consumer.es`

	es-eu		en-eu	
	μ	σ	μ	σ
f_1	1.1567	0.2346	1.0561	0.5497
f_2	1.1569	0.2349	1.0568	0.5510
f_3	1.1571	0.2349	1.0566	0.5433
f_4	1.1565	0.2363	1.0553	0.5352
f_5	1.1571	0.2348	1.0553	0.5467
$avg.$	1.1569	0.2351	1.0560	0.5452

Table 2: Length models estimated for each training partition $f_{1,\ldots,5}$. The values describe a normal distribution centred in $\mu \pm \sigma$, representing the expected length of the source text given the suspicious one.

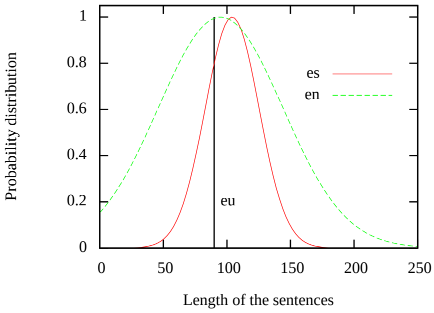


Figure 2: Example length factor for a sentence written in Basque (*eu*) d_q, such that $|d_q| = 90$. The normal distributions represent the expected lengths for the translation d', either in Spanish (*es*) or English (*en*).

of topically-related sentences that are potential sources of d_q, and not only those of a specific document.

CLPD is considered a ranking problem. Let $d_q \in D_q$ be a plagiarism suspicious sentence and $d' \in D'$ be its source sentence. We consider that the result of the process is correct if, given d_q, d' is properly retrieved. A 5-fold cross validation for both *en-eu* and *es-eu* was performed. Bilingual dictionaries, language and length models were estimated with the corresponding training partitions. The computed values for μ and σ are those included in Table 2. The values for the different partitions are very similar, showing the low variability in the translation lengths. On the basis of these estimated parameters, an example of length factor for a specific sentence is plotted in Fig. 2.

In the test partitions, for each suspicious sentence d_q, $11,640$ source candidate sentences exist for *es-eu* and $57,290$ for *en-eu*. This results in more than 135 million and 3 billion comparisons carried out for *es-eu* and *en-eu* respectively.

x_{eu}	y_{en}	$p(x,y)$	x_{eu}	y_{en}	$p(x,y)$
beste	another	0.288	**beste**	**other**	**0.348**
dokumentu	**document**	**0.681**	batzu	some	0.422
makro	**macro**	**0.558**	**ezin**	**not**	**0.179**
ezin	cannot	0.279	izan	is	0.241
izan	the	0.162	atzi	access	0.591
.	.	**0.981**			

Table 3: Entries in the bilingual dictionary for the words in d_q. Relevant entries for the example are in bold.

4.3 Illustration of Models

In order to clarify how the different models work, consider the following sentence pair, a suspicious sentence d_q written in Basque and its source d' written in English (sentences are short for illustrative purposes):

d_q *beste dokumentu batzuetako makroak ezin dira atzitu.*
d' *macros from other documents are not accessible.*

CL-CNG Example

In this case, symbols and spaces are discarded. Sentences become:

d_q *bestedokumentubatzuetakomakroakezindiraatzitu*
d' *macrosfromotherdocumentsarenotaccessible*

Only three 3-grams appear in both sentences (*ume*, *men*, *ent*). In order to keep the example simple, the 3-grams are weighted by tf only (in the actual experiments, tf-idf is used), resulting in a dot product of 3. The corresponding vectors magnitudes are $|\mathbf{d_q}| = 6.70$ and $|\mathbf{d'}| = 5.65$. Therefore, the estimated similarity is $\varphi(d_q, d') = 0.079$.

CL-ASA Example

In this case, the text must be tokenised and lemmatised, resulting in the following string:

d_q *beste dokumentu batzu makro ezin izan atzi .*
d' *macro from other document be not accessible .*

The sentences' lengths are $|d_q| = 38$ and $|d'| = 39$. Therefore, on the basis of Eq. 2, the length factor between them is $\varrho(d_q, d') = 0.998$.

The relevant entries of the previously estimated dictionary are included in Table 3. Such entries are substituted in Eq. 3, and the overall process results in a similarity $\varphi(d_q, d') = 2.74$. Whereas not a stochastic value, this is a weight used when ranking all the potential source sentences in D'.

T+MA Example

In this case, the same pre-processing than in CL-ASA is performed. In T+MA d_q is translated into L', resulting in the new pair:

d'_q *other document macro cannot be access .*
d' *macro from other document be not accessible .*

Note that d'_q is a valid translation of d_q. Nevertheless, it has few syntactic relation to d'. Therefore, applying more sophisticated codifications than the cosine measure over bag-of-words is not an option. The example is again simplified by weighting the words based on tf. Five words appear in both sentences, resulting in a dot product of 5. The vectors magnitudes are $|\mathbf{d'_q}| = |\mathbf{d'}| = \sqrt{7}$. The estimation by T+MA is $\varphi(d_q, d') = 0.71$, a high similarity level.

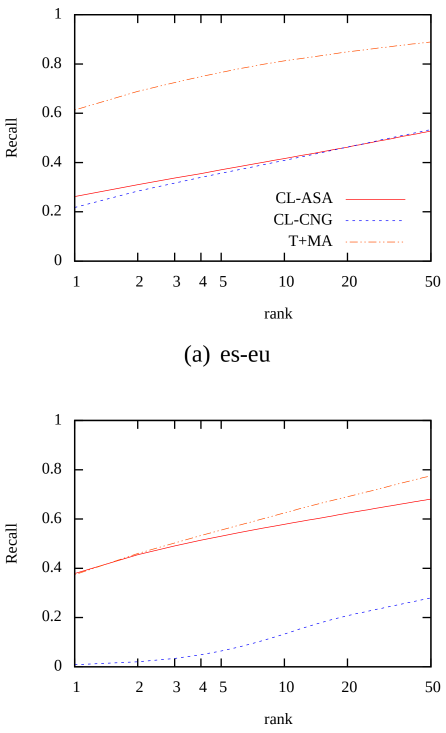
4.4 Results and Discussion

For evaluation we consider a standard measure: Recall. More specifically Recall after n texts have been retrieved ($n = [1 \ldots, 50]$). Figure 3 plots the average Recall value obtained in the 5-folds with respect to the rank position (n).

In both language pairs, CL-CNG obtained worse results than those reported for English-Polish by Potthast et al. (2010): $R@50 = 0.68$ vs. $R@50 = 0.53$ for *es-eu* and 0.28 for *en-eu*. This is due to the fact that neither the vocabulary nor its corresponding roots keep important relations. Therefore, when language pairs have a low syntactical relationship, CL-CNG is not an option. Still, CL-CNG performs better with *es-eu* than with *en-eu* because the first pair is composed of contact languages (cf. Section 1).

About CL-ASA, the results obtained with *es-eu* and *en-eu* are quite different: $R@50 = 0.68$ for *en-eu* and $R@50 = 0.53$ for *es-eu*. Whereas in the first case they are comparable to those of CL-CNG, in the second one CL-ASA completely outperforms it. The improvement of CL-ASA obtained for *en-eu* is due to the size of the training corpus available in this case (approximately five times the number of sentences available for *es-eu*). This shows the sensitivity of the model with respect to the size of the available resources.

Lastly, although T+MA is a simple approach that reduces the cross-language similarity estimation to a translation followed by a monolingual process, it obtained a good performance (R@50= 0.77 for *en-eu* and R@50=0.89 for es-eu). Moreover, this method proved to be less sensitive than CL-ASA to the lack of resources. This could be due to the fact that it considers both directions of the translation model (*e[n|s]-eu* and *eu-e[n|s]*). Additionally, the language model, applied in order to compose syntactically correct translations, reduces the amount of wrong translations and, indirectly, includes more syntactic information in the process. On the contrary, CL-ASA only considers one direction translation model *eu-e[n|s]* and completely disregards syntactical relations between the texts.



(a) es-eu

(b) en-eu

Figure 3: Evaluation of the cross-language ranking. Results plotted as rank versus Recall for the three evaluated models and the two language pairs ($R@[1, \ldots, 50]$).

Note that the better results come at the cost of higher computational demand. CL-CNG only requires easy to compute string comparisons. CL-ASA requires translation probabilities from aligned corpora, but once the probabilities are estimated, cross-language similarity can be computed very fast. T+MA requires the previous translation of all the texts, which can be very costly for large collections.

5 Conclusions and Further Work

In a society where information in multiple languages is available on the Web, cross-language plagiarism is occurring every day with increasing frequency. Still, cross-language plagiarism detection has not been approached sufficiently due to its intrinsic complexity. Though few attempts have been made, even less work has been made to tackle this problem for less resourced languages, and to explore distant language pairs.

We investigated the case of Basque, a language where, due to the lack of resources, cross-language plagiarism is often committed from texts in Spanish and English. Basque has no known relatives in the language family. However, it shares some of its vocabulary with Spanish.

Two state-of-the-art methods based on translation probabilities and n-gram overlapping, and a novel technique based on statistical machine translation were evaluated. The novel technique obtains the best results in both language pairs, with the n-gram overlap technique performing worst. In this sense, our results complement those of Potthast et al. (2010), which includes closely related language pairs as well.

Our results also show that better results come at the cost of more expensive processing time. For the future, we would like to investigate such performance trade-offs in more demanding datasets.

For future work we consider that exploring semantic text features across languages could improve the results. It could be interesting to further analyse how the reordering of words through translations might be relevant for this task. Additionally, working with languages even more distant from each other, such as Arabic or Hindi, seems to be a challenging and interesting task.

Acknowledgements

The research work of the first two authors is partially funded by CONACYT-Mexico and the MICINN project TEXT-ENTERPRISE 2.0 TIN2009-13391-C04-03 (Plan I+D+i). The research work of the last two authors is partially funded by the MICINN projects OPENMT-2 TIN2009-14675-C03-01 and KNOW2 TIN2009-14715-C04-01.

References

Alcázar, Asier. 2006. Towards Linguistically Searchable Text. In *Proceedings of the BIDE 2005*, Bilbao, Basque Country.

Alegria, Iñaki, Mikel L. Forcada, and Kepa Sarasola, editors. 2009. *Proceedings of the SEPLN 2009 Workshop on Information Retrieval and Information Extraction for Less Resourced Languages*, Donostia, Basque Country. University of the Basque Country.

Barrón-Cedeño, Alberto, Paolo Rosso, David Pinto, and Alfons Juan. 2008. On Cross-lingual Plagiarism Analysis Using a Statistical Model. In Stein, Stamatatos, and Koppel, editors, *ECAI 2008 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 2008)*, pages 9–13, Patras, Greece. CEUR-WS.org.

Barrón-Cedeño, Alberto, Andreas Eiselt, and Paolo Rosso. 2009. Monolingual Text Similarity Measures: A Comparison of Models over Wikipedia Articles Revisions. In Sharma, Verma, and Sangal, editors, *ICON 2009*, pages 29–38, Hyderabad, India. Macmillan Publishers.

Broder, Andrei Z. 1997. On the Resemblance and Containment of Documents. In *Compression and Complexity of Sequences (SEQUENCES'97)*, pages 21–29. IEEE Computer Society.

Brown, Peter F., Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

Ceska, Zdenek, Michal Toman, and Karel Jezek. 2008. Multilingual Plagiarism Detection. In *Proceedings of the 13th International Conference on Artificial Intelligence*, pages 83–92. Springer Verlag Berlin Heidelberg.

Clough, Paul and Mark Stevenson. 2010. Developing a Corpus of Plagiarised Short Answers. *Language Resources and Evaluation: Special Issue on Plagiarism and Authorship Analysis*.

Clough, Paul, Robert Gaizauskas, and Scott Piao. 2002. Building and Annotating a Corpus for the Study of Journalistic Text Reuse. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, volume V, pages 1678–1691, Las Palmas, Spain.

Dumais, Susan T., Todd A. Letsche, Michael L. Littman, and Thomas K. Landauer. 1997. Automatic Cross-Language Retrieval Using Latent Semantic Indexing. In *AAAI-97 Spring Symposium Series: Cross-Language Text and Speech Retrieval*, pages 24—26. Stanford University.

Hoad, Timothy C. and Justin Zobel. 2003. Methods for Identifying Versioned and Plagiarized Documents. *Journal of the American Society for Information Science and Technology*, 54(3):203–215.

Keselj, Vlado, Fuchun Peng, Nick Cercone, and Calvin Thomas. 2003. N-gram-based Author Profiles for Authorship Attribution. In *Proceedings of the Conference Pacific Association for Computational Linguistics, PACLING'03*, pages 255–264, Halifax, Canada.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Annual Meeting of the Association for Computational Linguistics (ACL), demonstration session*, Prague, Czech Republic.

Maurer, Hermann, Frank Kappe, and Bilal Zaka. 2006. Plagiarism - A Survey. *Journal of Universal Computer Science*, 12(8):1050–1084.

Mcnamee, Paul and James Mayfield. 2004. Character N-Gram Tokenization for European Language Text Retrieval. *Information Retrieval*, 7(1-2):73–97.

Munteanu, Dragos S., Alexander Fraser, and Daniel Marcu. 2004. Improved Machine Translation Performace via Parallel Sentence Extraction from Comparable Corpora. In *Proceedings of the Human Language Technology and North American Association for Computational Linguistics Conference (HLT/NAACL 2004)*, Boston, MA.

Och, Frank Josef and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51. See also http://www.fjoch.com/GIZA++.html.

Pinto, David, Jorge Civera, Alberto Barrón-Cedeño, Alfons Juan, and Paolo Rosso. 2009. A Statistical Approach to Crosslingual Natural Language Tasks. *Journal of Algorithms*, 64(1):51–60.

Potthast, Martin, Benno Stein, and Maik Anderka. 2008. A Wikipedia-Based Multilingual Retrieval Model. In Macdonald, Ounis, Plachouras, Ruthven, and White, editors, *30th European Conference on IR Research, ECIR 2008, Glasgow*, volume 4956 LNCS of *Lecture Notes in Computer Science*, pages 522–530, Berlin Heidelberg New York. Springer.

Potthast, Martin, Benno Stein, Andreas Eiselt, Alberto Barrón-Cedeño, and Paolo Rosso. 2009. Overview of the 1st International Competition on Plagiarism Detection. In Stein, Rosso, Stamatatos, Koppel, and Agirre, editors, *SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09)*, pages 1–9, San Sebastian, Spain. CEUS-WS.org.

Potthast, Martin, Alberto Barrón-Cedeño, Benno Stein, and Paolo Rosso. 2010. Cross-Language Plagiarism Detection. *Language Resources and Evaluation, Special Issue on Plagiarism and Authorship Analysis*.

Pouliquen, Bruno, Ralf Steinberger, and Camelia Ignat. 2003. Automatic Identification of Document Translations in Large Multilingual Document Collections. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-2003)*, pages 401–408, Borovets, Bulgaria.

Schleimer, Saul, Daniel S. Wilkerson, and Alex Aiken. 2003. Winnowing: Local Algorithms for Document Fingerprinting. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, New York, NY. ACM.

Shivakumar, Narayanan and Hector García-Molina. 1995. SCAM: A Copy Detection Mechanism for Digital Documents. In *Proceedings of the 2nd Annual Conference on the Theory and Practice of Digital Libraries*.

Stein, Benno, Sven Meyer zu Eissen, and Martin Potthast. 2007. Strategies for Retrieving Plagiarized Documents. In Clarke, Fuhr, Kando, Kraaij, and de Vries, editors, *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 825–826, Amsterdam, The Netherlands. ACM.

Steinberger, Ralf, Bruno Pouliquen, and Johan Hagman. 2002. Cross-lingual Document Similarity Calculation Using the Multilingual Thesaurus EUROVOC. *Computational Linguistics and Intelligent Text Processing. Proceedings of the CICLing 2002*, 2276:415—424.

Steinberger, Ralf, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Dániel Varga. 2006. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, volume 9, Genoa, Italy.

Stolcke, Andreas. 2002. SRILM - An Extensible Language Modeling toolkit. In *Intl. Conference on Spoken Language Processing*, Denver, Colorado.

Wikipedia. 2010a. Basque language. [Online; accessed 5-February-2010].

Wikipedia. 2010b. Party of European Socialists | Partido Socialista Europeo | Europako Alderdi Sozialista . [Online; accessed 10-February-2010].

Automatic Detection of Non-deverbal Event Nouns for Quick Lexicon Production

Núria Bel
IULA
Universitat Pompeu Fabra
nuria.bel@upf.edu

Maria Coll
IULA
Universitat Pompeu Fabra
maria.coll02@campus.upf.edu

Gabriela Resnik
Universidad Nacional de General Sarmiento and Universitat Pompeu Fabra
gresnik@ungs.edu.ar

Abstract

In this work we present the results of experimental work on the development of lexical class-based lexica by automatic means. Our purpose is to assess the use of linguistic lexical-class based information as a feature selection methodology for the use of classifiers in quick lexical development. The results show that the approach can help reduce the human effort required in the development of language resources significantly.

1 Introduction

Although language independent, many linguistic technologies are inherently tied to the availability of particular language data (i.e. Language Resources, LR). The nature of these data is very much dependent on particular technologies and the applications where are used. Currently, most systems are using LR collected by hand that still do not cover all languages, or all possible application domains, or all possible information required by the many applications that are being proposed. Methods for the automatic and quick development of new LR have to be developed in order to guarantee a supply of the required data. Esuli and Sebastiani (2006) did a classification experiment for creating lexica for opinion mining, for instance, and the importance of lexical information for event extraction in Biomedical texts has been addressed in Fillmore et al. (2006). One way of producing such resources is to classify words into lexical classes via methods based on their morphosyntactic contexts of occurrence.

In the next three sections we report on an experiment on cue-based lexical classification for non-deverbal event nouns, that is, nouns such as 'party' or 'conflict', which refer to an event but cannot be identified by their morphology, as is the case with deverbal nouns such as 'construction'. The purpose of this experiment was, as already stated, to investigate methods for the rapid generation of an event nouns lexicon for two different languages, using a reduced quantity of available texts. Assuming that linguistic information can be provided by occurrence distribution, as is usually done in linguistic theory to motivate lexical classes (e.g. Grimshaw, 1990), cue information has been gathered from texts and used to train and test a Decision Tree-based classifier. We experimented with two different languages to test the potential coverage of the proposed technique in terms of its adaptation to different languages, and also used different types of corpora to test its adaptability to different domains and sizes.

2 Some properties of Non-Deverbal Event Nouns in Spanish and English.

We based our experiment on the work by Resnik (2004) who proposes a specific lexical class for Spanish event nouns like *accidente* ('accident') or *guerra* ('war') which cannot be identified by suffixes such as '-ción' ('-tion') or 'miento' ('-ment'), i.e. the morphological marks of deverbal derivation. Her proposal of creating a new class is motivated by the syntactic behaviour of these non-deverbal event nouns that differ significantly both from deverbal nominalizations and from non event nouns. This proposal differs significantly from work such as Grimshaw (1990).

In Grimshaw (1990) a significant difference is shown to exist between process and result nominals, evident in certain ambiguous nouns such as *building*, which can have a process reading –

in a sentence like *The building of the access road took three* weeks (= 'process of building')– and a non-eventive or result reading –in a sentence like *The building collapsed* (= 'edifice'). These two types of nominals differ in many lexico-syntactic properties, such as the obligatory/optional internal argument realization, the manner of external argument realization, the determiner selection and their ability to control infinitival clauses. Simple event nouns such as *trip* share several syntactic properties with result nominals, although their lexical meaning is indeed similar to that of the process or complex event nouns. The main difference is the fact that result nominals and simple event nouns, contrary to complex event nominals, are not verb-like in the way they combine with their satellites (Grimshaw 1990). The similarity between result nominals and simple event nouns is accepted in Picallo's (1991, 1999) analysis of Catalan and Spanish nominalizations and in Alexiadou's (2001) work on nominalizations in Greek, English, Hebrew and other languages.

Although the similarities between non-deverbal event nouns like *accidente* and result nominals are undeniable, some evidence (Resnik, 2004 and 2009) has been found that non-deverbal event nouns cannot be assimilated to either result nominals or simple non event nouns like *tren* ('train'), in spite of their shared properties. In the next sections, we briefly present evidence that non-deverbal event nouns are a separate lexical class and that this evidence can be used for identifying the members of this class automatically, both in Spanish and in English. Our hypothesis is that whenever there is a lexical class motivated by a particular distributional behaviour, a learner can be trained to identify the members of this class. However, there are two main problems to lexical classification: noise and silence, as we will see in section 4.

Resnik (2004) shows that non-deverbal event nouns occur in a unique combination of syntactic patterns: they are basically similar to result nouns (and simple non event nouns) regarding the realization of argument structure, yet they pattern along process nominals regarding event structure, given that they accept the same range of aspectual adjuncts and quantifiers as these nouns and are selected as subjects by the same 'aspectual' verbs (*empezar*, 'to start'; *durar*, 'to last', etc.) (cf. section 3.2). As to other nominal properties, such as the mass/count distinction, the contexts show that non-deverbal event nouns are not quite like either of the two kinds of nominalizations, and they behave like simple non event nouns. The table below summarizes the lexico-syntactic properties of the different nouns described by Grimshaw (1990) with the addition of Resnik's proposed new one.

	NDV E N (*war*)	PR-N (*construction* = event)	RES-N (*construction* = result. obj.)	NEN (map)
Obligatory internal argument	no	yes	no	No
External argument realization	genitive DP	PP_by	genitive DP	genitive DP
Subject of aspectual verbs (*begin, last..*)	yes	yes	no	no
Aspectual quantifier (*a period of*)	yes	yes	no	no
Complement of *during, …*	yes	yes	no	no
Count/mass (determiners, plural forms)	mass/count	mass	count	mass/ count

Table 1. Lexico-syntactic properties of English Non-Deverbal Event Nouns (NDV E N), Process Nouns (PR-N) and Result Nouns (RES-N) and Non Event Nouns (NEN).

3 Automatic Detection of Non-deverbal Event Nouns

We have referred to the singularities of non-deverbal event nouns as a lexical class in contrast with other event and non-event nouns. In our experiment, we have extracted the characteristics of the contexts where we hypothesize that members of this class occur and we have used them as variables to train an automatic learner that can rely on these features to automatically classify words into those which are indeed non-deverbal event nouns and those which are not. Because deverbal result nouns are easily identifiable by the nominal suffix they bear (for instance, '-tion' for English and '-ción' for Spanish), our experiment has been centered in separating non-deverbal event nouns like *guerra/war* from non event nouns like *tren/train*.

Some work related to our experiments can be found in the literature dealing with the identification of new events for broadcast news and semantic annotation of texts, which are two possible applications of automatic event detection (Allan et al. 1998 and Saurí et al. 2005, respectively, for example). For these systems, however, it would be difficult to find non-deverbal event nouns because of the absence of morphological suffixes, and therefore they could benefit from our learner.

3.1 Cue-based Lexical Information Acquisition

According to the linguistic tradition, words that can be inserted in the same contexts can be said to belong to the same class. Thus, lexical classes are linguistic generalizations drawn from the characteristics of the contexts where a number of words tend to appear. Consequently, one of the approaches to lexical acquisition proposes to classify words taking as input characteristics of the contexts where words of the same class occur. The idea behind this is that differences in the distribution of the contexts will separate words in different classes, e.g. the class of transitive verbs will show up in passive constructions, while the intransitive verbs will not. Thus, the whole set of occurrences (tokens) of a word are taken as cues for defining its class (the class of the type), either because the word is observed in a number of particular contexts or because it is not. Selected references for this approach are: Brent, 1993; Merlo and Stevenson, 2001; Baldwin and Bond, 2003; Baldwin, 2005; Joanis and Stevenson, 2003; Joanis et al. 2007.

Different supervised Machine Learning (ML) techniques have been applied to cue-based lexical acquisition. A learner is supplied with classified examples of words represented by numerical information about matched and not matched cues. The final exercise is to confirm that the data characterized by the linguistically motivated cues support indeed the division into the proposed classes. This was the approach taken by Merlo and Stevenson (2001), who worked with a Decision Tree and selected linguistic cues to classify English verbs into three classes: *unaccusative*, *unergative* and *object-drop*. Animacy of the subject, for instance, is a significant cue for the class of object dropping verbs, in contrast with verbs in *unergative* and *unaccusative* classes. Baldwin and Bond (2003) used a number of linguistic cues (i.e. co-occurence with particular determiners, number, etc.) to learn the countability of English nouns. Bel et al. (2007) proposed a number of cues for classifying nouns into different types according to a lexical typology. The need for using more general cues has also been pointed out, such as the part of speech tags of neighboring words (Baldwin, 2005), or general linguistic information as in Joanis et al. (2007), who used the frequency of filled syntactic positions or slots, tense and voice features, etc., to describe the whole system of English verbal classes.

3.2 Cues for the Detection of Non-deverbal Event Nouns in Spanish

As we have seen in section 2, non-deverbal event nouns can be identified by their occurrence in particular syntactic and lexical contexts of co-occurrence.We have used 11 cues for separating non-deverbal event nouns from non event nouns in Spanish. These cues are the following:

Cues 1-3. Nouns occurring in PPs headed by prepositions such as *durante* ('during'), *hasta el final de* ('until the end of'), *desde el principio de* ('from the beginning of'), and similar expressions are considered to be eventive. Thus, occurrence after one of such expressions will be indicative of an event noun.

Cues 4-8. Nouns occurring as external or internal arguments of verbs such as *ocurrir* ('occur'), *producir* ('produce' or 'occur', in the case of ergative variant *producirse*), *celebrar* ('celebrate'), and others with similar meanings, are also events. Note that we identify as 'external arguments' the nouns occurring immediately after the verb in particular constructions, as our *pos*- tagged text does not contain information about subjects (see below). In many cases it is the internal argument occurring in these contexts. These verbs tend to appear in 'presentative' constructions such as *Se produjo un accidente* ('An accident occurred'), with the pronoun *se* signalling the lack of external argument. Verbs like *ocurrir* appear in participial absolute constructions or with participial adjectives, which means they are unaccusatives.

Cue 9. The presence of temporal quantifying expressions such as *dos semanas de* ('two weeks of') or similar would indicate the eventive character of a noun occurring with it, as mentioned in section 2.

Cue 10. Non-deverbal event nouns will not be in Prepositional Phrases headed by locative prepositions such as *encima de* ('on top of') or *debajo de* ('under'). These cues are used as negative evidence for non-event deverbal nouns.

Cue 11. Non-deverbal event nouns do have an external argument that can also be realized as an adjective. The alternation of DP arguments with adjectives was then a good cue for detecting non-deverbal events, even when some other nouns may appear in this context as well. For instance: *fiesta nacional* ('national party') vs. *mapa nacional* ('national map').

3.3 Cues for the Detection of Non-Deverbal Event Nouns in English

As for Spanish, cues for English were meant to separate the newly proposed class of non-deverbal event nouns from non-event nouns if such a class exists as well.

Cues 1-3. Process nominals and non-deverbal event nouns can be identified by appearing as complements of aspectual PPs headed by prepositions like *during*, *after* and *before*, and complex prepositions such as *at the end of* and *at the beginning of*.

Cues 4 and 5. Non-deverbal nouns may occur as external or internal arguments of aspectual as well as occurrence verbs such as *initiate*, *take place*, *happen*, *begin*, and *occur*. Those arguments are identified either as subjects of active or passive sentences, depending on the verb, i.e. *the therapy was initiated* and *the conflict took place*.

Cue 6. Likewise, nouns occurring in expressions such as *frequency of*, *occurrence of* and *period of* would probably be event nouns, i.e. *the frequency of droughts*.

Cue 7 and 8. Event nouns may as well appear as objects of aspectual and time-related verbs, such as in *have begun a campaign* or *have carried out a campaign*.

Cues 10 and 11. They are intended to register event nouns whose external argument, although optional, is realized as a genitive complement, e.g. *enzyme's loss*, even though this cue is shared with other types of nouns. Following the characterization suggested for Spanish, we also tried external arguments realized as adjectives in cue 11, as in *Napoleonic war*, but we found empirical evidence that it is not useful.

Cues 12-16. Finally, as in the experiment for Spanish, we have also included evidence that is more common for non-event nouns, that is, we have used negative evidence to tackle the problem of sparse data or silence discussed in the next section. It is considered a negative cue for a noun to be preceded by an indefinite determiner, to be in a PP headed by a locative preposition, and to be followed by the prepositions *by* or *of*, as a PP headed by one these prepositions could be an external argument and, as it has been noted above, the external argument of event nouns tends to be realized as a genitive complement (as in *John's trip/party*).

In the selection of these cues, we have concentrated on those that separate the class of non-deverbal event nouns from the class formed by simple non event nouns like *train*, where no particular deverbal suffix can assist their detection. If it is the case that these are really cues for detecting non-deverbal event nouns, the learner should confirm it by classifying non-deverbal event nouns correctly, separating them from other types of nouns.

4 Experiment and results

For our experiments we have used Regular Expressions to implement the patterns just mentioned, which look for the intended cues in a part-of-speech tagged corpus. We have used a corpus of 21M tokens from two Spanish newspapers (*El País* and *La Vanguardia*), and an English technical corpus made of texts dealing with varying subject matter (Economy, Medicine, Computer science and Environmental issues), of about 3.2M tokens. Both Spanish and English corpora are part of the Technical Corpus of IULA at the UPF (CT-IULA, Cabré et al. 2006). The positive or negative results of the n-pattern checking in all the occurrences of a word are stored in an n-dimension vector. Thus, a single vector summarizes all the occurrences of a word (the type) by encoding how many times each cue has been observed. Zero values, i.e. no matching, are also registered.

We used a Decision Tree (DT) classifier in the Weka (Witten and Frank, 2005) implementation of pruned C4.5 decision tree (Quinlan, 1993). The DT performs a general to specific search in a feature space, selecting the most informative attributes for a tree structure as the search proceeds. The goal is to select the minimal set of attributes that efficiently partitions the feature space into classes of observations and assemble them into a tree. During the experiment, we tuned the list of cues actually used in the classification task, because some of them turned out to be useless, as they did not show up even once in the corpus. This was especially true for the English corpus with cues 5, 11 and 12. Note that the English corpus is only 3.2 million words.

In the experiment we used a 10-fold cross-validation testing using manually annotated gold-standard files made of 99 non-event and 100 non-deverbal event nouns for Spanish and 93 non event and 74 non-deverbal event nouns for English[1]. In this first experiment, we decided to use mostly non-deverbal non event nouns such as *map*, because detecting result nouns like *construction* is easy enough, due to the deverbal suffix. However, for the English experiment, and because of the scarcity of non-deverbal nouns occurrences, we had to randomly select some deverbals that were not recognized by the suffix.

The results of our experiment gave a total accuracy of 80% for Spanish and 79.6% for English, which leads to think that corpus size is not a determinant factor and that this method can be used for addressing different languages, provided a good characterization of the lexical class in terms of particular occurrence distributions is achieved. Yet, although the accuracy of both English and Spanish test sets is similar, we will see later on that the size of the corpus does indeed affect the results.

An analysis of the errors shows that they can be classified in two groups: errors due to noise, and errors due to silence.

(i) Noise. In his seminal work, Brent (1993) already pointed out that “the cues occur in contexts that were not aimed at”. Noise can be due to errors in processing the text, because we had only used low-level analysis tools. For instance, in “during the first world war” our RE cannot detect that “world” is not the head of the Noun Phrase. Brent’s hypothesis, followed by most authors afterwards, is that noise can be eliminated by statistical methods because of its low frequency. However, the fact is that in our test set significant information is as sparse as noise, and the DT cannot correctly handle this. In our data sets, most of the false positives are due to noise.

(ii) Silence. Some nouns appear only once or twice in the corpus and do not show up in any of the sought contexts (for instance, *terremoto*, ‘earthquake’, in Spanish press). Moreover, this is independent of the size of the corpus, because the Zipfian distribution of tokens allows us to predict that there will always be low-frequency nouns. Low frequency words produce non informative vectors, with only zero-valued cues, and our classifier tends to classify non-informative vectors as non-event nouns, because most of the cues have been issued to identify event nouns. This was the main reason to introduce negative contexts as well as positive ones, as we mentioned in section 3.

However, these systematic sources of error can be taken as an advantage when assessing the usability of the resulting resources. Having about 80% of accuracy would not be enough to ensure the proper functioning of the application in which the resource is going to be used. So, in order to gain precision, we decided to separate the set of words that could be safely taken as correctly classified. Thus, we had used the confidence, i.e. probability of the classification de-

[1] **Positive:** accident, assembly, audience, battle, boycott, campaign, catastrophe, ceremony, cold, collapse, conference, conflict, course, crime, crisis, cycle, cyclone, change, choice, decline, disease, disaster, drought, earthquake, epidemic, event, excursion, fair, famine, feast, festival, fever, fight, fire, flight, flood, growth, holiday, hurricane, impact, incident, increase, injury, interview, journey, lecture, loss, meal, measurement, meiosis, marriage, mitosis, monsoon, period, process, program, quake, response, seminar, snowstorm, speech, storm, strike, struggle, summit, symposium, therapy, tour, treaty, trial, trip, vacation, war. **Negative:** agency, airport, animal, architecture, bag, battery, bird, bridge, bus, canal, circle, city, climate, community, company, computer, constitution, country, creature, customer, chain, chair, channel, characteristic, child, defence, director, drug, economy, ecosystem, energy, face, family, firm, folder, food, grade, grant, group, health, hope, hospital, house, illusion, information, intelligence, internet, island, malaria, mammal, map, market, mountain, nation, nature, ocean, office, organism, pencil, people, perspective, phone, pipe, plan, plant, profile, profit, reserve, river, role, satellite, school, sea, shape, source, space, star, statistics, store, technology, television, temperature, theme, theory, tree, medicine, tube, university, visa, visitor, water, weather, window, world.

cisions to assess which are below a reasonable level of confidence.

In the Spanish test set, for instance, precision of the positive classification, i.e. the percentage of words correctly classified as event nouns, raises from 0.82 to 0.95 when only instances of classification with a confidence of more than 0.8 are selected. In the figure below, we can see the precision curve for the Spanish test set.

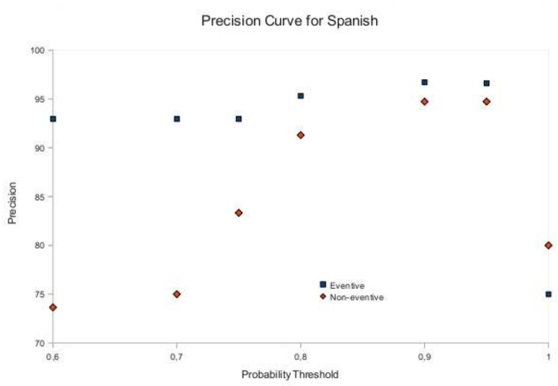


Figure 1: Precision curve for the Spanish test set.

In general, precision is higher when confidence is higher, except for complete confidence, 1, as we will explain later with the English case. This general behavior could be interpreted as a guarantee that there is a significant number of classified nouns (87 out of 199 for the Spanish test set with a threshold of 0.8 confidence) that need not to be manually reviewed, i.e. a 43% of the automatically acquired lexica can safely be considered correct. From figure 1, we can also see that the classifier is consistently identifying the class of non-deverbal event nouns even with a lower threshold. However, the resulting non-event noun set contains a significant number of errors. From the point of view of the usability, we could also say that only those words that are classified as non-event nouns must be revised.

Figure 2 for English test set shows a different behavior, which can only be justified because of the difference in corpus size. A small corpus increases the significance of silence errors. Fewer examples give less information to the classifier, which still makes the right decisions but with less confidence in general. However, for the extreme cases, for instance the case of 7 word vectors with only zero-values, the confidence is very high, that is 1, but the decisions are wrong. These cases of mostly zero values are wrongly considered to be non-events. This is the reason for the low precision of very confident decisions in English, i.e. sparse data and its consequence, silence.

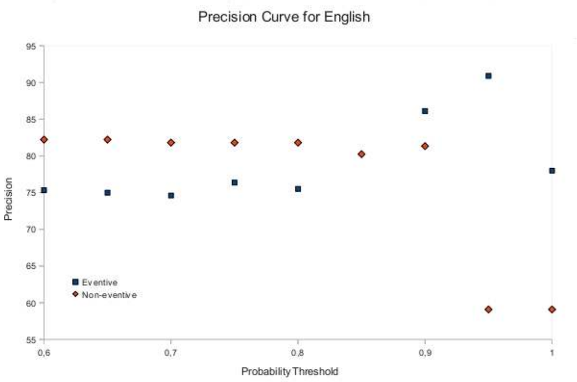


Figure 2: Precision curve for the English test set.

5 Conclusions

In this paper we have proposed the use of lexical classification methods based on differences in the distributional behavior of word classes for the quick production of lexica containing the information required by particular applications. We have dealt with non-deverbal event nouns, which cannot be easily recognized by any suffixes, and we have carried out a classification experiment, which consisted in training a DT with the information used in the linguistic literature to justify the existence of this class. The results of the classifier, close to 80% accuracy in two different languages and with different size and types of source corpora, show the validity of this very simple approach, which can be decisive in the production of lexica with the knowledge required by different technologies and applications in a time-efficient way. From the point of view of usability, this approach can be said to reduce the amount of work in more than a 40%.

Acknowledgements

We want to thank Muntsa Padró for her valuable contribution. This work was partially supported by the PANACEA project (EU-7FP-ITC-248064).

References

Alexiadou, A. (2001). Functional Structure in Nominals: Nominalization and Ergativity. Amsterdam/Philadelphia: John Benjamins Publishing Company.

Allan, J.; Papka, R.; Lavrenko, V. (1998). On-line New Event Detection and Tracking. SIGIR98, Melbourne, Australia.

Baldwin, T. and F. Bond. 2003. "Learning the Countability of English Nouns from Corpus Data". *Proceedings of the 41st. Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan.

Baldwin, T. 2005. General-Purpose Lexical Acquisition: Procedures, Questions and Results, In *Proceedings of the Pacific Association for Computational Linguistics 2005*, Tokyo, Japan

Bel, N.; Espeja, S.; Marimon, M. 2007. Automatic Acquisition of Grammatical Types for Nouns. In *H LT 2007: The Conference of the NAACL*. Companion Volume, Short Papers. Rochester, USA.

Brent, M. R. 1993, 'From grammar to lexicon: unsupervised learning of lexical syntax'. *Computational Linguistics* 19: 243-262.

Cabré, M. T.; Bach, C.; Vivaldi, J. 2006. *10 anys del Corpus de l'IULA*. Barcelona: Institut Universitari de Lingüística Aplicada. Universitat Pompeu Fabra

Esuli, A. and Sebastiani, F.. 2006. Determining term subjectivity and term orientation for opinion mining. In Proceedings of EACL-06, 11th Conference of the European Chapter of the Association for Computational Linguistics, Trento, IT.

Fillmore, Charles J.Srini Narayanan, and Collin F. Baker. 2006. What Can Linguistics Contribute to Event Extraction? *Proceedings of the 2006 AAAI Workshop on Event Extraction and Synthesis, page 18--23.*

Grimshaw, J. (1990). Argument Structure. Cambridge: The MIT Press.

Joanis, E; Stevenson, S; and James, D. 2007. A General Feature Space for Automatic Verb Classification. *Natural Language Engineering, 14*.

Korhonen, A. 2002. 'Subcategorization acquisition'. As Technical Report UCAM-CL-TR-530, University of Cambridge, UK.

Merlo P. and Stevenson S. 2001. Automatic Verb Classification based on Statistical Distribution of Argument Structure, *Computational Linguistics*, 27:3.

Picallo, M. C. (1999). "La estructura del sintagma nominal: Las nominalizaciones y otros sustantivos con complementos argumentales", en Bosque, I. & V. Demonte (eds.) *Gramática descriptiva de la lengua española*. Madrid: Real Academia Española / Espasa Calpe. Vol. 1, Cap. 6, 363-394.

Quinlan, R.J. 1993. C4.5: Programs for Machine Learning. Series in Machine Learning. Morgan Kaufman, San Mateo, CA.

Resnik, G. (2004). *Los nombres eventivos no deverbales en español*. Proyecto de Tesis. Institut Universitari de Lingüística Aplicada, Universitat Pompeu Fabra.

Resnik, G. (2009) "La determinación de la eventividad nominal en español". En G. Ciapuscio (ed.) *De la palabra al texto: estudios lingüísticos del español*. Buenos Aires: Eudeba.

Saurí, R.; R. Knippen, M. Verhagen and J. Pustejovsky. 2005. Evita: A Robust Event Recognizer for QA Systems. *Proceedings of HLT/EMNLP 2005:* 700-70

Witten, I. H. and Frank E. 2005. Data Mining: Practical machine learning tools and techniques. 2nd Edition, Morgan Kaufmann, San Francisco.

Fast and Accurate Arc Filtering for Dependency Parsing

Shane Bergsma
Department of Computing Science
University of Alberta
sbergsma@ualberta.ca

Colin Cherry
Institute for Information Technology
National Research Council Canada
colin.cherry@nrc-cnrc.gc.ca

Abstract

We propose a series of learned arc filters to speed up graph-based dependency parsing. A cascade of filters identify implausible head-modifier pairs, with time complexity that is first linear, and then quadratic in the length of the sentence. The linear filters reliably predict, in context, words that are roots or leaves of dependency trees, and words that are likely to have heads on their left or right. We use this information to quickly prune arcs from the dependency graph. More than 78% of total arcs are pruned while retaining 99.5% of the true dependencies. These filters improve the speed of two state-of-the-art dependency parsers, with low overhead and negligible loss in accuracy.

1 Introduction

Dependency parsing finds direct syntactic relationships between words by connecting head-modifier pairs into a tree structure. Dependency information is useful for a wealth of natural language processing tasks, including question answering (Wang et al., 2007), semantic parsing (Poon and Domingos, 2009), and machine translation (Galley and Manning, 2009).

We propose and test a series of **arc filters** for graph-based dependency parsers, which rule out potential head-modifier pairs before parsing begins. In doing so, we hope to eliminate implausible links early, saving the costs associated with them, and speeding up parsing. In addition to the scaling benefits that come with faster processing, we hope to enable richer features for parsing by constraining the set of arcs that need to be considered. This could allow extremely large feature sets (Koo et al., 2008), or the look-up of expensive corpus-based features such as word-pair mutual information (Wang et al., 2006). These filters could also facilitate expensive learning algorithms, such as semi-supervised approaches (Wang et al., 2008).

We propose three levels of filtering, which are applied in a sequence of increasing complexity:

Rules: A simple set of machine-learned rules based only on parts-of-speech. They prune over 25% of potential arcs with almost no loss in coverage. Rules save on the wasted effort for assessing implausible arcs such as DT → DT.

Linear: A series of classifiers that tag words according to their possible roles in the dependency tree. By treating each word independently and ensuring constant-time feature extraction, they operate in linear time. We view these as a dependency-parsing analogue to the span-pruning proposed by Roark and Hollingshead (2008). Our fast linear filters prune 54.2% of potential arcs while recovering 99.7% of true pairs.

Quadratic: A final stage that looks at pairs of words to prune unlikely arcs from the dependency tree. By employing a light-weight feature set, this high-precision filter can enable more expensive processing on the remaining plausible dependencies.

Collectively, we show that more than 78% of total arcs can be pruned while retaining 99.5% of the true dependencies. We test the impact of these filters at both train and test time, using two state-of-the-art discriminative parsers, demonstrating speed-ups of between 1.9 and 5.6, with little impact on parsing accuracy.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 53–61,
Beijing, August 2010

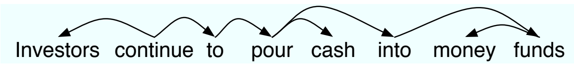


Figure 1: An example dependency parse.

2 Dependency Parsing

A dependency tree represents the syntactic structure of a sentence as a directed graph (Figure 1), with a node for each word, and arcs indicating head-modifier pairs (Meĺčuk, 1987). Though dependencies can be extracted from many formalisms, there is a growing interest in predicting dependency trees directly. To that end, there are two dominant approaches: graph-based methods, characterized by arc features in an exhaustive search, and transition-based methods, characterized by operational features in a greedy search (McDonald and Nivre, 2007). We focus on graph-based parsing, as its exhaustive search has the most to gain from our filters.

Graph-based dependency parsing finds the highest-scoring tree according to a scoring function that decomposes under an exhaustive search (McDonald et al., 2005). The most natural decomposition scores individual arcs, represented as head-modifier pairs $[h, m]$. This enables search by either minimum spanning tree (West, 2001) or by Eisner's (1996) projective parser. This paper focuses on the projective case, though our techniques transfer to spanning tree parsing. With a linear scoring function, the parser solves:

$$\mathrm{parse}(s) = \mathrm{argmax}_{t \in s} \sum_{[h,m] \in t} \bar{w} \cdot \bar{f}(h, m, s)$$

The weights $\bar{w}$ are typically learned using an online method, such as an averaged perceptron (Collins, 2002) or MIRA (Crammer and Singer, 2003). 2nd-order searches, which consider two siblings at a time, are available with no increase in asymptotic complexity (McDonald and Pereira, 2006; Carreras, 2007).

The complexity of graph-based parsing is bounded by two processes: parsing (carrying out the argmax) and arc scoring (calculating $\bar{w} \cdot \bar{f}(h, m, s)$). For a sentence with n words, projective parsing takes $O(n^3)$ time, while the spanning tree algorithm is $O(n^2)$. Both parsers require scores for arcs connecting each possible $[h, m]$ pair in s; therefore, the cost of arc scoring is also $O(n^2)$, and may become $O(n^3)$ if the features include words in s between h and m (Galley and Manning, 2009). Arc scoring also has a significant constant term: the number of features extracted for an $[h, m]$ pair. Our in-house graph-based parser collects on average 62 features for each potential arc, a number larger than the length of most sentences. With the cluster-based features suggested by Koo et al. (2008), this could easily grow by a factor of 3 or 4.

The high cost of arc scoring, coupled with the parsing stage's low grammar constant, means that graph-based parsers spend much of their time scoring potential arcs. Johnson (2007) reports that when arc scores have been precomputed, the dynamic programming component of his 1st-order parser can process an amazing 3,580 sentences per second.[1] Beyond reducing the number of features, the easiest way to reduce the computational burden of arc scoring is to score only plausible arcs.

3 Related Work

3.1 Vine Parsing

Filtering dependency arcs has been explored primarily in the form of vine parsing (Eisner and Smith, 2005; Dreyer et al., 2006). Vine parsing establishes that, since most dependencies are short, one can parse quickly by placing a hard constraint on arc length. As this coarse filter quickly degrades the best achievable performance, Eisner and Smith (2005) also consider conditioning the constraint on the part-of-speech (PoS) tags being linked and the direction of the arc, resulting in a separate threshold for each $[\mathrm{tag}(h), \mathrm{tag}(m), \mathrm{dir}(h, m)]$ triple. They sketch an algorithm where the thresholded length for each triple starts at the highest value seen in the training data. Thresholds are then decreased in a greedy fashion, with each step producing the smallest possible reduction in reachable training arcs. We employ this algorithm as a baseline in our experiments. To our knowledge, vine parsing

[1] To calibrate this speed, consider that the publicly available 1st-order MST parser processes 16 sentences per second on modern hardware. This includes I/O costs in addition to the costs of arc scoring and parsing.

has not previously been tested with a state-of-the-art, discriminative dependency parser.

3.2 CFG Cell Classification

Roark and Hollingshead (2008) speed up another exhaustive parsing algorithm, the CKY parser for CFGs, by classifying each word in the sentence according to whether it can open (or close) a multi-word constituent. With a high-precision tagger that errs on the side of permitting constituents, they show a significant improvement in speed with no reduction in accuracy.

It is difficult to port their idea directly to dependency parsing without committing to a particular search algorithm,[2] and thereby sacrificing some of the graph-based formalism's modularity. However, some of our linear filters (see Section 4.3) were inspired by their constraints.

3.3 Coarse-to-fine Parsing

Another common method employed to speed up exhaustive parsers is a coarse-to-fine approach, where a cheap, coarse model prunes the search space for later, more expensive models (Charniak et al., 2006; Petrov and Klein, 2007). This approach assumes a common forest or chart representation, shared by all granularities, where one can efficiently track the pruning decisions of the coarse models. One could imagine applying such a solution to dependency parsing, but the exact implementation of the coarse pass would vary according to the choice in search algorithm. Our filters are much more modular: they apply to both 1st-order spanning tree parsing and 2nd-order projective parsing, with no modification.

Carreras et al. (2008) use coarse-to-fine pruning with dependency parsing, but in that case, a graph-based dependency parser provides the coarse pass, with the fine pass being a far-more-expensive tree-adjoining grammar. Our filters could become a 0th pass, further increasing the efficiency of their approach.

4 Arc Filters

We propose arc filtering as a preprocessing step for dependency parsing. An arc filter removes implausible head-modifier arcs from the complete dependency graph (which initially includes all head-modifier arcs). We use three stages of filters that operate in sequence on progressively sparser graphs: 1) rule-based, 2) linear: a single pass through the n nodes in a sentence ($O(n)$ complexity), and 3) quadratic: a scoring of all remaining arcs ($O(n^2)$). The less intensive filters are used first, saving time by leaving fewer arcs to be processed by the more intensive systems.

Implementations of our rule-based, linear, and quadratic filters are publicly available at:
`http://code.google.com/p/arcfilter/`

4.1 Filter Framework

Our filters assume the input sentences have been PoS-tagged. We also add an artificial root node to each sentence to be the head of the tree's root. Initially, this node is a potential head for all words in the sentence.

Each filter is a supervised classifier. For example, the quadratic filter directly classifies whether a proposed head-modifier pair is *not* a link in the dependency tree. Training data is created from annotated trees. All possible arcs are extracted for each training sentence, and those that are present in the annotated tree are labeled as class -1, while those not present are $+1$. A similar process generates training examples for the other filters. Since our goal is to only filter very implausible arcs, we bias the classifier to high precision, increasing the cost for misclassifying a true arc during learning.[3]

Class-specific costs are command-line parameters for many learning packages. One can interpret the learning objective as minimizing regularized, weighted loss:

$$\min_{\bar{w}} \frac{1}{2}||\bar{w}||^2 + C_1 \sum_{i:y_i=1} l(\bar{w}, y_i, \bar{x}_i) + C_2 \sum_{i:y_i=-1} l(\bar{w}, y_i, \bar{x}_i) \qquad (1)$$

where $l()$ is the learning method's loss function, $\bar{x}_i$ and y_i are the features and label for the ith

[2] Johnson's (2007) split-head CFG could implement this idea directly with little effort.

[3] Learning with a cost model is generally preferable to first optimizing error rate and then thresholding the prediction values to select a high-confidence subset (Joachims, 2005), but the latter approach was used successfully for cell classification in Roark and Hollingshead (2008).

not a h	” “ , . ; | CC PRP$ PRP EX -RRB- -LRB-
no $* \leftarrow m$	EX LS POS PRP$
no $m \rightarrow *$	. RP
not a root	, DT

no $h \leftarrow m$	DT←{DT,JJ,NN,NNP,NNS,.}
	CD←CD NN←{DT,NNP}
	NNP←{DT,NN,NNS}

no $m \rightarrow h$	{DT,IN,JJ,NN,NNP}→DT NNP→IN IN→JJ

Table 1: Learned rules for filtering dependency arcs using PoS tags. The rules filter 25% of possible arcs while recovering 99.9% of true links.

training example, $\bar{w}$ is the learned weight vector, and C_1 and C_2 are the class-specific costs. High precision is obtained when $C_2 >> C_1$. For an SVM, $l(\bar{w}, y_i, \bar{x}_i)$ is the standard hinge loss.

We solve the SVM objective using LIBLINEAR (Fan et al., 2008). In our experiments, each filter is a linear SVM with the typical L1 loss and L2 regularization.[4] We search for the best combination of C_1 and C_2 using a grid search on development data. At test time, an arc is filtered if $\bar{w} \cdot \bar{x} > 0$.

4.2 Rule-Based Filtering

Our rule-based filters seek to instantly remove those arcs that are trivially implausible on the basis of their head and modifier PoS tags. We first extract labeled examples from gold-standard trees for whenever a) a word is not a head, b) a word does not have a head on the left (resp. right), and c) a pair of words is not linked. We then trained high-precision SVM classifiers. The only features in $\bar{x}$ are the PoS tag(s) of the head and/or modifier. The learned feature weights identify the tags and tag-pairs to be filtered. For example, if a tag has a positive weight in the not-a-head classifier, all arcs having that node as head are filtered.

The classier selects a small number of high-precision rules, shown in Table 1. Note that the rules tend to use common tags with well-defined roles. By focusing on weighted loss as opposed to arc frequency, the classifier discovers structural zeros (Mohri and Roark, 2006), events which could have been observed, but were not. We consider this an improvement over the frequency-based length thresholds employed previously in tag-specific vine parsing.

4.3 Linear-Time Filtering

In the linear filtering stage, we filter arcs on the basis of single nodes and their contexts, passing through the sentences in linear time. For each node, eight separate classifiers decide whether:

1. It is *not* a head (i.e., it is a leaf of the tree).
2. Its head is on the left/right.
3. Its head is within 5 nodes on the left/right.
4. Its head is immediately on the left/right.
5. It is the root.

For each of these decisions, we again train high-precision SVMs with $C_2 >> C_1$, and filter directly based on the classifier output.

If a word is not a head, all arcs with the given word as head can be pruned. If a word is deemed to have a head within a certain range on the left or right, then all arcs that do not obey this constraint can be pruned. If a root is found, no other words should link to the artificial root node. Furthermore, in a projective dependency tree, no arc will cross the root, i.e., there will be no arcs where a head and a modifier lie on either side of the root. We can therefore also filter arcs that violate this constraint when parsing projectively.

Søgaard and Kuhn (2009) previously proposed a tagger to further constrain a vine parser. Their tags are a subset of our decisions (items 4 and 5 above), and have not yet been tested in a state-of-the-art system.

Development experiments show that if we could perfectly make decisions 1-5 for each word, we could remove 91.7% of the total arcs or 95% of negative arcs, close to the upper bound.

Features

Unlike rule-based filtering, linear filtering uses a rich set of features (Table 2). Each feature is a

[4] We also tried L1-regularized filters. L1 encourages most features to have zero weight, leading to more compact and hence faster models. We found the L1 filters to prune fewer arcs at a given coverage level, providing less speed-up at parsing time. Both L1 and L2 models are available in our publicly available implementation.

PoS-tag features	Other features
tag_i	word_i
tag_i, tag_{i-1}	word_{i+1}
tag_i, tag_{i+1}	word_{i-1}
tag_{i-1}, tag_{i+1}	shape_i
tag_{i-2}, tag_{i-1}	prefix_i
tag_{i+1}, tag_{i+2}	suffix_i
tag_j, Left, $_{j=i-5...i-1}$	i
tag_j, Right, $_{j=i+1...i+5}$	i, n
tag_j, (i-j), $_{j=i-5...i-1}$	n - i
tag_j, (i-j), $_{j=i+1...i+5}$	

Table 2: Linear filter features for a node at position i in a sentence of length n. Each feature is also conjoined (unless redundant) with word_i, tag_i, shape_i, prefix_i, and suffix_i (both 4 letters). The shape is the word normalized using the regular expressions [A-Z]+ → A and [a-z]+ → a.

Binary features	
sign(h-m)	tags_{hm}
tag_{m-1}, tags_{hm}	tag_{m+1}, tags_{hm}
tag_{h-1}, tags_{hm}	tag_{h+1}, tags_{hm}
sign(h-m), tag_h, word_m	
sign(h-m), word_h, tag_m	
Real features ⇒ values	
sign(h-m) ⇒ h-m	
tag_h, tag_m ⇒ h-m	
tag_k, tags_{hm} ⇒ Count($\text{tag}_k \in \text{tags}_{h...m}$)	
word_k, tags_{hm} ⇒ Count($\text{word}_k \in \text{words}_{h...m}$)	

Table 3: Quadratic filter features for a head at position h and a modifier at position m in a sentence of length n. Here tags_{hm} = (sign(h-m), tag_h, tag_m), while $\text{tags}_{h...m}$ and $\text{words}_{h...m}$ are all the tags (resp. words) between h and m, but within ±5 positions of h or m.

binary indicator feature. To increase the speed of applying eight classifiers, we use the same feature vector for each of the decisions; learning gives eight different weight vectors, one corresponding to each decision function. Feature extraction is constrained to be $O(1)$ for each node, so that overall feature extraction and classification remain a fast $O(n)$ complexity. Feature extraction would be $O(n^2)$ if, for example, we had a feature for *every* tag on the left or right of a node.

Combining linear decisions

We originally optimized the C_1 and C_2 parameter separately for each linear decision function. However, we found we could substantially improve the collective performance of the linear filters by searching for the optimal combination of the component decisions, testing different levels of precision for each component. We selected a few of the best settings for each decision when optimized separately, and then searched for the best combination of these candidates on development data (testing 12960 combinations in all).

4.4 Quadratic-Time Filtering

In the quadratic filtering stage, a single classifier decides whether each head-modifier pair should be filtered. It is trained and applied as described in Section 4.1.

While theoretically of the same complexity as the parser's arc-scoring function ($O(n^2)$), this process can nevertheless save time by employing a compact feature set. We view quadratic filtering as a light preprocessing step, using only a portion of the resources that might be used in the final scoring function.

Features

Quadratic filtering uses both binary *and* real-valued features (Table 3). Real-valued features promote a smaller feature space. For example, one value can encode distance rather than separate features for different distances. We also generalize the "between-tag features" used in McDonald et al. (2005) to be the count of each tag between the head and modifier. The count may be more informative than tag presence alone, particularly for high-precision filters. We follow Galley and Manning (2009) in using only between-tags within a fixed range of the head or modifier, so that the extraction for each pair is $O(1)$ and the overall feature extraction is $O(n^2)$.

Using only a subset of the between-tags as features has been shown to improve speed but impair parser performance (Galley and Manning, 2009). By filtering quickly first, then scoring all remaining arcs with a cubic scoring function in the parser, we hope to get the best of both worlds.

5 Filter Experiments

Data

We extract dependency structures from the Penn Treebank using the Penn2Malt extraction tool,[5] which implements the head rules of Yamada and Matsumoto (2003). Following convention, we divide the Treebank into train (sections 2–21), development (22) and test sets (23). The development and test sets are re-tagged using the Stanford tagger (Toutanova et al., 2003).

Evaluation Metrics

To measure intrinsic filter quality, we define **Reduction** as the proportion of total arcs removed, and **Coverage** as the proportion of true head-modifier arcs retained. Our evaluation asks, for each filter, what Reduction can be obtained at a given Coverage level? We also give **Time**: how long it takes to apply the filters to the test set (excluding initialization).

We compute an **Upper Bound** for Reduction on development data. There are 1.2 million potential dependency links in those sentences, 96.5% of which are not present in a gold standard dependency tree. Therefore, the maximum achievable Reduction is 96.5%.

Systems

We evaluate the following systems:

- **Rules**: the rule-based filter (Section 4.2)
- **Lin.**: the linear-time filters (Section 4.3)
- **Quad.**: the quadratic filter (Section 4.4)

The latter two approaches run on the output of the previous stage. We compare to the two vine parsing approaches described in Section 3.1:

- **Len-Vine** uses a hard limit on arc length.
- **Tag-Vine** (later, **Vine**) learns a maximum length for dependency arcs for every head/modifier tag-combination and order.

5.1 Results

We set each filter's parameters by selecting a Coverage-Reduction tradeoff on development data (Figure 2). The Lin curve is obtained by varying both the C_1/C_2 cost parameters and the combination of components (plotting the best Reduction at each Coverage level). We chose the linear filters with 99.8% Coverage at a 54.2% Reduction. We apply Quad on this output, varying the cost parameters to produce its curve. Aside from Len-Vine, all filters remove a large number of arcs with little drop in Coverage.

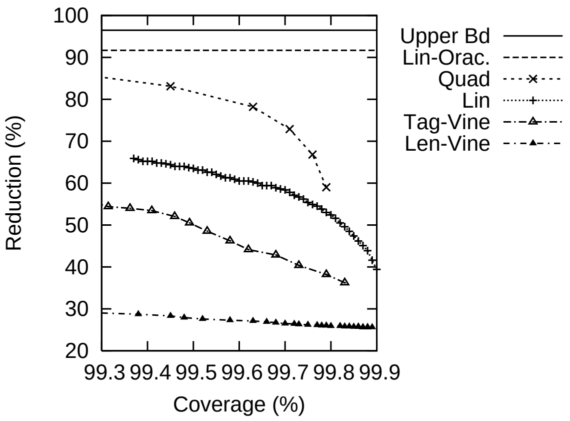


Figure 2: Filtering performance for different filters and cost parameters on development data. Lin-Orac indicates the percentage filtered using perfect decisions by the linear components.

Filter	Coverage	Reduct.	Time (s)
Vine	99.62	44.0	2.9s
Rules	99.86	25.8	1.3s
Lin.	99.73	54.2	7.3s
Quad.	99.50	78.4	16.1s

Table 4: Performance (%) of filters on test data.

After selecting a desired trade-off for each classifier, we move to final filtering experiments on unseen test data (Table 4). The linear filter removes well over half the links but retains an astounding 99.7% of correct arcs. Quad removes 78.4% of arcs at 99.5% Coverage. It thus reduces the number of links to be scored by a dependency parser by a factor of five.

The time for filtering the 2416 test sentences varies from almost instantaneous for Vine and Rules to around 16 seconds for Quad. Speed numbers are highly machine, design, and implemen-

[5] http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html

Decision	Precision	Recall
No-Head	99.9	44.8
Right-∅	99.9	28.7
Left-∅	99.9	39.0
Right-5	99.8	31.5
Left-5	99.9	19.7
Right-1	99.7	6.2
Left-1	99.7	27.3
Root	98.6	25.5

Table 5: Linear Filters: Test-set performance (%) on decisions for components of the combined 54.2 Reduct./99.73 Coverage linear filter.

Type	Coverage	Reduct.	Oracle
All	**99.73**	**54.2**	**91.8**
All\No-Head	99.76	46.4	87.2
All\Left-∅	99.74	53.2	91.4
All\Right-∅	99.75	53.6	90.7
All\Left-5	99.74	53.2	89.7
All\Right-5	99.74	51.6	90.4
All\Left-1	99.75	53.5	90.8
All\Right-1	99.73	53.9	90.6
All\Root	99.76	50.2	90.0

Table 6: Contribution of different linear filters to test set performance (%). Oracle indicates the percentage filtered by perfect decisions.

tation dependent, and thus we have stressed the asymptotic complexity of the filters. However, the timing numbers show that arc filtering can be done quite quickly. Section 6 confirms that these are very reasonable costs in light of the speed-up in overall parsing.

5.2 Linear Filtering Analysis

It is instructive to further analyze the components of the linear filter. Table 5 gives the performance of each classifier on its specific decision. **Precision** is the proportion of positive classifications that are correct. **Recall** is the proportion of positive instances that are classified positively (e.g. the proportion of actual roots that were classified as roots). The decisions correspond to items 1-5 in Section 4.3. For example, *Right-∅* is the decision that a word has *no* head on the right.

Most notably, the optimum *Root* decision has much lower Precision than the others, but this has little effect on its overall accuracy as a filter (Table 6). This is perhaps because the few cases of false positives are still likely to be main verbs or auxiliaries, and thus still still likely to have few links crossing them. Thus many of the filtered links are still correct.

Table 6 provides the performance of the classifier combination when each linear decision is excluded. *No-Head* is the most important component in the oracle and the actual combination.

6 Parsing Experiments

6.1 Set-up

In this section, we investigate the impact of our filters on graph-based dependency parsers. We train each parser unfiltered, and then measure its speed and accuracy once filters have been applied. We use the same training, development and test sets described in Section 5. We evaluate unlabeled dependency parsing using head **accuracy**: the percentage of words (ignoring punctuation) that are assigned the correct head.

The filters bypass feature extraction for each filtered arc, and replace its score with an extremely low negative value. Note that 2$^{\text{nd}}$-order features consider $O(n^3)$ $[h, m_1, m_2]$ triples. These triples are filtered if at least one component arc ($[h, m_1]$ or $[h, m_2]$) is filtered.

In an optimal implementation, we might also have the parser re-use features extracted during filtering when scoring the remaining arcs. We did not do this. Instead, filtering was treated as a preprocessing step, which maximizes the portability of the filters across parsers. We test on two state-of-the art parsers:

MST We modified the publicly-available MST parser (McDonald et al., 2005)[6] to employ our filters before carrying out feature extraction. MST is trained with 5-best MIRA.

DepPercep We also test an in-house dependency parser, which conducts projective first and 2$^{\text{nd}}$-order searches using the split-head CFG described by Johnson (2007), with a weight vector trained using an averaged perceptron (Collins,

[6] `http://sourceforge.net/projects/mstparser/`

		DepPercep-1		DepPercep-2		MST-1		MST-2	
Filter	Cost	Acc.	Time	Acc.	Time	Acc.	Time	Acc.	Time
None	+0	91.8	348	92.5	832	91.2	153	91.9	200
Vine	+3	91.7	192	92.3	407	91.2	99	91.8	139
Rules	+1	91.7	264	92.4	609	91.2	125	91.9	167
Linear	+7	91.7	168	92.4	334	91.2	88	91.8	121
Quad.	+16	91.7	79	92.3	125	91.2	58	91.8	80

Table 7: The effect of filtering on the speed and accuracy on 1$^{\text{st}}$ and 2$^{\text{nd}}$-order dependency parsing.

2002). Its features are a mixture of those described by McDonald et al. (2005), and those used in the Koo et al. (2008) baseline system; we do not use word-cluster features.

DepPercep makes some small improvements to MST's 1$^{\text{st}}$-order feature set. We carefully determined which feature types should have distance appended in addition to direction. Also, inspired by the reported utility of mixing PoS tags and word-clusters (Koo et al., 2008), we created versions of all of the "Between" and "Surrounding Word" features described by McDonald et al. (2005) where we mix tags and words.[7]

DepPercep was developed with quadratic filters in place, which enabled a fast development cycle for feature engineering. As a result, it does not implement many of the optimizations in place in MST, and is relatively slow unfiltered.

6.2 Results

The parsing results are shown in Table 7, where times are given in seconds, and **Cost** indicates the additional cost of filtering. Note that the impact of all filters on accuracy is negligible, with a decrease of at most 0.2%. In general, parsing speed-ups mirror the amount of arc reduction measured in our filter analysis (Section 5.1).

Accounting for filter costs, the benefits of quadratic filtering depend on the parser. The extra benefit of quadratic over linear is substantial for DepPercep, but less so for 1$^{\text{st}}$-order MST.

MST shows more modest speed-ups than DepPercep, but MST is already among the fastest publicly-available data-driven parsers. Under quadratic filtering, MST-2 goes from processing 12 sentences per second to 23 sentences.[8]

DepPercep-2 starts slow, but benefits greatly from filtering. This is because, unlike MST-2, it does not optimize feature extraction by factoring its ten 2$^{\text{nd}}$-order features into two triple ($[h, m_1, m_2]$) and eight sibling ($[m_1, m_2]$) features. This suggests that filtering could have a dramatic effect on a parser that uses more than a few triple features, such as Koo et al. (2008).

7 Conclusion

We have presented a series of arc filters that speed up graph-based dependency parsing. By treating filtering as weighted classification, we learn a cascade of increasingly complex filters from tree-annotated data. Linear-time filters prune 54% of total arcs, while quadratic-time filters prune 78%. Both retain at least 99.5% of true dependencies. By testing two state-of-the-art dependency parsers, we have shown that our filters produce substantial speed improvements in even carefully-optimized parsers, with negligible losses in accuracy. In the future we hope to leverage this reduced search space to explore features derived from large corpora.

References

Carreras, Xavier, Michael Collins, and Terry Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *CoNLL*.

Carreras, Xavier. 2007. Experiments with a higher-order projective dependency parser. In *EMNLP-CoNLL*.

[7] This was enabled by using word features only when the word is among the 800 most frequent in the training set.

[8] This speed accounts for 25 total seconds to apply the rules, linear, and quadratic filters.

Charniak, Eugene, Mark Johnson, Micha Elsner, Joseph Austerweil, David Ellis, Isaac Haxton, Catherine Hill, R. Shrivaths, Jeremy Moore, Michael Pozar, and Theresa Vu. 2006. Multilevel coarse-to-fine PCFG parsing. In *HLT-NAACL*.

Collins, Michael. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

Crammer, Koby and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *JMLR*, 3:951–991.

Dreyer, Markus, David A. Smith, and Noah A. Smith. 2006. Vine parsing and minimum risk reranking for speed and precision. In *CoNLL*.

Eisner, Jason and Noah A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *IWPT*.

Eisner, Jason. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING*.

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874.

Galley, Michel and Christopher D. Manning. 2009. Quadratic-time dependency parsing for machine translation. In *ACL-IJCNLP*.

Joachims, Thorsten. 2005. A support vector method for multivariate performance measures. In *ICML*.

Johnson, Mark. 2007. Transforming projective bilexical dependency grammars into efficiently-parsable CFGs with unfold-fold. In *ACL*.

Koo, Terry, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *ACL-08: HLT*.

McDonald, Ryan and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *EMNLP-CoNLL*.

McDonald, Ryan and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *ACL*.

Mel̓čuk, Igor A. 1987. *Dependency syntax: theory and practice*. State University of New York Press.

Mohri, Mehryar and Brian Roark. 2006. Probabilistic context-free grammar induction based on structural zeros. In *HLT-NAACL*.

Petrov, Slav and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *HLT-NAACL*.

Poon, Hoifung and Pedro Domingos. 2009. Unsupervised semantic parsing. In *EMNLP*.

Roark, Brian and Kristy Hollingshead. 2008. Classifying chart cells for quadratic complexity context-free inference. In *COLING*.

Søgaard, Anders and Jonas Kuhn. 2009. Using a maximum entropy-based tagger to improve a very fast vine parser. In *IWPT*.

Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*.

Wang, Qin Iris, Colin Cherry, Dan Lizotte, and Dale Schuurmans. 2006. Improved large margin dependency parsing via local constraints and Laplacian regularization. In *CoNLL*.

Wang, Mengqiu, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *EMNLP-CoNLL*.

Wang, Qin Iris, Dale Schuurmans, and Dekang Lin. 2008. Semi-supervised convex training for dependency parsing. In *ACL-08: HLT*.

West, D. 2001. *Introduction to Graph Theory*. Prentice Hall, 2nd edition.

Yamada, Hiroyasu and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *IWPT*.

A Hierarchical Classifier Applied to Multi-way Sentiment Detection

Adrian Bickerstaffe and Ingrid Zukerman
Faculty of Information Technology
Monash University
bickerstaffe.adrian@gmail.com,Ingrid.Zukerman@monash.edu

Abstract

This paper considers the problem of document-level multi-way sentiment detection, proposing a hierarchical classifier algorithm that accounts for the inter-class similarity of tagged sentiment-bearing texts. This type of classifier also provides a natural mechanism for reducing the feature space of the problem. Our results show that this approach improves on state-of-the-art predictive performance for movie reviews with three-star and four-star ratings, while simultaneously reducing training times and memory requirements.

1 Introduction

A key problem in sentiment detection is to determine the polarity of sentiment in text. Much of the work on this problem has considered binary sentiment polarity (positive or negative) at granularity levels ranging from sentences (Yu and Hatzivassiloglou, 2003; Mao and Lebanon, 2006; McDonald et al., 2007) to documents (Wilson et al., 2005; Allison, 2008).

This paper considers the more general problem of multi-way sentiment classification for discrete, ordinal rating scales, focusing on the document level, i.e., the problem of predicting the "star" rating associated with a review. This is a supervised learning task involving textual reviews that have been tagged with a rating. Ultimately, the goal is to use classifiers which have been trained on tagged datasets to predict the ratings of untagged reviews.

Typical approaches to the rating scale problem include standard k-way classifiers, e.g., (Pang and Lee, 2005). However, these methods do not explicitly account for sample similarities, e.g., the samples with a "four star" rating being more similar to "three star" samples than to "one star" samples. Consequently, these methods generally do not perform well, while methods which incorporate sample similarity information achieve improved performance (Pang and Lee, 2005).

Sample similarity in the multi-way sentiment detection setting has previously been considered by using Support Vector Machines (SVMs) in conjunction with a metric labeling meta-algorithm (Pang and Lee, 2005); by taking a semi-supervised graph-based learning approach (Goldberg and Zhu, 2006); and by using "optimal stacks" of SVMs (Koppel and Schler, 2006). However, each of these methods have shortcomings (Section 2). Additionally, during the learning process, all approaches employ a set of word/punctuation features collected across *all* rating categories. Hence, the number of features may be very large compared to the number of training samples, which can lead to the model overfitting the data.

The main contribution of this paper is the use of hierarchical classifier trees which combine standard binary classifiers to perform multi-way classification (another approach to reduce multi-class classification to binary classifications is described in (Beygelzimer et al., 2009)). The hierarchical classifier accounts for inter-class similarity by

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 62–70,
Beijing, August 2010

means of tree structures which are obtained using inter-class similarity measures in conjunction with a shortest-spanning algorithm. The tree structures reduce training times since they require only $k-1$ nodes for a k-rating problem. Training times are further reduced by the fact that classifier nodes lower in the tree consider fewer rating classes than those higher up, thereby naturally reducing the number of training samples relevant to lower-level nodes. Additionally, the tree structures offer a means to safely cull irrelevant features at non-root nodes of the tree, thus reducing the dimensionality of the training data for these nodes without loss of information. Our experiments show that our new classifier outperforms state-of-the-art methods on average, achieving improvements of up to 7.00% and 7.72% for three-way and four-way classification problems respectively (Section 4).

2 Related Work

Pang and Lee (2005) incorporated information about label similarities using *metric labeling*, where label relations were encoded via a distance metric. The output of standard k-ary classifiers was then modified such that similar items were more likely to be assigned similar labels. Metric labeling required a label-corrected item-similarity function, which was based on the observation that the *Percentage of Positive Sentences* (*PSP*) in reviews increased as their ratings increased. Notice, however, that item similarity was not incorporated into the first stage of classifier training. Metric labeling adjusted the output of the classifiers only after they were trained without considering rating similarities. Our approach accounts for inter-category relationships from the outset of classifier design, rather than addressing this issue with later adjustments.

Goldberg and Zhu (2006) proposed a semi-supervised learning approach to the rating inference problem in scenarios where labeled training data is scarce. Using a graph-based optimisation approach, Goldberg and Zhu demonstrated that the inclusion of unlabeled reviews in the learning process could produce significantly higher prediction accuracy than predictors trained without unlabeled data. This approach outperformed competing methods when it considered relatively small numbers of labeled samples from the four-category movie review dataset (Pang and Lee, 2005). However, the graph-based method did not perform well when a large number of labeled samples was available. Furthermore, Goldberg and Zhu's graph-based learning method was transductive: new samples could not be classified until they were added to the graph — a problem avoided by our approach.

Koppel and Schler (2006) considered neutral examples, which may express a mixed opinion or may not express any opinion at all, in addition to positive/negative samples. Their experiments showed that neutral examples often did not lie close to the positive/negative decision boundary as previously believed. This gave rise to the idea of "optimal stacks" of SVMs, which were pairwise combinations of binary classifiers that distinguish between two categories for the ternary positive/neutral/negative problem (instead of a single binary classifier trained using only positive and negative samples). The search for an optimal stack is exponential in time. Hence, finding suitable stacks is feasible for the ternary problem, but becomes intractable for larger numbers of categories (in the general case).

Snyder and Barzilay (2007) proposed the "Good Grief" algorithm, which considers multiple aspects of a situation (e.g., a restaurant review that covers service, ambiance and food), and yields a prediction that minimises the dissatisfaction (grief) regarding these aspects. This method significantly outperformed baseline methods and individual classifiers. At present, we do not consider separately different aspects of a review — a task we intend to undertake in the future.

3 Multiclass SVM Classifiers

Since SVMs are binary classifiers, they are often employed for binary sentiment detection. However, as seen above, it is not straightforward to use SVMs for multi-way classification, particularly when there is inter-class similarity.

One might initially expect that a hierarchical SVM classifier could be built using pairwise comparisons of adjacent class labels. However, pairwise comparisons alone do not form a complete classifier, raising the question of how to combine pairwise classifications. The standard techniques to build k-way SVM classifiers are OVA and OVO (Hsu and Lin, 2002), and DAGSVM schemes (Platt et al., 2000). An OVA classifier requires k SVMs for a k-category problem, where the i^{th} SVM is trained using all samples from the i^{th} category versus all other samples. A sample is classified by evaluating all k trained SVMs, and the label of the class which maximizes the decision function is chosen. The OVO scheme trains $\frac{k(k-1)}{2}$ classifiers derived from a pairwise comparison of the target categories. A prediction is made by evaluating each SVM and recording "votes" for the favoured category: the class with the most votes is selected as the predicted category. The DAGSVM scheme builds a *Directed Acyclic Graph* (*DAG*) where each non-leaf node has an SVM that discriminates between two classes. A DAGSVM is iteratively constructed in a top-down fashion by forming a list of all the class labels, and creating a decision node that discriminates between the first and last element of the list. This decision node yields two child nodes, each of which omits one of the two classes that were compared. Each of these nodes then discriminates between the first and last element in its list of classes, and so on. This process continues for each decision path until only one element remains in the list. A sample is classified by successively making decisions down the graph until a leaf node is reached. Like OVO, DAGSVM schemes require training $\frac{k(k-1)}{2}$ decision nodes.

All three techniques suffer from long training times — an issue that is exacerbated by large data sets such as our corpus of approximately 5000 movie reviews (Section 4.1). Additional problems associated with these techniques are: (1) there is no bound on the generalisation error of OVA, (2) OVO schemes tend to overfit, and (3) the performance of a DAGSVM relies on the order in which classes are processed. This order is based on the class labels (rather than similarity between samples), and no practical method is known to optimize this order.

Overfitting also arises when the number of features is very large compared to the number of training samples. In this case, the SVM training process may discover a decision plane that separates the training data well, but performs poorly on unseen test samples. While SVM training algorithms use regularisation to address the overfitting problem, research has shown that a careful reduction in feature vector dimensionality can help combat overfitting (Weston et al., 2003).

A fundamental problem with the above three schemes is that the similarity between samples of nearby classes is not considered. Instead, categories are assumed to be independent. This problem may be addressed by considering SVM regression (SVM-R) (Smola and Schölkopf, 1998), where class labels are assumed to come from a discretisation of a continuous function that maps the feature space to a metric space. However, SVM-R, like the SVM schemes described here, trains on the entire feature set for all the classes in the dataset. In the case of sentiment detection, where words and punctuation marks are commonly taken as features, the sheer number of features may overwhelm the number of training samples, and lead to the model overfitting the data. SVM-R also poses the question of how to quantise the regressor's output to produce discrete class predictions.

3.1 The MCST-SVM Classifier

To address the above problems, we build a decision tree of SVMs that reduces the set of possible classes at each decision node, and takes relative class similarity into account during the tree construction process. We construct the decision tree as a Minimum Cost Spanning Tree (MCST), denoted *MCST-SVM*, based on inter-class similarity measured from feature values (Lorena and de Carvalho, 2005). Each of the decision tree leaves corresponds to a target class, and the interior nodes group classes into disjoint sets. For each internal node in the MCST, an SVM is trained to separate all the samples belonging to classes in its left subtree from those in its right subtree. We use linear SVMs, which have been shown to be effective text classifiers (Pang et al., 2002; Pang and Lee, 2005), and set the SVM parameters to match those used in (Pang and Lee, 2005).[1] Figure 1 contrasts

[1] SVMs are implemented using the C/C++ library `liblinear`, a variant of `libsvm` (Chang and Lin, 2001).

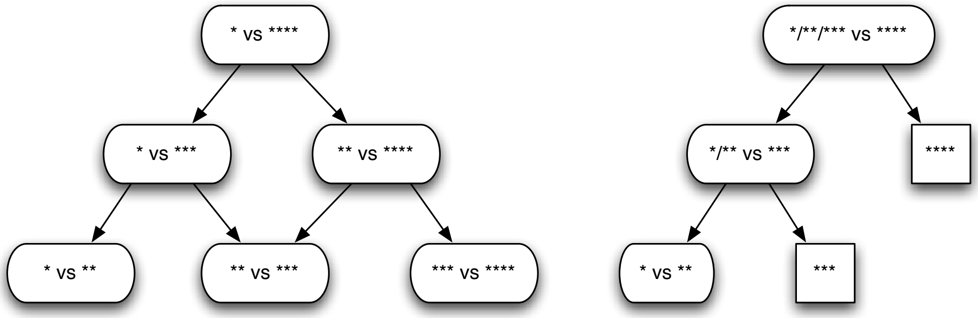


Figure 1: Top section of DAGSVM (left) versus MCST-SVM (right).

the DAGSVM and MCST-SVM approaches for a four-class example.

The MCST is constructed using Kruskal's algorithm (1956), which works in polynomial time (Algorithm 1). This algorithm requires a measure of the similarity between every pair of classes, which is calculated using the distance between a *representative vector* for each class (Section 3.2). The MCST is iteratively built in a bottom-up fashion, beginning with all classes as singleton nodes. In each iteration, the algorithm constructs a node comprising the most similar sets of classes from two previously generated nodes. The similarity between two sets of classes is the shortest distance between the representative vectors of the classes in each set. For instance, the shortest distance between the sets of classes $\{*/**\}$ and $\{***/****\}$ is $\min\{$dist(*,***), dist(*,****), dist(**,***), dist(**,****)$\}$. An SVM is then trained to discriminate between the children of the constructed nodes.

With respect to the example in Figure 1, the classes $\{*\}$ and $\{**\}$ are first found to be the most similar, thus forming a node which discriminates between these two classes. In the next iteration, the classes $\{**\}$ and $\{***\}$ are found to be the next most similar, producing a new node which discriminates between $\{*/**\}$ and $\{***\}$. Since the most similar sets are considered lower in the tree, the sets closer to the root of the tree are progressively more dissimilar, until the root node discriminates between the two most dissimilar sets of classes.

Our approach resembles DAGSVMs in that the structure of the decision tree is important. However, unlike DAGSVMs, the MCST-SVM structure is inferred on the basis of similarity between the observed *features* of the data, which are known, rather than the *labels* of the classes, which we are trying to predict. We assume that classes with adjacent labels are similar in the feature space, but if this does not happen in the training data, the MCST-SVM will yield a structure that exploits inter-class similarity irrespective of class labels. Further, our reliance on features supports experimentation with different methods for calculating inter-class similarity (Section 3.2). An additional advantage of MCST-SVM classifiers over the other schemes is that MCST-SVM requires only $k-1$ decision nodes for a k-class problem (and a maximum of $k-1$ decisions to make a prediction). That is, only $k-1$ SVMs must be trained, thereby reducing training time.

3.2 Class Similarity Measures

As mentioned in Section 3.1, the construction of an MCST-SVM classifier requires the computation of a similarity measure between classes. The MCST-SVM method may use any measure of inter-class similarity during the tree construction stage, and many such methods exist (e.g., linear discriminant analysis to order a tree of classifiers (Li et al., 2007)). We elected to use class prototypes to calculate similarity since they have achieved good performance in previous MCST-SVM applications (Lorena and de Carvalho, 2005; Bickerstaffe et al., 2007), and are fast to compute over many documents with a large feature space.

Algorithm 1 Constructing the MCST-SVM

```
Let $V$ be a set of graph vertices, where each vertex $v_i \in V$ represents rating class $i$ and its available training samples. $\forall i$ compute $r_i$, the class representative for rating class $i$.
Let $E$ be a set of graph edges. $\forall i, j$ where $i \neq j$, compute $e_{i,j} \in E$, the distance between class representatives $r_i$ and $r_j$.
Sort the members of $E$ in ascending order.
$\forall i$, let $S_i = v_i$, and add $S_i$ as a singleton node to the MCST-SVM tree $T$.
Let $i = 0$ and $j = 0$ be counting variables.
while $i < |V| - 1$ do
    Select the $j$-th edge according to the ordering of inter-class distances.
    if the vertices of the edge are in disjoint sets $S_p$ and $S_q$ then
        Define $S_p$ as a positive class and $S_q$ as a negative class.
        Let $S_t = S_p \cup S_q$, and add a new node containing $S_t$ to $T$.
        Connect the left and right branches of the node containing $S_t$ to the nodes containing $S_p$ and $S_q$ respectively.
        Remove $S_p$ and $S_q$.
        $i = i + 1$.
    end if
    $j = j + 1$.
end while
Train a binary SVM for each non-leaf node of $T$.
Return the MCST-SVM tree $T$.
```

We first determine a representative feature vector for each class, and then calculate the distance between these representative vectors.

Determining a representative vector. Each review is represented as a vector of boolean attributes, where each attribute indicates the presence or absence of a word or punctuation mark in the text. We elect to use boolean attributes since they have been shown to be advantageous over term-frequency approaches for sentiment detection, particularly when SVMs are employed (Pang et al., 2002). We considered two ways of determining a representative vector: *centroid* and *sample selection*.

- **Centroid.** Given N boolean feature vectors $\boldsymbol{a}_i$ of length n, compute the centroid vector $\boldsymbol{m}$ with values

$$m_j = \frac{1}{N} \sum_{i=1}^{N} a_{i,j} \quad \text{for } j = 1, \ldots, n \,. \qquad (1)$$

This measure produces a representative vector that contains the proportion of training samples for which each feature occurs.

- **Sample selection.** From the training samples of each class, select one sample which maximises the average *Tanimoto coefficient* (Tanimoto, 1957) with respect to all other samples in that class. The Tanimoto coefficient is an extension of cosine similarity which yields the Jaccard coefficient for boolean feature vectors. Given two boolean vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, the Tanimoto coefficient is defined as

$$d_t(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2 - \boldsymbol{a} \cdot \boldsymbol{b}} \,, \qquad (2)$$

where larger values of d_t indicate a higher degree of similarity between boolean vectors. This measure chooses a representative vector which on average has the most "overlap" with all other vectors in the class. We use Tanimoto distance, rather than the classical cosine similarity measure, since we employ boolean valued features instead of term-frequency features.

Calculating distance between vectors. We propose two methods to perform this task: *Euclidean distance* and the *Tanimoto coefficient*.

- **Euclidean distance** is used when the vectors that represent a class are centroid vectors (real-valued).
- The **Tanimoto coefficient** is used when the representative vectors of a class are boolean valued. It is calculated using Equation 2.

3.3 Irrelevant Feature Culling

The MCST-SVM scheme provides a natural mechanism for reducing the dimensionality of feature vectors in order to address the overfitting problem. This is due to the fact that each internal decision node is trained using *only* the samples that belong to the classes relevant to this node. The reviews for these classes are likely to omit some of the words that appear in the reviews for classes that are relevant to other nodes, in particular in the lower layers of the tree. Consequently, an internal node can be trained using a *subset* of the features that occur in the entire training dataset. This subset contains only those features which are observed in the samples relevant to training the node in question.[2] Section 4.2 shows that when tested on "real world" datasets, this method can remove thousands of irrelevant features and improve classifier performance, while reducing memory requirements and training times.

4 Experiments and Results

In this section, we evaluate the MCST-SVM classifier described in Section 3. First, we systematically compare the performance of the different variants of this method: (1) with or without culling irrelevant features, and (2) using the centroid/Euclidean-distance combination or the Tanimoto coefficient to measure inter-class similarity. We then compare the best of these methods with Pang and Lee's (2005). Our results show that a combination of relatively small improvements can achieve a substantial boost in classifier performance, yielding significant improvements over Pang and Lee's results.

All our experiments are performed with 10-fold cross validation, and the results are assessed using classification accuracy.[3] "Significance" refers to statistical significance determined by a paired t-test, with $p < 0.05$.

4.1 Dataset

Our experiments were conducted on the *Sentiment Scale* dataset (v1.0),[4] which comprises four sub-corpora of 1770, 902, 1307 and 1027 movie reviews with an associated mapping to a three and four-star rating for each review.[5] Each sub-corpus is written by a different author (denoted Author A, B, C and D respectively), thus avoiding calibration error between individual authors and their ratings. Review texts are automatically filtered to leave only subjective sentences (motivated by the results described in (Pang and Lee, 2004)); the mean number of words per review in each subjective-filtered sub-corpus is 435, 374, 455 and 292 respectively.

4.2 MCST-SVM Variants

Table 1 summarizes the results for the four MCST-SVM variants (the results that are statistically significant compared to the centroid/no-culling option are boldfaced).

Feature culling. Our results show that feature culling produces some improvement in classifier accuracy for all the three-class and four-class datasets. The impact of feature culling is statistically significant for all the four-class datasets when coupled with the Tanimoto coefficient. However, such an effect was not observed for the centroid/Euclidean-distance measure. In the three-class datasets, the improvements from feature culling are marginal for Authors A, B and C, but statistically significant for Author D (4.61%), both when using the centroid/Euclidean-distance measure and the Tanimoto coefficient. We posit that feature culling affects Author D because it reduces the overfitting problem, which caused the initially poor performance of MCST-SVM without culling on this author's short review texts (the reviews by this author, with 292 words on average, are the shortest in the Sentiment Scale dataset by a large margin, Section 4.1). Despite this improvement, all the MCST-SVM variants (as well as Pang and Lee's methods) exhibit worse performance for Authors B and D, who have shorter reviews, than for Authors A and C.

The culling of irrelevant features also has the benefit of reducing node training times and facil-

[2] The root node always considers all classes and therefore considers all features across the whole training dataset.

[3] We also have results for mean absolute error (MAE), which confirm our classification accuracy results.

[4] `http://www.cs.cornell.edu/People/pabo/moviereview-data.`

[5] In principle, classifiers for the three- and four-class ratings of the Sentiment Scale dataset could be enumerated using optimal stacks of SVMs. However, we wish to directly compare our method with Pang and Lee's (2005). Higher-discrimination datasets (for which optimal stacks are infeasible) will be tested in the future.

	Centroid, no culling	**Tanimoto, no culling**	**Centroid, with culling**	**Tanimoto, with culling**
Three-class				
Author A	70.396	70.396	71.017	71.997
Author B	60.556	60.556	61.111	61.111
Author C	75.154	75.481	76.231	76.923
Author D	59.608	59.608	**64.216**	**64.216**
Four-class				
Author A	62.429	63.810	63.090	**65.720**
Author B	49.111	49.792	50.622	**52.890**
Author C	64.846	65.689	65.692	**66.985**
Author D	49.118	49.626	51.177	**51.873**

Table 1: Performance accuracy (percentage correct predictions) for MCST-SVM variants.

itating a memory-efficient implementation. For example, without feature culling, the nodes of an MCST-SVM for Author A in the four-class dataset take training samples with 19752 features. In contrast, when irrelevant feature culling is applied, the number of features for each of the two non-root decision nodes reduces to 15445 and 17297. This corresponds to a total space saving of 6582 features $((19752 - 15445) + (19752 - 17297))$, yielding an in-memory reduction of 16.7%. Such memory reductions are particularly important for large datasets that may have trouble fitting within typical memory limitations. Node training times are also reduced by up to approximately 10%.

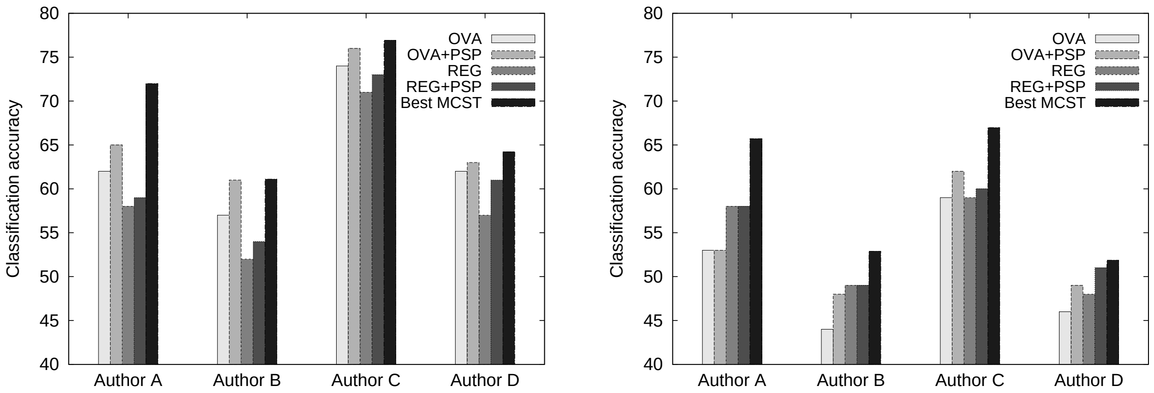
Class similarity measures. As mentioned above, Table 1 shows that the Tanimoto coefficient, coupled with feature culling, yields marginally better results than the centroid/no-culling option for most authors in the three-class dataset, and significantly better results for all the authors in the four-class dataset. The Tanimoto coefficient generally matches or outperforms the centroid/Euclidean-distance measure both with feature culling (Columns 4 and 5 in Table 1) and without feature culling (Columns 2 and 3). However, without feature culling, these improvements are not statistically significant.

For most cases in the three-star dataset, the tree structures found using the Tanimoto coefficient are identical to those found using the Euclidean-centroid option, hence the performance of the classifier is unchanged. For some validation folds, the Tanimoto coefficient discovered tree structures that differed from those found by the Euclidean-centroid option, generally yielding small accuracy improvements (e.g., 0.98% for Author A in the three-star dataset, with feature culling). The Tanimoto coefficient provides a greater benefit for the four-class dataset. Specifically, when feature culling is used (Columns 4 and 5 in Table 1), accuracy improves by 2.63% and 2.27% for Authors A and B respectively (statistically significant), and by 1.29% and 0.70% for Authors C and D respectively. This may be explained by the fact that there are many more tree structures possible for the four-class case than the three-class case, thereby increasing the impact of the inter-class similarity measure for the four-class case. However, this impact is significant only in conjunction with feature culling.

4.3 Comparison with Pang and Lee (2005)

Figure 2 compares the performance of the algorithms presented in (Pang and Lee, 2005) against the performance of the best MCST-SVM variant, which employs feature culling and uses the Tanimoto coefficient to compute inter-class similarity (Section 4.2). As per (Pang and Lee, 2005), REG indicates SVM-R, which is the baseline ordinal regression method. The suffix "+PSP" denotes methods that use the metric labeling scheme. We excluded DAGSVM from our results to maintain consistency with Pang and Lee's experiments. However, according to (Platt et al., 2000), the performance difference between DAGSVM and OVA is not statistically significant.

Generally, the MCST-SVM is competitive against all the classifiers presented in (Pang and Lee, 2005), and in some cases significantly outperforms these methods. Specifically, the hierar-



(a) Three-class data.

(b) Four-class data.

Figure 2: Best MCST-SVM versus competing methods.

chical classifier outperforms OVA+PSP by 7% in the three-class case for Author A (statistically significant), while in the four-class case the MCST-SVM outperforms the best competing methods by 7.72%, 3.89% and 4.98% for Authors A, B, and C respectively (statistically significant). The small improvement of 0.87% for Author D indicates that our approach has the most impact for reviews that contain a relatively large amount of subjective text.

5 Conclusion and Future Work

This paper described a hierarchical classifier applied to multi-way sentiment detection. The classifier is built by exploiting inter-class similarities to arrange high-performance binary discriminators (SVMs) into a tree structure. Since our inter-class similarity measures are based on sample features, they make the problem of structure determination tractable, and enable experimentation with different similarity measures. The resultant structures provide a natural mechanism to remove irrelevant features at each level of the tree, thus reducing the dimensionality of the feature space, which in turn reduces memory requirements. Importantly, these benefits are achieved while improving upon state-of-the-art classification performance, in particular with respect to higher-discrimination datasets.

The MCST-SVM classifier can be generalised to any number of classes, and is extendable in the sense that the classifier algorithm employed in each tree node may be replaced by other classifier algorithms as technology advances. The MCST-SVM classifier is also versatile, and may be applied to variations on the rating classification problem, e.g., traditional text classification.

The MCST-SVM algorithm is not specific to sentiment detection. However, it has several properties which make it particularly suitable for the rating inference problem. Firstly, the MCST-SVM accounts for inter-class similarity and is therefore capable of capturing the ordinal nature of ratings. Secondly, the tree structures permit irrelevant feature culling, which in turn reduces memory requirements and training times.

Future work will involve testing our approach with higher-discrimination datasets, developing methods to pre-process review texts (e.g., improved negation tagging, and incorporating part-of-speech tagging), and further addressing the problem of overfitting. To this effect we will investigate different feature selection algorithms, e.g., (Weston et al., 2003), and their utilisation within the classifier trees. We also propose to consider aspects of reviews (Snyder and Barzilay, 2007), and investigate other methods that measure class similarity, such as selecting typical instances (Zhang, 1992).

Acknowledgments

This research is supported in part by ARC grant LP0883416 and GapBuster Worldwide.

References

Allison, B. 2008. Sentiment detection using lexically-based classifiers. In *Proceedings of the 11th International Conference on Text, Speech and Dialogue*, pages 21–28, Brno, Czech Republic.

Beygelzimer, A., J. Langford, and P. Ravikumar. 2009. Error-correcting tournaments. In *Proceedings of the 20th International Conference on Algorithmic Learning Theory*, pages 247–262, Porto, Portugal.

Bickerstaffe, A., A. Lane, B. Meyer, and K. Marriott. 2007. Building smart diagram environments with domain-specific gesture recognizers. In *Proceedings of the 7th IAPR International Workshop on Graphics Recognition*, pages 145–156, Curitiba, Brazil.

Chang, C.C. and C.J. Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Goldberg, A.B. and X. Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *TextGraphs: Workshop on Graph Based Methods For NLP*, pages 45–52, New York, New York.

Hsu, C. W. and C. J. Lin. 2002. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425.

Koppel, M. and J. Schler. 2006. The importance of neutral examples for learning sentiment. *Computational Intelligence*, 22(2):100–109.

Kruskal, J. B. 1956. On the shortest spanning subtree and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50.

Li, T., S. Zhu, and M. Ogihara. 2007. Hierarchical document classification using automatically generated hierarchy. *Journal of Intelligent Information Systems*, 29(2):211–230.

Lorena, A. C. and A. C. P. L. F. de Carvalho. 2005. Minimum spanning trees in hierarchical multiclass Support Vector Machines generation. *Innovations in Applied Artificial Intelligence*, 3533:422–431.

Mao, Y. and G. Lebanon. 2006. Isotonic conditional random fields and local sentiment flow. In *Proceedings of the 20th Annual Conference on NIPS*, pages 961–968, British Columbia, Canada.

McDonald, R., K. Hannan, T. Neylon, M. Wells, and J. Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *Proceedings of the 45th Annual Meeting of the ACL*, pages 432–439, Prague, Czech Republic.

Pang, B. and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the ACL*, pages 271–278, Barcelona, Spain.

Pang, B. and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 115–124, Ann Arbor, Michigan.

Pang, B., L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in NLP*, pages 79–86, Philadelphia, Pennsylvania.

Platt, J. C., N. Cristinini, and J. Shawe-Taylor. 2000. Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems*, 12:547–553.

Smola, A. and B. Schölkopf. 1998. A Tutorial on Support Vector regression. Technical Report COLT NC-TR-98-030, University of London.

Snyder, B. and R. Barzilay. 2007. Multiple aspect ranking using the Good Grief algorithm. In *Proceedings of HLT/NAACL*, pages 300–307, Rochester, New York.

Tanimoto, T.T. 1957. IBM internal report.

Weston, J., A. Elisseff, B. Schölkopf, and M. Tipping. 2003. Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3:1439–1461.

Wilson, T., J. Wiebe, and P. Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Empirical Methods in NLP*, pages 347–354, Vancouver, Canada.

Yu, H. and V. Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the Conference on Empirical Methods in NLP*, pages 129–136, Sapporo, Japan.

Zhang, J. 1992. Selecting typical instances in instance-based learning. In *Proceedings of the 9th International Workshop on Machine Learning*, pages 470–479, Aberdeen, Scotland.

Fluency Constraints for Minimum Bayes-Risk Decoding of Statistical Machine Translation Lattices

Graeme Blackwood and **Adrià de Gispert** and **William Byrne**
Machine Intelligence Laboratory, Department of Engineering, Cambridge University
{gwb24|ad465|wjb31}@cam.ac.uk

Abstract

A novel and robust approach to improving statistical machine translation fluency is developed within a minimum Bayes-risk decoding framework. By segmenting translation lattices according to confidence measures over the maximum likelihood translation hypothesis we are able to focus on regions with potential translation errors. Hypothesis space constraints based on monolingual coverage are applied to the low confidence regions to improve overall translation fluency.

1 Introduction and Motivation

Translation quality is often described in terms of *fluency* and *adequacy*. Fluency reflects the 'nativeness' of the translation while adequacy indicates how well a translation captures the meaning of the original text (Ma and Cieri, 2006).

From a purely utilitarian view, adequacy should be more important than fluency. But fluency and adequacy are subjective and not easy to tease apart (Callison-Burch et al., 2009; Vilar et al., 2007). There is a human tendency to rate less fluent translations as less adequate. One explanation is that errors in grammar cause readers to be more critical. A related phenomenon is that the nature of translation errors changes as fluency improves so that any errors in fluent translations must be relatively subtle. It is therefore not enough to focus solely on adequacy. SMT systems must also be fluent if they are to be accepted and trusted. It is possible that the reliance on automatic metrics may have led SMT researchers to pay insufficient attention to fluency: BLEU (Papineni et al., 2002), TER (Snover et al., 2006), and METEOR (Lavie and Denkowski, 2009) show broad correlation with human rankings of MT quality, but are incapable of fine distinctions between fluency and adequacy.

There is concern that the fluency of current SMT is inadequate (Knight, 2007b). SMT is robust, in that a translation is nearly always produced. But unlike translators who should be skilled in at least one of the languages, SMT systems are limited in both source and target language competence. Fluency and accuracy therefore tend to suffer together as translation quality degrades. This should not be the case. Ideally, an SMT system should never be any less fluent than the best *stochastic text generation* system available in the target language (Oberlander and Brew, 2000). What is needed is a good way to enhance the fluency of SMT hypotheses.

The maximum likelihood (ML) formulation (Brown et al., 1990) of translation of source language sentence F to target language sentence $\hat{E}$

$$\hat{E} = \underset{E}{\operatorname{argmax}} P(F|E)P(E) \tag{1}$$

makes it clear why improving SMT fluency is a difficult modelling problem. The language model $P(E)$, the closest thing to a 'fluency component' in the original formulation, only affects candidates likely under the translation model $P(F|E)$. Given the weakness of current translation models this is a severe limitation. It often happens that SMT systems assign $P(F|\bar{E}) = 0$ to a correct reference translation $\bar{E}$ of F (see the discussion in Section 9). The problem is that in ML decoding the language model can only encourage the production of fluent translations; it cannot easily enforce constraints on fluency or introduce new hypotheses.

In Hiero (Chiang, 2007) and syntax-based SMT (Knight and Graehl, 2005; Knight, 2007a), the primary role of syntax is to drive the translation process. Translations produced by these systems respect the syntax of their translation models, but

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 71–79,
Beijing, August 2010

this does not force them to be grammatical in the way that a typical human sentence is grammatical; they produce many translations which are not fluent. The problem is robustness. Generating fluent translations demands a tightly constraining target language grammar but such a grammar is at odds with broad-coverage parsing needed for robust translation.

We have described two problems in translation fluency: (1) SMT may fail to generate fluent hypotheses and there is no simple way to introduce them into the search; (2) SMT produces many translations which are not fluent but enforcing constraints to improve fluency can hurt robustness. Both problems are rooted in the ML decoding framework in which robustness and fluency are conflicting objectives.

We propose a novel framework to improve the fluency of any SMT system, whether syntactic or phrase-based. We will perform Minimum Bayes-risk search (Kumar and Byrne, 2004) over a space of fluent hypotheses $\mathcal{H}$:

$$\hat{E}_{\text{MBR}} = \operatorname*{argmin}_{E' \in \mathcal{H}} \sum_{E \in \mathcal{E}} L(E, E') P(E|F) \tag{2}$$

In this approach the MBR evidence space $\mathcal{E}$ is generated by an SMT system as a k-best list or lattice. The system runs in its best possible configuration, ensuring both translation robustness and good baselines. Rather than decoding in the output of the baseline SMT system, translations will be sought among a collection of fluent sentences that are close to the top SMT hypotheses as determined by the loss function $L(E, E')$.

Decoupling the MBR hypothesis space from first-pass translation offers great flexibility. Hypotheses in $\mathcal{H}$ may be arbitrarily constrained according to lexical, syntactic, semantic, or other considerations, with no effect on translation robustness. This is because constraints on fluency do not affect the production of the evidence space by the baseline system. Robustness and fluency are no longer conflicting objectives. This framework also allows the MBR hypothesis space to be augmented with hypotheses produced by an NLG system, although this is beyond the scope of the present paper.

This paper focuses on searching out fluent strings amongst the vast number of hypotheses encoded in SMT lattices. Oracle BLEU scores computed over k-best lists (Och et al., 2004) show that many high quality hypotheses are produced by first-pass SMT decoding. We propose reducing the difficulty of enhancing the fluency of complete hypotheses by first identifying regions of high-confidence in the ML translations and using these to guide the fluency refinement process. This has two advantages: (1) we keep portions of the baseline hypotheses that we trust and search for alternatives elsewhere, and (2) the task is made much easier since the fluency of sentence fragments can be refined in context.

In what follows, we use posterior probabilities over SMT lattices to identify useful subsequences in the ML translations (Sections 2 & 3). These subsequences drive the segmentation and transformation of lattices into smaller subproblems (Sections 4 & 5). Subproblems are mined for fluent strings (Section 6), resulting in improved translation fluency (Sections 7 & 8). Our results show that, when guided by the careful selection of subproblems, fluency can be improved with no real degradation of the BLEU score.

2 Lattice MBR Decoding

The formulation of the MBR decoder in Equation (2) separates the hypothesis space from the evidence space. We apply the linearised lattice MBR decision rule (Tromble et al., 2008)

$$\hat{E}_{\text{LMBR}} = \operatorname*{argmax}_{E' \in \mathcal{H}} \left\{ \theta_0 |E'| + \sum_{u \in \mathcal{N}} \theta_u \#_u(E') p(u|\mathcal{E}) \right\}, \tag{3}$$

where $\mathcal{H}$ is the hypothesis space, $\mathcal{E}$ is the evidence space, $\mathcal{N}$ is the set of all n-grams in $\mathcal{H}$ (typically, $n = 1 \dots 4$), and θ are constants estimated on held-out data. The quantity $p(u|\mathcal{E})$ is the path posterior probability of n-gram u

$$p(u|\mathcal{E}) = \sum_{E \in \mathcal{E}_u} P(E|F), \tag{4}$$

where $\mathcal{E}_u = \{E \in \mathcal{E} : \#_u(E) > 0\}$ is the subset of paths containing n-gram u at least once. The path posterior probabilities $p(u|\mathcal{E})$ of Equation (4) can be efficiently calculated (Blackwood et al., 2010) using general purpose WFST operations (Mohri et al., 2002).

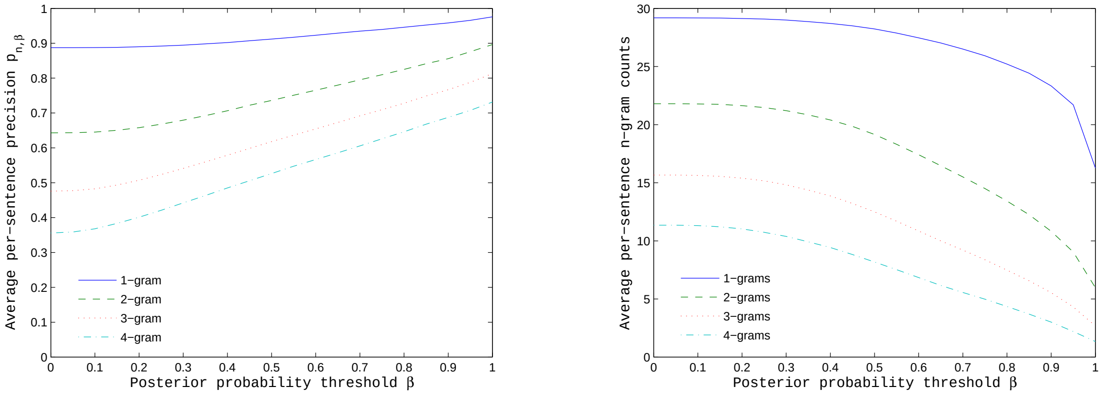


Figure 1: Average n-gram precisions (left) and counts (right) for 2075 sentences of NIST Arabic→English ML translations at a range of posterior probability thresholds $0 \leq \beta \leq 1$. The left plot shows at $\beta = 0$ the n-gram precisions used in the BLEU score of the ML baseline system.

3 Posterior Probability Confidence Measures

In the formulation of Equations (3) and (4) the path posterior n-gram probabilities play a crucial role. MBR decoding under the linear approximation to BLEU is driven mainly by the presence of high posterior n-grams in the lattice; the low posterior n-grams contribute relatively little to the MBR decision criterion. Here we investigate the predictive power of these statistics. We will show that the n-gram posterior is a good predictor as to whether or not an n-gram is to be found in a set of reference translations.

Let $\mathcal{N}_n$ denote the set of n-grams of order n in the ML hypothesis $\hat{E}$, and let $\mathcal{R}_n$ denote the set of n-grams of order n in the union of the references. For confidence threshold β, let $\mathcal{N}_{n,\beta} = \{u \in \mathcal{N}_n : p(u|\mathcal{E}) \geq \beta\}$ denote the n-grams in $\mathcal{N}_n$ with posterior probability greater than or equal to β, where $p(u|\mathcal{E})$ is computed using Equation (4). This is equivalent to identifying all substrings of length n in the translation hypotheses for which the system assigns a posterior probability of β or higher. The precision at order n for threshold β is the proportion of n-grams in $\mathcal{N}_{n,\beta}$ also present in the references:

$$\mathcal{P}_{n,\beta} = \frac{|\mathcal{R}_n \cap \mathcal{N}_{n,\beta}|}{|\mathcal{N}_{n,\beta}|} \qquad (5)$$

The left plot in Figure 1 shows average per-sentence n-gram precisions $\mathcal{P}_{n,\beta}$ at orders $1 \ldots 4$ for an Arabic→English translation task at a range of thresholds $0 \leq \beta \leq 1$. Sentence start and end tokens are ignored when computing unigram precisions. We note that precision at all orders improves as the threshold β increases. This confirms that these intrinsic measures of translation confidence have strong predictive power.

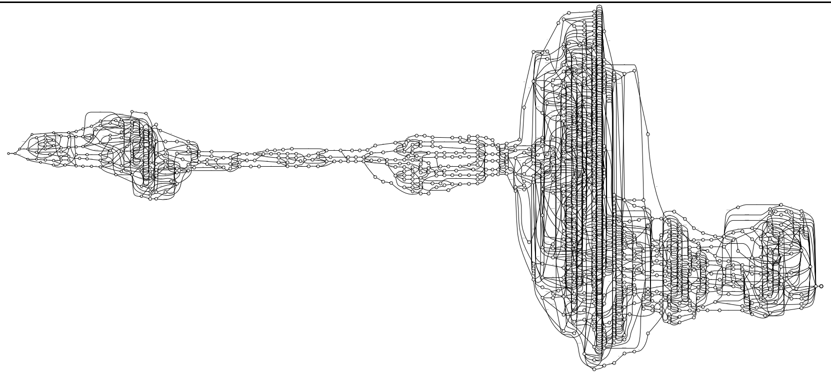
The right-hand side of the figure shows the average number of n-grams per sentence for the same range of β. We see that for high β, there are few n-grams with $p(u|\mathcal{E}) \geq \beta$; this is as expected. However, even at a high threshold of $\beta = 0.9$ there are still on average three 4-grams per sentence with posterior probabilities that exceed β. Even at this very high confidence level, high posterior n-grams occur frequently enough that we can expect them to be useful.

These precision results motivate our use of path posterior n-gram probabilities as a confidence measure. We assign confidence $p(\hat{E}_i^j|\mathcal{E})$ to subsequences $\hat{E}_i \ldots \hat{E}_j$ of the ML hypothesis.

Prior work focuses on word-level confidence extracted from k-best lists and lattices (Ueffing and Ney, 2007), while Zens and Ney (2006) rescore k-best lists with n-gram posterior probabilities. Similar experiments with a slightly different motivation are reported by DeNero et al. (2009); they show that expected n-gram counts in a lattice can be used to predict which n-grams appear in the references.

4 Lattice Segmentation

We have shown that current SMT systems, although flawed, can identify with confidence par-the newspaper “ constitution ” quoted brigadier abdullah krishan **, the chief of police in** karak governorate **(521 km south @-@ west of amman) as saying that the seizure** took place after **police received information that** there were attempts by the group to sell for more than $ 100 thousand dollars , **the police rushed to** the arrest in possession .



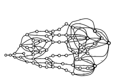
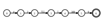
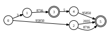
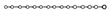
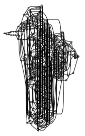
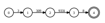
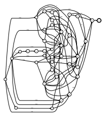
$\mathcal{H}_1$	$\mathcal{H}_2$	$\mathcal{H}_3$	$\mathcal{H}_4$	$\mathcal{H}_5$	$\mathcal{H}_6$	$\mathcal{H}_7$	$\mathcal{H}_8$	$\mathcal{H}_9$
								
433	1	4	1	6	1	6860	1	76

Figure 2: ML translation $\hat{E}$, word lattice $\mathcal{E}$, and decomposition as a sequence of four string and five sublattice regions $\mathcal{H}_1 \ldots \mathcal{H}_9$ using *n*-gram posterior probability threshold $p(u|\mathcal{E}) \geq 0.8$.

tial hypotheses that can be trusted. We wish to constrain MBR decoding to include these trusted partial hypotheses but allow decoding to consider alternatives in regions of low confidence. In this way we aim to improve the best possible output of the best available systems.

We use the path posterior n-gram probabilities of Equation (4) to segment lattice $\mathcal{E}$ into regions of high and low confidence. As shown in the example of Figure 2, the lattice segmentation process is performed relative to the ML hypothesis $\hat{E}$, i.e. relative to the best path through $\mathcal{E}$.

For confidence threshold β, we find all 4-grams $u = \hat{E}_i, \ldots, \hat{E}_{i+3}$ in the ML translation for which $p(u|\mathcal{E}) > \beta$. We then segment $\hat{E}$ into regions of high and low confidence where the high confidence regions are identified by consecutive, overlapping high confidence 4-grams. The high confidence regions are contiguous strings of words for which there is consensus amongst the translations in the lattice. If we trust the path posterior n-gram probabilities, any hypothesised translation should include these high confidence substrings. This approach differs from simple posterior-based pruning in that we discard paths, rather than words or n-grams, which are not consistent with high-confidence regions of the ML hypothesis.

The hypothesis string $\hat{E}$ is in this way segmented into R alternating subsequences of high and low confidence. The segment boundaries are i_r and j_r so that $\hat{E}_{i_r}^{j_r}$ is either a high confidence or a low confidence subsequence. Each subsequence is associated with an unweighted subspace $\mathcal{H}_r$; this subspace has the form of a string for high confidence regions and the form of a lattice for low confidence regions.

If the r^{th} segment is a high confidence region then $\mathcal{H}_r$ accepts only the string $\hat{E}_{i_r}^{j_r}$. If the r^{th} segment is a region of low confidence, then $\mathcal{H}_r$ is built to accept relevant substrings from $\mathcal{E}$. It is constructed as follows. The r^{th} low confidence region $\hat{E}_{i_r}^{j_r}$ has a high confidence left context $\hat{e}_{r-1}$ and a high confidence right context $\hat{e}_{r+1}$ formed from subsequences of the ML translation hypothesis $\hat{E}$ as

$$\hat{e}_{r-1} = \hat{E}_{i_{r-1}}^{j_{r-1}}, \quad \hat{e}_{r+1} = \hat{E}_{i_{r+1}}^{j_{r+1}}$$

Note that when $r = 1$ the left context $\hat{e}_{r-1}$ is the empty string and when $r = R$ the right context $\hat{e}_{r+1}$ is the empty string. We build a transducer

$\mathcal{T}_r$ for the regular expression $/.*\hat{e}_{r-1}(.*)\hat{e}_{r+1}.*/\backslash 1/$.[1] Composition with $\mathcal{E}$ yields $\mathcal{H}_r = \mathcal{E} \circ \mathcal{T}_r$, so that $\mathcal{H}_r$ contains all the reasonable alternatives to $\hat{E}_{i_r}^{j_r}$ in $\mathcal{E}$ consistent with the high confidence left and right contexts $\hat{e}_{r-1}$ and $\hat{e}_{r+1}$. If $\mathcal{H}_r$ is aligned to a high confidence subsequence of $\hat{E}$, we call it a *string region* since it contains a single path; if it is aligned to a low confidence region it is a lattice and we call it a *sublattice region*. The series of high and low confidence subspace regions $\mathcal{H}_1, \ldots, \mathcal{H}_R$ defines the lattice segmentation.

5 Hypothesis Space Construction

We now describe a general framework for improving the fluency of the MBR hypothesis space.

The segmentation of the lattice described in Section 4 considerably simplifies the problem of improving the fluency of its hypotheses since each region of low confidence may be considered independently. The low confidence regions can be transformed one-by-one and then reassembled to form a new MBR hypothesis space.

In order to transform the hypothesis region $\mathcal{H}_r$ it is important to know the context in which it occurs, i.e. the sequences of words that form its prefix and suffix. Some transformations might need only a short context; others may need a sentence-level context, i.e. the full sequence of ML words $\hat{E}_1^{j_r-1}$ and $\hat{E}_{i_{r+1}}^N$ to the left and right of the region $\mathcal{H}_r$ that is to be transformed.

To put this formally, each low confidence sublattice region is transformed by the application of some function Ψ:

$$\mathcal{H}_r \leftarrow \Psi(\hat{E}_1^{j_r-1},\, \mathcal{H}_r,\, \hat{E}_{i_{r+1}}^N) \tag{6}$$

The hypothesis space is then constructed from the concatenation of high confidence string and transformed low confidence sublattice regions

$$\mathcal{H} = \mathcal{E} \circ \bigotimes_{1 \le r \le R} \mathcal{H}_r \tag{7}$$

The composition with the original lattice $\mathcal{E}$ discards any new hypotheses that might be created via the unconstrained concatenation of strings from the $\mathcal{H}_r$. It may be that in some circumstances the introduction of new paths is good, but in what follows we test the ability to improve fluency by searching among existing hypotheses, and this ensures that nothing new is introduced.

Size of the Hypothesis Space If no new hypotheses are introduced by the operations Ψ, the size of the hypothesis space $\mathcal{H}$ is determined by the posterior probability threshold β. Only the ML hypothesis remains at $\beta = 0$, since all its subsequences are of high confidence, i.e. can be covered by n-grams with non-zero path posterior probability. At the other extreme, for $\beta = 1$, it follows that $\mathcal{H} = \mathcal{E}$ and no paths are removed, since any string regions created are formed from subsequences that occur on every path in $\mathcal{E}$.

We can therefore use β to tighten or relax constraints on the LMBR hypothesis space. At $\beta = 0$, LMBR returns only the ML hypothesis; at $\beta = 1$, LMBR is done over the full translation lattice. This is shown in Table 1, where the BLEU score approaches the BLEU score of unconstrained LMBR as β increases.

Note also that the size of the resulting hypothesis space is the product of the number of sequences in the sublattice regions. For Figure 2 at $\beta = 0.8$, this product is ~5.4 billion hypotheses. Even for fairly aggressive constraints on the hypothesis space, many hypotheses remain.

6 Monolingual Coverage Constraints

This section describes one implementation of the transformation function Ψ that we will show leads to improved fluency of machine translation output. This transformation is based on n-gram coverage in a large target language text collection: where possible, we filter the sublattice regions so that they contain only long-span n-grams observed in the text. Our motivation is that large monolingual text collections are good guides to fluency. If a hypothesis is composed entirely of previously seen high order n-grams, it is likely to be fluent and should be favoured.

Initial attempts to identify fluent hypotheses in sublattice regions by ranking according to n-gram LM scores were ineffective. Figure 3 shows the difficulties. We see that both the 4-gram Kneser-Ney and 5-gram stupid-backoff language models

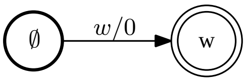
[1] In this notation parentheses indicate string matches so that $/.*y(a*)w.*/\backslash 1/$ applied to $xyaaawzz$ yields aaa.

LM	Translation hypothesis E and n-gram orders used by the LM to score each word	Score
4g	<s>$_1$ the$_2$ reactor$_3$ produces$_3$ plutonium$_2$ *needed*$_2$ *to*$_3$ *manufacture*$_4$ atomic$_3$ bomb$_2$.$_3$ </s>$_4$	-22.59
	<s>$_1$ the$_2$ reactor$_3$ produces$_3$ plutonium$_2$ *needed*$_2$ *to*$_3$ *manufacture*$_4$ *the*$_4$ atomic$_2$ bomb$_3$.$_4$ </s>$_4$	-23.61
5g	<s>$_1$ the$_2$ reactor$_3$ produces$_4$ plutonium$_5$ *needed*$_3$ *to*$_3$ *manufacture*$_4$ atomic$_5$ bomb$_2$.$_3$ </s>$_4$	-16.04
	<s>$_1$ the$_2$ reactor$_3$ produces$_4$ plutonium$_5$ *needed*$_3$ *to*$_3$ *manufacture*$_4$ *the*$_4$ atomic$_4$ bomb$_5$.$_4$ </s>$_5$	-17.96

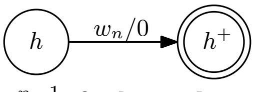
Figure 3: Scores and n-gram orders for hypotheses using 4-gram Kneser-Ney and 5-gram stupid-backoff (estimated from 1.1B and 6.6B tokens, resp.) LMs. Low confidence regions are in italics.

favour the shorter but disfluent hypothesis; normalising by length was not effective. However, the stupid-backoff LM has better coverage and the backing-off behaviour is a clue to the presence of disfluency. Similar cues have been observed in ASR analysis (Chase, 1997). The shorter hypothesis backs off to a bigram for "atomic bomb", whereas the longer hypothesis covers the same words with 4-grams and 5-grams. We therefore disregard the language model scores and focus on n-gram coverage. This is an example where robustness and fluency are at odds. The n-gram models are robust, but often favour less fluent hypotheses.

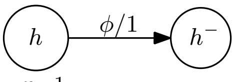
Let $\mathcal{S}$ denote the set of all n-grams in the monolingual training data. To identify partial hypotheses in sublattice regions that have complete monolingual coverage at the maximum order n, we build a coverage acceptor $\mathcal{C}_n$ with a similar form to the WFST representation of an n-gram backoff language model (Allauzen et al., 2003). $\mathcal{C}_n$ assigns a penalty to every n-gram not found in $\mathcal{S}$. In $\mathcal{C}_n$ word arcs have no cost and backoff arcs are assigned a fixed cost of 1. Firstly, arcs from the start state are added for each unigram $w \in \mathcal{N}_1$:



Then for n-grams $u \in \mathcal{S} \cap \{\cup_{i=2}^{n} \mathcal{N}_i\}$, where $u = w_1^n$ consisting of history $h = w_1^{n-1}$ and target word w_n, arcs are added



where $h^+ = w_2^{n-1}$ if u has order n and $h^+ = w_1^n$ if u has order less than n. Backoff arcs are added for each u as



where $h^- = w_2^{n-1}$ if u has order > 2, and bigrams backoff to the null history start state $\emptyset$.

For each sublattice region $\mathcal{H}_r$, we wish to penalise each path proportionally to the number of its n-grams not found in the monolingual text collection $\mathcal{S}$. We wish to do this in context, so that we include the effect of the neighbouring high confidence regions $\mathcal{H}_{r-1}$ and $\mathcal{H}_{r+1}$. Given that we are counting n-grams at order n we form the left context machine $\mathcal{L}_r$ which accepts the *last* $n-1$ words in $\mathcal{H}_{r-1}$; similarly, $\mathcal{R}_r$ accepts the *first* $n-1$ words of $\mathcal{H}_{r+1}$. The concatenation $\mathcal{X}_r = \mathcal{L}_r \otimes \mathcal{H}_r \otimes \mathcal{R}_r$ represents the partial translation hypotheses in $\mathcal{H}_r$ padded with $n-1$ words of left and right context from the neighbouring high confidence regions. Composing $\mathcal{X}_r \circ \mathcal{C}_n$ assigns each partial hypothesis a cost equal to the number of times it was necessary to back off to lower order n-grams while reading the string. Partial hypotheses with cost 0 did not back off at all and contain only maximum order n-grams.

In the following experiments, we look at each $\mathcal{X}_n \circ \mathcal{C}_n$ and if there are paths with cost 0, only these are kept and all others discarded. We introduce this as a constraint on the hypothesis space which we will evaluate for improvement on fluency. Here the transformation function Ψ returns $\mathcal{H}_r$ as $\mathcal{X}_r \circ \mathcal{C}_n$ after pruning. If $\mathcal{X}_r \circ \mathcal{C}_n$ has no zero cost paths, the transformation function Ψ returns $\mathcal{H}_r$ as we find it, since there is not enough monolingual coverage to guide the selection of fluent hypotheses. After applying monolingual coverage constraints to each region, the modified hypothesis space used for MBR search is formed by concatenation using Equation (7).

We note that $\mathcal{C}_n$ is a simplistic NLG system. It generates strings by concatenating n-grams found in $\mathcal{S}$. We do not allow it to run 'open loop' in these experiments, but instead use it to find the strings in $\mathcal{X}_r$ with good n-gram coverage.

7 LMBR Over Segmented Lattices

The effect of fluency constraints on LMBR decoding is evaluated in the context of the NIST Arabic→English MT task. The set *tune* consists

ML	... view , especially with *the open chinese economy* to the world and ...
+LMBR	... view , especially with *the open chinese economy* to the world and ...
+LMBR+CC	... view , especially with *the opening of the chinese economy* to the world and ...
ML	... revision of the constitution *of the japanese public* , which dates back ...
+LMBR	... revision of the constitution *of the japanese public* , which dates back ...
+LMBR+CC	... revision of the constitution *of japan* , which dates back ...

Figure 4: Improved fluency through the application of monolingual coverage constraints to the hypothesis space in MBR decoding of NIST MT 08 Arabic→English newswire lattices.

of the odd numbered sentences of the MT02–MT05 testsets; the even numbered sentences form *test*. MT08 performance on *nw08* (newswire) and *ng08* (newsgroup) data is also reported.

First-pass translation is performed using HiFST (Iglesias et al., 2009), a hierarchical phrase-based decoder. The first-pass LM is a modified Kneser-Ney (Kneser and Ney, 1995) 4-gram estimated over the English side of the parallel text and an 881M word subset of the English GigaWord 3rd Edition. Prior to LMBR, the first-pass lattices are rescored with zero-cutoff stupid-backoff 5-gram language models (Brants et al., 2007) estimated over more than 6B words of English text. The LMBR factors $\theta_0, \ldots, \theta_4$ are set as in Tromble et al. (2008) using unigram precision $p = 0.85$ and recall ratio $r = 0.74$.

The effect of performing LMBR over the segmented hypothesis space is shown in Table 1. The hypothesis subspaces $\mathcal{H}_r$ are constructed at various confidence thresholds as described in Section 4 with $\mathcal{H}$ formed via Equation (7); no coverage constraints are applied yet. Constraining the search space using $\beta = 0.6$ leads to little degradation in LMBR performance under BLEU. This shows lattice segmentation works as intended.

We next investigate the effect of monolingual coverage constraints on BLEU. We build acceptors $\mathcal{C}_n$ as described in Section 6 with $\mathcal{S}$ consisting of all n-grams in the English GigaWord. At $\beta = 0.6$ we found 181 sentences with sublattices $\mathcal{H}_r$ spanned by maximum order n-grams from $\mathcal{S}$, i.e. for which $\mathcal{X}_r \circ \mathcal{C}_n$ have paths with cost 0; these are filtered as described. LMBR over these coverage-constrained sublattices is denoted LMBR+CC. On *nw08* the BLEU score for LMBR+CC is 52.0 which is +0.7 over the ML decoder and only -0.2 BLEU below unconstrained LMBR decoding. Done in this way, constraining hypotheses to have 5-grams from the GigaWord has little impact on BLEU.

		tune	*test*	*nw08*	*ng08*
ML		54.2	53.8	51.3	36.3
β	0.0	54.2	53.8	51.3	36.3
	0.2	54.3	53.8	51.3	36.3
	0.4	54.6	54.2	51.6	36.7
	0.6	54.9	54.4	52.1	36.6
	0.8	54.9	54.4	52.1	36.6
	1.0	54.9	54.4	52.2	36.7
LMBR		54.9	54.4	52.2	36.8

Table 1: BLEU scores for ML hypotheses and LMBR decoding in $\mathcal{H}$ over $0 \leq \beta \leq 1$.

At this value of β, 116 of the 813 *nw08* sentences have a low confidence region (1) completely covered by 5-grams, and (2) within which the ML hypothesis and the LMBR+CC hypothesis differ. It is these regions which we will inspect for improved fluency.

8 Human Fluency Evaluation

We asked 17 native speakers to judge the fluency of sentence fragments from *nw08*. We compared hypotheses from the ML and the LMBR+CC decoders. Each fragment consisted of the partial translation hypothesis from a low confidence region together with its left and right high confidence contexts (examples given in Figure 4). For each sample, judges were asked: “Could this fragment occur in a fluent sentence?”

The results are shown in Table 2. Most of the time, the ML and LMBR+CC sentence fragments were both judged to be fluent; it often happened that they differed by only a single noun or verb substitution which didn't affect fluency. In a small number of cases, both ML and LMBR+CC were judged to be disfluent. We are most interested in the ‘off-diagonal’ cases. In cases when one system was judged to be fluent and the other was not, LMBR+CC was preferred about twice as often as the ML baseline (26.9% to 9.7%). In other words, the monolingual fluency constraints were judged

		LMBR+CC	
		Fluent	Not Fluent
ML	Fluent	1175 (59.6%)	192 (9.7%)
	Not Fluent	530 (26.9%)	75 (3.8%)

Table 2: Partial hypothesis fluency judgements.

to have improved the fluency of the low confidence region more than twice as often as a fluent hypothesis was made disfluent.

Some examples of improved fluency are shown in Figure 4. Although both the ML and unconstrained LMBR hypotheses might satisfy adequacy, they lack the fluency of the LMBR+CC hypotheses generated using monolingual fluency constraints.

9 Summary and Discussion

We have described a general framework for improving SMT fluency. Decoupling the hypothesis space from the evidence space allows for much greater flexibility in lattice MBR search.

We have shown that high path posterior probability n-grams in the ML translation can be used to guide the segmentation of a lattice into regions of high and low confidence. Segmenting the lattice simplifies the process of refining the hypothesis space since low confidence regions can be refined in the context of their high confidence neighbours. This can be done independently before reassembling the refined regions. Lattice segmentation facilitates the application of post-processing and rescoring techniques targeted to address particular deficiencies in ML decoding.

The techniques we presented are related to consensus decoding and system combination for SMT (Matusov et al., 2006; Sim et al., 2007), and to segmental MBR for automatic speech recognition (Goel et al., 2004). Mohit et al. (2009) describe an alternative approach to improving specific portions of translation hypotheses. They use an SVM classifier to identify a single phrase in each source language sentence that is “difficult to translate”; such phrases are then translated using an adapted language model estimated from parallel data. In contrast to their approach, our approach is able to exploit large collections of monolingual data to refine multiple low confidence regions using posterior probabilities obtained from a high-quality evidence space of first-pass translations.

Testset	Sentences	Reachability
tune	2075	15%
test	2040	14%
nw08	813	11%
ng08	547	9%

Table 3: Arabic→English reference reachability.

We applied hypothesis space constraints based on monolingual coverage to low confidence regions resulting in improved fluency with no real degradation in BLEU score relative to unconstrained LMBR decoding. This approach is limited by the coverage of sublattices using monolingual text. We expect this to improve with larger text collections or in tightly focused scenarios where in-domain text is less diverse.

However, fluency will be best improved by integrating more sophisticated natural language generation. NLG systems capable of generating sentence fragments in context can be incorporated directly into this framework. If the MBR hypothesis space $\mathcal{H}$ contains a generated hypothesis $\bar{E}$ for which $P(F|\bar{E}) = 0$, $\bar{E}$ could still be produced as a translation, since it can be ‘voted for’ by nearby hypotheses produced by the underlying system.

Table 3 shows the proportion of NIST testset sentences that can be aligned to any of the reference translations using our high quality baseline hierarchical decoder with a powerful grammar. The low level of reachability suggests that NLG may be required to achieve high levels of translation quality and fluency. Other rescoring approaches (Kumar et al., 2009; Li et al., 2009) may also benefit from NLG when the baseline is incapable of generating the reference.

We note that our approach could also be used to improve the fluency of ASR, OCR and other language processing tasks where the goal is to produce fluent natural language output.

Acknowledgments

We would like to thank Matt Gibson and the human judges who participated in the evaluation. This work was supported in part under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022 and the European Union Seventh Framework Programme (FP7-ICT-2009-4) under Grant Agreement No. 247762.

References

Allauzen, Cyril, Mehryar Mohri, and Brian Roark. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of ACL 2003*.

Blackwood, Graeme, Adrià de Gispert, and William Byrne. 2010. Efficient path counting transducers for minimum Bayes-risk decoding of statistical machine translation lattices. In *Proceedings of ACL 2010*.

Brants, Thorsten, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the EMNLP 2007*.

Brown, Peter F., John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *WMT 2009*.

Chase, Lin Lawrance. 1997. Error-responsive feedback mechanisms for speech recognizers, Ph.D. Thesis, Carnegie Mellon University.

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

DeNero, John, David Chiang, and Kevin Knight. 2009. Fast consensus decoding over translation forests. In *Proceedings of ACL-IJCNLP 2009*.

Goel, V., S. Kumar, and W. Byrne. 2004. Segmental minimum Bayes-risk decoding for automatic speech recognition. *IEEE Transactions on Speech and Audio Processing*, 12:234–249.

Iglesias, Gonzalo, Adrià de Gispert, Eduardo R. Banga, and William Byrne. 2009. Hierarchical phrase-based translation with weighted finite state transducers. In *Proceedings of the 2009 Annual Conference of the NAACL*.

Kneser, R. and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing*.

Knight, K and J Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In *Proceedings of CICLING 2005*.

Knight, K. 2007a. Capturing practical natural language transformations. *Machine Translation*, 21(2).

Knight, Kevin. 2007b. Automatic language translation generation help needs badly. In *MT Summit XI Workshop on Using Corpora for NLG: Keynote Address*.

Kumar, Shankar and William Byrne. 2004. Minimum Bayes-risk decoding for statistical machine translation. In *NAACL 2004*.

Kumar, Shankar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of ACL-IJCNLP 2009*.

Lavie, Alon and Michael J. Denkowski. 2009. The METEOR metric for automatic evaluation of machine translation. *Machine Translation Journal*.

Li, Zhifei, Jason Eisner, and Sanjeev Khudanpur. 2009. Variational decoding for statistical machine translation. In *Proceedings of ACL-IJCNLP 2009*.

Ma, Xiaoyi and Christopher Cieri. 2006. Corpus support for machine translation at LDC. In *LREC 2006*.

Matusov, Evgeny, Nicola Ueffing, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *11th Conference of the EACL*.

Mohit, B., F. Liberato, and R. Hwa. 2009. Language model adaptation for difficult-to-translate phrases. In *Proceedings of the 13th Annual Conference of the EAMT*.

Mohri, Mehryar, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. In *CSL*, volume 16, pages 69–88.

Oberlander, Jon and Chris Brew. 2000. Stochastic text generation. In *Philosophical Transactions of the Royal Society*.

Och, F., D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, and D. Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of the HLT Conference of the NAACL*.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL 2002*.

Sim, K.-C., W. Byrne, M. Gales, H. Sahbi, and P.C. Woodland. 2007. Consensus network decoding for statistical machine translation system combination. In *ICASSP 2007*.

Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, , and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*.

Tromble, Roy, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice minimum Bayes-risk decoding for statistical machine translation. In *Proceedings of the 2008 Conference on EMNLP*.

Ueffing, Nicola and Hermann Ney. 2007. Word-level confidence estimation for machine translation. *Computational Linguistics*, 33(1):9–40.

Vilar, D, G Leusch, H Ney, and R Banchs. 2007. Human evaluation of machine translation through binary system comparisons. In *Proceedings of WMT 2007*.

Zens, Richard and Hermann Ney. 2006. N -gram posterior probabilities for statistical machine translation. In *Proceedings of WMT 2006*.

Self-Annotation for Fine-Grained Geospatial Relation Extraction

Andre Blessing **Hinrich Schütze**
Institute for Natural Language Processing
Universität Stuttgart
ner@ifnlp.org

Abstract

A great deal of information on the Web is represented in both textual and structured form. The structured form is machine-readable and can be used to augment the textual data. We call this augmentation – the annotation of texts with relations that are included in the structured data – *self-annotation*. In this paper, we introduce self-annotation as a new supervised learning approach for developing and implementing a system that extracts fine-grained relations between entities. The main benefit of self-annotation is that it does not require manual labeling. The input of the learned model is a representation of the free text, its output structured relations. Thus, the model, once learned, can be applied to any arbitrary free text. We describe the challenges for the self-annotation process and give results for a sample relation extraction system. To deal with the challenge of fine-grained relations, we implement and evaluate both shallow and deep linguistic analysis, focusing on German.

1 Introduction

In the last years, information extraction has become more important in domains like context-aware systems (e.g. Nexus (Dürr et al., 2004)) that need a rich knowledge base to make the right decisions in different user contexts. Geospatial data are one of the key features in such systems and need to be represented on different levels of detail. Data providers do not cover all these levels completely. To overcome this problem, *fine-grained* information extraction (IE) methods can be used to acquire the missing knowledge. We define fine-grained IE as methods that recognize entities at a finer grain than standard categories like person, location, and organization. Furthermore, the quality of the data in context-aware systems plays an important role and updates by an information extraction component can increase the overall user acceptance.

For both issues an information extraction system is required that can handle *fine-grained relations*, e.g., “X is a suburb of Y” or “the river X is a tributary of Y” – as opposed to simple containment. The World Wide Web offers a wealth of information about geospatial data and can be used as source for the extraction task. The extraction component can be seen as a kind of sensor that we call *text senor* (Blessing et al., 2006).

In this paper, we address the problem of developing a flexible system for the acquisition of relations between entities that meets the above desiderata. We concentrate on *geospatial* entities on a fine-grained level although the approach is in principle applicable to any domain. We use a supervised machine learning approach, including several features on different linguistic levels, to build our system. Such a system highly depends on the quality and amount of labeled data in the training phase. The main contribution of this paper is the introduction of self-annotation, a novel approach that allows us to eliminate manual labeling (although training set creation also involves costs other than labeling). Self-annotation is based on the fact that Word Wide Web sites like Wikipedia include, in addition to unstructured text, structured data. We use structured data sources to automatically annotate unstructured texts. In this paper, we use German Wikipedia data because it is a good source for the information required for our context-aware system and show that a system created without manual labeling has good performance.

Our trained model only uses text, not the structured data (or any other markup) of the input documents. This means that we can train an information extractor on Wikipedia and then apply it to any text, regardless of whether this text also contains structured information.

In the first part of this paper, we discuss the challenges of self-annotation including some heuristics which can easily be adapted to different relation types. We then describe the architecture of the extraction system. The components we develop are based on the UIMA (Unstructured Information Management Architecture) framework (Hahn et al., 2008) and include two linguistic engines (OpenNLP[1], FSPar). The extraction task is performed by a supervised classifier; this classifier is also implemented as a UIMA component and uses the ClearTK framework. We evaluate our approach on two types of fine-grained relations.

2 Related work

Jiang (2009) also addresses the issue of supervised relation extraction when no large manually labeled data set is available. They use only a few seed instances of the target relation type to train a supervised relation extraction system. However, they use multi-task transfer learning including a large amount of labeled instances of other relation types for training their system. In contrast, our work eliminates manual labeling by using structured data to annotate the relations.

Wu and Weld (2007) extract facts from infoboxes and link them with their corresponding representation in the text. They discuss several issues that occur when using infoboxes as a knowledge base, in particular, (i) the fact that infoboxes are incomplete; and (ii) *schema drift*. Schema drift occurs when authors over time use different attribute names to model facts or the same attributes are used to model different facts. So the semantics of the infoboxes changes slightly and introduces noise into the structured information. Their work differs from self-annotation in that they are not interested in the creation of self-annotated corpora that can be used as training data for other tasks. Their goal is to develop methods that make infoboxes more consistent.

Zhang and Iria (2009) use a novel entity extraction method to automatically generate gazetteers from seed lists using Wikipedia as knowledge source. In contrast to our work they need structured data for the extraction while our system focuses on the extraction of information from unstructured text. Methods that are applicable to any unstructured text (not just the text in the Wikipedia) are needed to increase coverage beyond the limited number of instances covered in Wikipedia.

Nothman et al. (2009) also annotate Wikipedia's unstructured text using structured data. The type of structured data they use is hyperlinking (as opposed to infoboxes) and they use it to derive a labeled named entity corpus. They show that the quality of the annotation is comparable to other manually labeled named entity recognition gold standards. We interpret their results as evidence that self-annotation can be used to create high quality gold standards.

3 Task definition

In this section, we describe the annotation task; give a definition of the relation types covered in this paper; and introduce the extraction model.

We focus on binary relations between two relation arguments occurring in the same sentence. To simplify the self-annotation process we restrict the first argument of the relation to the main entity of the Wikipedia article. As we are building text sensors for a context aware system, relations between geospatial entities are of interest. Thus we consider only relations that use a geospatial named entity as second argument.

We create the training set by automatically identifying all correct binary relations in the text. To this end, we extract the relations from the structured part of the Wikipedia, the infoboxes. Then we automatically find the corresponding

[1] http://opennlp.sourceforge.net/

sentences in the text and annotate the relations (see section 4). All other not yet marked binary relations between the main entity and geospatial entities are annotated as negative samples. The result of this step is a self-annotated training set.

In the second step of our task, the self-annotated training set is used to train the extraction model. The model only takes textual features as input and can be applied to any free text.

3.1 Classification task and relations used

Our relation extraction task is modeled as a classification task which considers a pair of named entities and decides whether they occur in the requested relation or not. The classifier uses extracted features for this decision. Features belong to three different classes. The first class contains *token-based features* and their linguistic labels like part-of-speech, lemma, stem. In the second class, we have *chunks* that aggregate one or more tokens into complex units. *Dependency relations* between the tokens are represented in the third class.

Our classifier is applicable to a wide spectrum of geospatial relation types. For the purposes of a focused evaluation, we selected two relations. The first type contains rivers and the bodies of water into which they flow. We call it *river-bodyOfWater* relation. Our second type is composed of relations between towns and the corresponding suburb. We call this *town-suburb* relation.

3.2 Wikipedia as resource

Wikipedia satisfies all corpus requirements for our task. It contains a lot of knowledge about geospatial data with unstructured (textual) and structured information. We consider only German Wikipedia articles because our target application is a German context aware system. In relation extraction for German, we arguably face more challenges – e.g., more complex morphology and freer word order – than we would in English.

For this work we consider only a subset of the German Wikipedia. We use all articles that belong to the following categories: Rivers by country, Mountains by country, Valleys by country, Islands by country, Mountain passes by country, Forests by country and Settlements by country.

For the annotation task we use the structural content of Wikipedia articles. Most articles belonging to the same categories use similar templates to represent structured information. One type of template is the infobox, which contains pairs of attributes and their values. These attribute-value pairs specify a wide range of geospatial relation types including fine-grained relations. In this work we consider only the infobox data and the article names from the structured data.

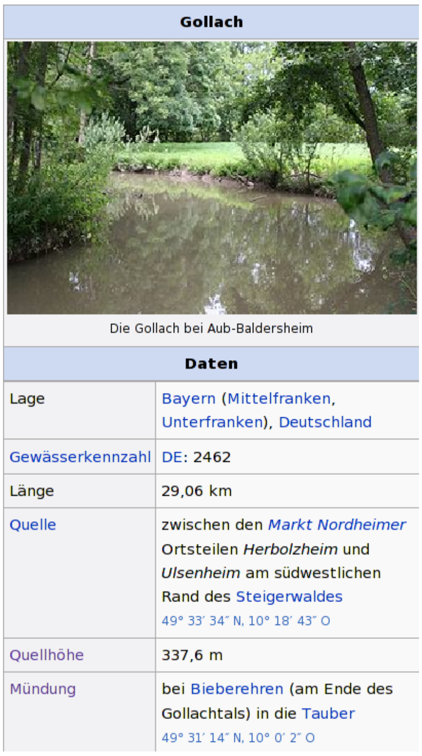
For context-aware systems fine-grained relation types are particularly relevant. Such relations are not represented in resources like DBPedia (Auer et al., 2007) or Yago (Suchanek et al., 2007) although they also consist of infobox data. Hence, we have to build our own extraction component (see section 5.2) when using infoboxes.

4 Self-Annotation

Self-annotation is a two-fold task. First, the structured data, in our case the infoboxes of Wikipedia articles, must be analyzed to get all relevant attribute-value pairs. Then all relevant geospatial entities are marked and extracted. In a second step these entities must be matched with the unstructured data.

In most cases, the extraction of the named entities that correspond to the required relations is trivial because the values in the infoboxes consist only of one single entity or one single link. But in some cases the values contain mixed content which can include links, entities and even free text. In order to find an accurate extraction method for those values we have developed several heuristics. See section 5.2 for discussion.

The second task links the extracted structured data to tokens in the textual data. Pattern based string matching methods are not sufficient to identify all relations in the text. In many cases, morphological rules need to be applied to identify the entities in the text. In other cases, the preprocessed text must be retokenized because the borders of multi-word expressions are not consistent with the extracted names in step one. One other issue is that some named entities are a subset of other named entities (*Lonau* vs. *kleine Lonau*;



Gollach

Die Gollach bei Aub-Baldersheim

Daten

Lage	Bayern (Mittelfranken, Unterfranken), Deutschland
Gewässerkennzahl	DE: 2462
Länge	29,06 km
Quelle	zwischen den *Markt Nordheimer* Ortsteilen *Herbolzheim* und *Ulsenheim* am südwestlichen Rand des Steigerwaldes 49° 33′ 34″ N, 10° 18′ 43″ O
Quellhöhe	337,6 m
Mündung	bei Bieberehren (am Ende des Gollachtals) in die Tauber 49° 31′ 14″ N, 10° 0′ 2″ O

Figure 1: Infobox of the German Wikipedia article about *Gollach*.

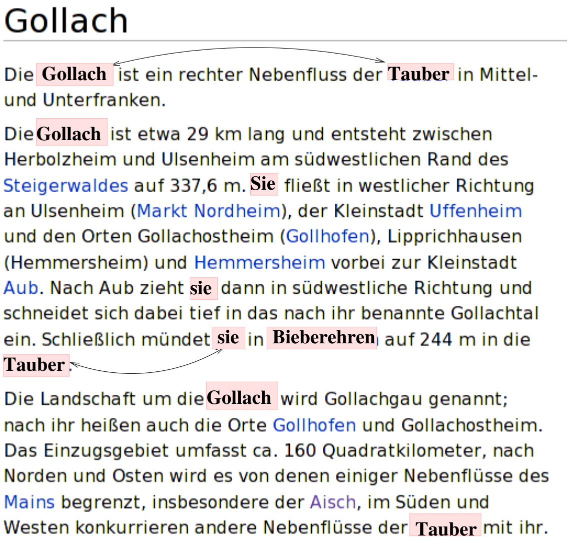
similar to *York* vs. *New York*). We have to use a longest match strategy to avoid such overlapping annotations.

The main goal of the self-annotation task is to reach the highest possible annotation quality. Thus, only complete extracted relations are used for the annotation process while incomplete data are excluded from the training set. This procedure reduces the noise in the labeled data.

4.1 Example

We use the river-bodyOfWater relation between the two rivers *Gollach* and *Tauber* to describe the self-annotation steps.

Figure 1 depicts a part of the infobox for the German Wikipedia article about the river *Gollach*. For this relation the attribute Mündung 'mouth' is relevant. The value contains unstructured information (i.e., text, e.g. *bei 'at' Bieberehren*) and structured information (the link from *Bieberehren* to its Wikipedia page). The relation we want to extract is that the river *Gollach* flows into the river *Tauber*.



Gollach

Die **Gollach** ist ein rechter Nebenfluss der **Tauber** in Mittel- und Unterfranken.

Die **Gollach** ist etwa 29 km lang und entsteht zwischen Herbolzheim und Ulsenheim am südwestlichen Rand des Steigerwaldes auf 337,6 m. **Sie** fließt in westlicher Richtung an Ulsenheim (Markt Nordheim), der Kleinstadt Uffenheim und den Orten Gollachostheim (Gollhofen), Lipprichhausen (Hemmersheim) und Hemmersheim vorbei zur Kleinstadt Aub. Nach Aub zieht **sie** dann in südwestliche Richtung und schneidet sich dabei tief in das nach ihr benannte Gollachtal ein. Schließlich mündet **sie** in **Bieberehren** auf 244 m in die **Tauber**.

Die Landschaft um die **Gollach** wird Gollachgau genannt; nach ihr heißen auch die Orte Gollhofen und Gollachostheim. Das Einzugsgebiet umfasst ca. 160 Quadratkilometer, nach Norden und Osten wird es von denen einiger Nebenflüsse des Mains begrenzt, insbesondere der Aisch, im Süden und Westen konkurrieren andere Nebenflüsse der **Tauber** mit ihr.

Figure 2: Textual content of the German Wikipedia article about *Gollach*. All named entities which are relevant for the river-bodyOfWater relation are highlighted. This article contains two instances for the relation between *Gollach* and *Tauber*.

Figure 2 shows the textual content of the *Gollach* article. We have highlighted all relevant named entities for the self-annotation process. This includes the name of the article and instances of the pronoun *sie* referring to *Gollach*. Our matching algorithm identifies two sentences as positive samples for the relation between *Gollach* and *Tauber*:

- (i) Die *Gollach* ist ein rechter Nebenfluss der *Tauber* in Mittel- und Unterfranken. (The *Gollach* is a right tributary of the *Tauber* in Middle and Lower Franconia.)
- (ii) Schließlich mündet *sie* in Bieberehren auf 244 m in die *Tauber*. (Finally, *it* discharges in Bieberehren at 244 m above MSL into the *Tauber*.)

5 Processing

In this section we describe how the self-annotation method and relation extraction is implemented. First we introduce the interaction with the Wikipedia resource to acquire the structured and unstructured information for the processing pipeline. Second we present the components of the UIMA pipeline which are used for the relation extraction task.

5.1 Wikipedia interaction

We use the JWPL API (Zesch et al., 2008) to pre-process the Wikipedia data. This interface provides functions to extract structured and unstructured information from Wikipedia. However, many Wikipedia articles do not adhere to valid Wikipedia syntax (missing closing brackets etc.). The API also does not correctly handle all Wikipedia syntax constructions. We therefore have enhanced the API for our extraction task to get high quality data for German Wikipedia articles.

5.2 Infobox extraction

As discussed in section 4 infoboxes are the key resource for the self-annotation step. However the processing of infoboxes that include attribute-value pairs with mixed content is not trivial.

For each new relation type an initial manual effort is required. However, in comparison to the complete annotation of a training corpus, this effort is small. First the attributes used in the infoboxes of the Wikipedia articles relevant for a specific relation have to be analyzed. The results of this analysis simplify the choice of the correct attributes. Next, the used values of these attributes must be investigated. If they contain only single entries (links or named entities) the extraction is trivial. However, if they consist of mixed content (see section 4.1) then specific extraction methods have to be applied. We investigated different heuristics for the self-annotation process to get a method that can easily be adapted to new relation types.

Our first heuristic includes a set of rules specifying the extraction of the values from the infoboxes. This heuristic gives an insufficient basis for the self-annotation task because the rich morphology and free word order in German can not be modeled with simple rules. Moreover, hand-crafted rules are arguably not as robust and maintainable as a statistical classifier trained on self-annotated training material.

Our second heuristic is a three step process. In step one we collect all links in the mixed content and replace them by a placeholder. In the second step we tag the remaining content with the OpenNLP tokenizer to get all named entities. Both collected lists are then looked up in a lexicon that contains named entities and the corresponding geospatial classes. This process requires a normalization procedure that includes the application of morphological methods. The second method can be easily adapted to new relation types.

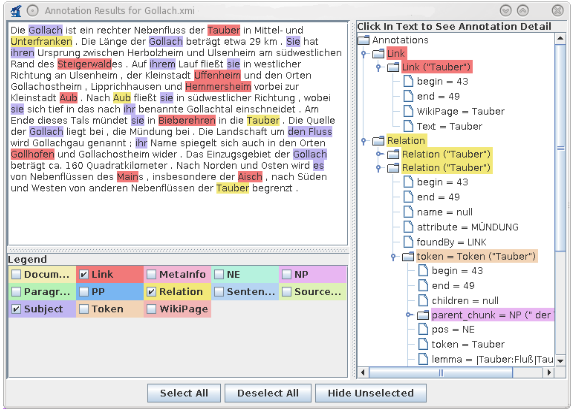
5.3 UIMA

The self-annotated corpora are processed by several components of the UIMA (Müller et al., 2008) pipeline. The advantage of exchangeable collection readers is that they seamlessly handle structured and unstructured data. Another advantage of using UIMA is the possibility to share components with other research groups. We can easily exchange different components, like the usage of the commonly known OpenNLP processing tools or the FSPar NLP engine (Schiehlen, 2003) (which includes the TreeTagger (Schmid, 1995)). This allows us to experiment with different approaches, e.g., shallow vs. deep analysis. The components we use provide linguistic analysis on different levels: tokens, morphology, part of speech (POS), chunking and partial dependency analysis. Figure 4 shows the results after the linguistic processing of our sample sentence. For this work only a few annotations are wrapped as UIMA types: token (incl. lemma, POS), multiword, sentence, NP, PP and dependency relations (labeled edges between tokens). We will introduce our machine learning component in section 5.5. Finally, the CAS consumers allow us to store extracted facts in a context model.

Figure 3 shows the article about *Gollach* after linguistic processing. In the legend all annotated categories are listed. We highlighted all marked relations, all references to the article name (referred to as subject in the figure) and links. After selection of the *Tauber* relation, all annotations for this token are listed in the right panel.

5.4 Coreference resolution

Using anaphora to refer to the main entity is a common practice of the authors of Wikipedia ar-



Die Gollach ist ein rechter Nebenfluss der Tauber in Mittel- und Unterfranken . Die Länge der Gollach beträgt etwa 29 km . Sie hat ihren Ursprung zwischen Herbolzheim und Ulsenheim am südwestlichen Rand des Steigerwaldes . Auf ihrem Lauf fließt sie in westlicher Richtung an Ulsenheim , der Kleinstadt Uffenheim und den Orten Gollachostheim , Lipprichhausen und Hemmersheim vorbei zur Kleinstadt Aub . Nach Aub fließt sie in südwestlicher Richtung , wobei sie sich tief in das nach ihr benannte Gollachtal einschneidet . Am Ende dieses Tals mündet sie in Bieberehren in die Tauber . Die Quelle der Gollach liegt bei , die Mündung bei . Die Landschaft um den Fluss wird Gollachgau genannt ; ihr Name spiegelt sich auch in den Orten Gollhofen und Gollachostheim wider . Das Einzugsgebiet der Gollach beträgt ca. 160 Quadratkilometer . Nach Norden und Osten wird es von Nebenflüssen des Mains , insbesondere der Aisch , nach Süden und Westen von anderen Nebenflüssen der Tauber begrenzt .

Legend

□ Docum...	☑ Link	□ MetaInfo	□ NE	□ NP
□ Paragr...	□ PP	☑ Relation	□ Senten...	□ Source...
☑ Subject	□ Token	□ WikiPage		

Click In Text to See Annotation Detail

```
Annotations
  Link
    Link ("Tauber")
      begin = 43
      end = 49
      WikiPage = Tauber
      Text = Tauber
  Relation
    Relation ("Tauber")
    Relation ("Tauber")
      begin = 43
      end = 49
      name = null
      attribute = MÜNDUNG
      foundBy = LINK
      token = Token ("Tauber")
        begin = 43
        end = 49
        children = null
        parent_chunk = NP (" der
        pos = NE
        token = Tauber
        lemma = |Tauber:Fluß|Tau
```

Figure 3: Screenshot of the UIMA Annotation-Viewer.

ticles. Coreference resolution is therefore necessary for our annotation task. A shallow linguistic analysis showed that the writing style is similar throughout Wikipedia articles. Based on this observation, we empirically investigated some geospatial articles and came to the conclusion that a simple heuristic is sufficient for our coreference resolution problem. In almost all articles, pronouns refer to the main entity of the article. In addition we include some additional rules to be able to establish coreference of markables such as *der Fluss* 'the river' or *der Bach* 'the creek' with the main entity.

5.5 Supervised relation extraction

We use the ClearTK (Ogren et al., 2008) toolkit, which is also an UIMA component, for the relation extraction task. It contains wrappers for different machine learning suites. Our initial experiments showed that the MaximumEntropy classifier achieved the best results for our classification task. The toolkit provides additional extensible feature methods. Because we view self-annotation and fine-grained named entity recognition as our main contributions, not feature selection, we only give a brief overview of the features we use.

F1 is a window based bag-of-words feature (window size = 3). It considers lemma and part-of-speech tag of the tokens. F2 is a phrase based extractor that uses the parent phrase of both entities (max 2 levels). F3 is a representation of all possible dependency paths between the article's main entity and a target entity, where each path is represented as a feature vector. In most cases, more than one path is returned by the partial dependency parser (which makes no disambiguation decisions) and included in the feature representation. Figure 4 depicts the dependency parser output of our sample sentence. Each pair of square and circle with the same number corresponds to one dependency. These different possible dependency combinations give rise to 8 possible paths between the relation entities *Tauber* and *sie 'she'* although our example sentence is a very simple sentence.

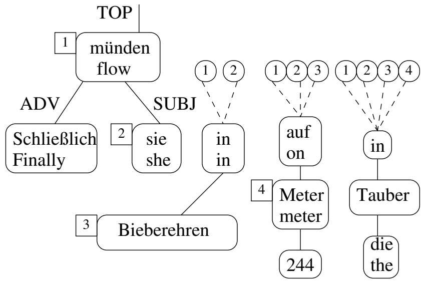


Figure 4: Dependency parser output of the FSPar framework.

	linguistic effort	description
F1	pos-tagging	window size 3, LEMMA
F2	chunk-parse	parent chunks
F3	dependency-parse	dependency paths betw. NEs

Table 1: List of feature types

6 Evaluation

We evaluate the system in two experiments. The first considers the relation between suburbs and their parent towns. In the second experiment the river-bodyOfWater relation is extracted. The experiments are based on the previously described extracted Wikipedia corpus. For each experiment a new self-annotated corpus is created that is split into three parts. The first part (60%) is used as training corpus. The second part (20%) is used as development corpus. The remaining 20% is used for the final evaluation and was not inspected while we were developing the extraction algorithms.

6.1 Metric used

Our gold standard includes all relations of each article. Our metric works on the level of type and is independent of how often the same relation occurs in the article. The metric counts a relation as true positive (TP) if the system extracted it at least once. If the relation was not found by the system a false negative (FN) is counted. A false positive (FP) is given if the system extracts a relation between two entities that is not part of the (infobox-derived) gold standard for the article. All three measures are used to calculate precision ($P = \frac{TP}{TP+FP}$), recall ($R = \frac{TP}{TP+FN}$), and F_1-score ($F_1 = 2\frac{P*R}{P+R}$).

6.2 Town-suburb extraction

The town-suburb extractor uses one attribute of the infobox to identify the town-suburb relation. There is no schema drift in the infobox data and the values contain only links. Therefore the self-annotation works almost perfectly. The only exceptions are articles without an infobox which cannot be used for training. However, this is not a real issue because the amount of remaining data is sufficient: 9000 articles can be used for this task. The results in table 2 show that the classifier that uses F1, F2 and F3 (that is, including the dependency features) performs best.

engine	features	F_1	recall	precision
FSPar	F1	64.9	79.0%	55.7%
FSPar	F1, F2	89.6	90.2%	89.5%
FSPar	F1, F2, F3	98.3	98.8%	97.8%

Table 2: Results of different feature combinations on the test set for town-suburb relation

6.3 River-bodyOfWater extraction

For the extraction of the river-bodyOfWater relation the infobox processing is more difficult. We have to handle more attributes because there is schema drift between the different users. It is hence necessary to merge information coming from different attribute values. The other difficulty is the usage of mixed contents in the values. Another main difference to the town-suburb relation is that the river-bodyOfWater relation is often not mentioned in the first sentence (which usually gives a short definition about the the main entity). Thus, the self-annotation method has to deal with the more complex sentences that are common later in the article. This also contributes to a more challenging extraction task.

Our river-bodyOfWater relation corpus consists of 3000 self-annotated articles.

Table 3 shows the performance of the extractor using two different linguistic components as described in section 5.3. As in the case of town-suburb extraction the classifier that uses all features, including dependency features, performs best.

engine	features	F_1	recall	precision
FSPar	F1	51.8%	56.6%	47.8%
FSPar	F1,F2	72.1%	68.9%	75.7%
FSPar	F1,F2,F3	78.3%	74.1%	83.0%
OpenNLP	F1	48.0%	62.8%	38.8%
OpenNLP	F1,F2	73.3%	71.7%	74.7%

Table 3: Results of different feature combinations on the test set for river-bodyOfWater extraction

6.4 Evaluation of self-annotation

To evaluate the quality of self-annotation, we randomly selected one set of 100 self-annotated articles from each data set and labeled these sets manually. These annotations are used to calculate the inter-annotator agreement between the human annotated and machine annotated instances. We use Cohen's κ as measure and get a result of 1.00 for the town-suburb relation. For the river-bodyOfWater relation we got a κ-value of 0.79, which also indicates good agreement.

We also use a gazetteer to evaluate the quality of all town-suburb relations that were extracted for our self-annotated training set. The accuracy is nearly perfect (only one single error), which is good evidence for the high quality of Wikipedia.

Required size of self-annotated training set. The performance of a supervised system depends on the size of the training data. In the self-annotation step a minimum of instances has to be annotated, but it is not necessary to self-annotate all available articles.

We reduced the number of articles used in the training size to test this hypothesis. Reducing the entire training set of 9000 (respectively, 3000) self-annotated articles to 1000 reduces F1 by 2.0% for town-suburb and by 2.4% for river-bodyOfWater; a reduction to 100 reduces F1 by 8.5% for town-suburb and by 9.3% for river-bodyOfWater (compared to the 9000/3000 baseline).

7 Discussion

Wu and Weld (2007) observed schema drift in their work: Wikipedia authors do not not use infobox attributes in a consistent manner. However, we did not find schema drift to be a large problem in our experiments. The variation we found can easily be handled with a small number of rules. This can be due to the fact that the quality of Wikipedia articles improved a lot in the last years through the introduction of automatic maintenance tools like bots[2]. Nevertheless, the development of self-annotation for a new relation type requires some manual work. The developer has to check the quality of the extraction relations in the infoboxes. This can lead to some additional adaptation work for the used attributes such as merging or creating rules. However, a perfect coverage is not required because the extraction system is only used for training purposes; we only need to find a sufficiently large number of positive training instances and do not require exhaustive labeling of all articles.

It is important to note that considering partially found relations as negative samples has to be avoided. Wrong negative samples have a generally unwanted impact on the performance of the learned extraction model. A developer has to be aware of this fact. In one experiment, the learned classifiers were applied to the training data and returned a number of false positive results – 40 in case of the river-bodyOfWater relation. 31 of these errors were not actual errors because the self-annotation missed some true instances. Nevertheless, the trained model recognizes these samples as correct; this could perhaps be used to further improve the quality of self-annotation.

Manually labeled data also includes noise and the benefit of self-annotation is substantial when the aim is to build a fine-grained relation extraction system in a fast and cheap way.

The difference of the results between OpenNLP and FSPar engines are smaller than expected. Although sentence splitting is poorly done by OpenNLP the effect on the extraction result is rather low. Another crucial point is that the lexicon-based named entity recognizer of the FSPar engine that was optimized for named entities used in Wikipedia has no significant impact on the overall performance. Thus, a basic set of NLP components with moderate error rates may be sufficient for effective self-annotation.

8 Conclusion

This paper described a new approach to developing and implementing a complete system to extract fine-grained geospatial relations by using a supervised machine learning approach without expensive manual labeling. Using self-annotation, systems can be rapidly developed and adapted for new relations without expensive manual annotation. Only some manual work has to be done to find the right attributes in the infoboxes. The matching process between infoboxes and text is not in all cases trivial and for some attributes additional rules have to be modeled.

9 Acknowledgment

This project was funded by DFG as part of Nexus (Collaborative Research Centre, SFB 627).

References

Auer, Sören, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *In 6th Intl Semantic Web Conference, Busan, Korea*, pages 11–15. Springer.

Blessing, Andre, Stefan Klatt, Daniela Nicklas, Steffen Volz, and Hinrich Schütze. 2006. Language-derived information and context models. In *Proceedings of 3rd IEEE PerCom Workshop on Context Modeling and Reasoning (CoMoRea) (at 4th IEEE International Conference on Pervasive Computing and Communication (PerCom'06)).*

Dürr, Frank, Nicola Hönle, Daniela Nicklas, Christian Becker, and Kurt Rothermel. 2004. Nexus–a platform for context-aware applications. In Roth, Jörg,

[2] See en.wikipedia.org/wiki/Wikipedia:Bots. The edit history of many articles shows that there is a lot of automatic maintenance by bots to avoid schema drift.

editor, *1. Fachgespräch Ortsbezogene Anwendungen und Dienste der GI-Fachgruppe KuVS*, pages 15–18, Hagen, Juni. Informatik-Bericht der FernUniversität Hagen.

Hahn, Udo, Ekaterina Buyko, Rico Landefeld, Matthias Mühlhausen, Michael Poprat, Katrin Tomanek, and Joachim Wermter. 2008. An overview of JCoRe, the JULIE lab UIMA component repository. In *Proceedings of the LREC'08 Workshop 'Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP'*, Marrakech, Morocco, May.

Jiang, Jing. 2009. Multi-task transfer learning for weakly-supervised relation extraction. In *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*, pages 1012–1020, Morristown, NJ, USA. Association for Computational Linguistics.

Müller, Christof, Torsten Zesch, Mark-Christoph Müller, Delphine Bernhard, Kateryna Ignatova, Iryna Gurevych, and Max Mühlhäuser. 2008. Flexible uima components for information retrieval research. In *Proceedings of the LREC 2008 Workshop 'Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP',* Marrakech, Morocco, May 31, 2008. 24–27.

Nothman, Joel, Tara Murphy, and James R. Curran. 2009. Analysing wikipedia and gold-standard corpora for ner training. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 612–620, Morristown, NJ, USA. Association for Computational Linguistics.

Ogren, Philip V., Philipp G. Wetzler, and Steven Bethard. 2008. Cleartk: A uima toolkit for statistical natural language processing. In *UIMA for NLP workshop at Language Resources and Evaluation Conference (LREC).*

Schiehlen, Michael. 2003. Combining deep and shallow approaches in parsing german. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 112–119, Morristown, NJ, USA. Association for Computational Linguistics.

Schmid, Helmut. 1995. Improvements in part-of-speech tagging with an application to german. In *In Proceedings of the ACL SIGDAT-Workshop*, pages 47–50.

Suchanek, Fabian M., Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA. ACM Press.

Wu, Fei and Daniel S. Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, pages 41–50.

Zesch, Torsten, Christof Müller, and Iryna Gurevych. 2008. Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary. In *Proceedings of the Conference on Language Resources and Evaluation (LREC).*

Zhang, Ziqi and José Iria. 2009. A novel approach to automatic gazetteer generation using wikipedia. In *People's Web '09: Proceedings of the 2009 Workshop on The People's Web Meets NLP*, pages 1–9, Morristown, NJ, USA. Association for Computational Linguistics.

Very High Accuracy and Fast Dependency Parsing is not a Contradiction

Bernd Bohnet
University of Stuttgart
Institut für Maschinelle Sprachverarbeitung
bernd.bohnet@ims.uni-stuttgart.de

Abstract

In addition to a high accuracy, short parsing and training times are the most important properties of a parser. However, parsing and training times are still relatively long. To determine why, we analyzed the time usage of a dependency parser. We illustrate that the mapping of the features onto their weights in the support vector machine is the major factor in time complexity. To resolve this problem, we implemented the passive-aggressive perceptron algorithm as a Hash Kernel. The Hash Kernel substantially improves the parsing times and takes into account the features of negative examples built during the training. This has lead to a higher accuracy. We could further increase the parsing and training speed with a parallel feature extraction and a parallel parsing algorithm. We are convinced that the Hash Kernel and the parallelization can be applied successful to other NLP applications as well such as transition based dependency parsers, phrase structrue parsers, and machine translation.

1 Introduction

Highly accurate dependency parsers have high demands on resources and long parsing times. The training of a parser frequently takes several days and the parsing of a sentence can take on average up to a minute. The parsing time usage is important for many applications. For instance, dialog systems only have a few hundred milliseconds to analyze a sentence and machine translation systems, have to consider in that time some thousand translation alternatives for the translation of a sentence.

Parsing and training times can be improved by methods that maintain the accuracy level, or methods that trade accuracy against better parsing times. Software developers and researchers are usually unwilling to reduce the quality of their applications. Consequently, we have to consider at first methods to improve a parser, which do not involve an accuracy loss, such as faster algorithms, faster implementation of algorithms, parallel algorithms that use several CPU cores, and feature selection that eliminates the features that do not improve accuracy.

We employ, as a basis for our parser, the second order maximum spanning tree dependency parsing algorithm of Carreras (2007). This algorithm frequently reaches very good, or even the best labeled attachment scores, and was one of the most used parsing algorithms in the shared task 2009 of the Conference on Natural Language Learning (CoNLL) (Hajič et al., 2009). We combined this parsing algorithm with the passive-aggressive perceptron algorithm (Crammer et al., 2003; McDonald et al., 2005; Crammer et al., 2006). A parser build out of these two algorithms provides a good baseline and starting point to improve upon the parsing and training times.

The rest of the paper is structured as follows. In Section 2, we describe related work. In section 3, we analyze the time usage of the components of

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 89–97,
Beijing, August 2010

the parser. In Section 4, we introduce a new Kernel that resolves some of the bottlenecks and improves the performance. In Section 5, we describe the parallel parsing algorithms which nearly allowed us to divide the parsing times by the number of cores. In Section 6, we determine the optimal setting for the Non-Projective Approximation Algorithm. In Section 7, we conclude with a summary and an outline of further research.

2 Related Work

The two main approaches to dependency parsing are transition based dependency parsing (Nivre, 2003; Yamada and Matsumoto., 2003; Titov and Henderson, 2007) and maximum spanning tree based dependency parsing (Eisner, 1996; Eisner, 2000; McDonald and Pereira, 2006). Transition based parsers typically have a linear or quadratic complexity (Nivre et al., 2004; Attardi, 2006). Nivre (2009) introduced a transition based non-projective parsing algorithm that has a worst case quadratic complexity and an expected linear parsing time. Titov and Henderson (2007) combined a transition based parsing algorithm, which used a beam search with a latent variable machine learning technique.

Maximum spanning tree dependency based parsers decomposes a dependency structure into parts known as “factors”. The factors of the first order maximum spanning tree parsing algorithm are edges consisting of the head, the dependent (child) and the edge label. This algorithm has a quadratic complexity. The second order parsing algorithm of McDonald and Pereira (2006) uses a separate algorithm for edge labeling. This algorithm uses in addition to the first order factors: the edges to those children which are closest to the dependent. The second order algorithm of Carreras (2007) uses in addition to McDonald and Pereira (2006) the child of the dependent occurring in the sentence between the head and the dependent, and the an edge to a grandchild. The edge labeling is an integral part of the algorithm which requires an additional loop over the labels. This algorithm therefore has a complexity of O(n^4). Johansson and Nugues (2008) reduced the needed number of loops over the edge labels by using only the edges that existed in the training corpus for a distinct head and child part-of-speech tag combination.

The transition based parsers have a lower complexity. Nevertheless, the reported run times in the last shared tasks were similar to the maximum spanning tree parsers. For a transition based parser, Gesmundo et al. (2009) reported run times between 2.2 days for English and 4.7 days for Czech for the joint training of syntactic and semantic dependencies. The parsing times were about one word per second, which speeds up quickly with a smaller beam-size, although the accuracy of the parser degrades a bit. Johansson and Nugues (2008) reported training times of 2.4 days for English with the high-order parsing algorithm of Carreras (2007).

3 Analysis of Time Usage

We built a baseline parser to measure the time usage. The baseline parser resembles the architecture of McDonald and Pereira (2006). It consists of the second order parsing algorithm of Carreras (2007), the non-projective approximation algorithm (McDonald and Pereira, 2006), the passive-aggressive support vector machine, and a feature extraction component. The features are listed in Table 4. As in McDonald et al. (2005), the parser stores the features of each training example in a file. In each epoch of the training, the feature file is read, and the weights are calculated and stored in an array. This procedure is up to 5 times faster than computing the features each time anew. But the parser has to maintain large arrays: for the weights of the sentence and the training file. Therefore, the parser needs 3GB of main memory for English and 100GB of disc space for the training file. The parsing time is approximately 20% faster, since some of the values did not have to be recalculated.

Algorithm 1 illustrates the training algorithm in pseudo code. τ is the set of training examples where an example is a pair (x_i, y_i) of a sentence and the corresponding dependency structure. $\overrightarrow{w}$ and $\overrightarrow{v}$ are weight vectors. The first loop extracts features from the sentence x_i and maps the features to numbers. The numbers are grouped into three vectors for the features of all possible edges $\phi_{h,d}$, possible edges in combination with siblings $\phi_{h,d,s}$ and in combination with grandchil-

	t_{e+s}	t_r	t_p	t_a	rest	total	t_e	pars.	train.	sent.	feat.	LAS	UAS
Chinese	4582	748	95	-	3	846	3298	3262	84h	22277	8.76M	76.88	81.27
English	1509	168	12.5	20	1.5	202	1223	1258	38.5h	39279	8.47M	90.14	92.45
German	945	139	7.7	17.8	1.5	166	419	429	26.7h	36020	9.16M	87.64	90.03
Spanish	3329	779	36	-	2	816	2518	2550	16.9h	14329	5.51M	86.02	89.54

Table 1: t_{e+s} is the elapsed time in milliseconds to extract and store the features, t_r to read the features and to calculate the weight arrays, t_p to predict the projective parse tree, t_a to apply the non-projective approximation algorithm, *rest* is the time to conduct the other parts such as the update function, *train.* is the total training time per instance ($t_r + t_p + t_a$+*rest*), and t_e is the elapsed time to extract the features. The next columns illustrate the parsing time in milliseconds per sentence for the test set, training time in hours, the number of sentences in the training set, the total number of features in million, the labeled attachment score of the test set, and the unlabeled attachment score.

Algorithm 1: Training – baseline algorithm

```
$\tau = \{(x_i, y_i)\}_{i=1}^{I}$ // Training data
$\overrightarrow{w} = 0, \overrightarrow{v} = 0$
$\gamma = E * I$ // passive-aggresive update weight
for i = 1 to I
   $t^s_{s+e}$; extract-and-store-features($x_i$); $t^e_{s+e}$;
for n = 1 to $E$ // iteration over the training epochs
   for i = 1 to $I$ // iteration over the training examples
      $k \leftarrow (n-1) * I + i$
      $\gamma = E * I - k + 2$ // passive-aggressive weight
      $t^s_{r,k}$; $A$ = read-features-and-calc-arrays(i,$\overrightarrow{w}$) ; $t^e_{r,k}$
      $t^s_{p,k}$; $y_p$ = predicte-projective-parse-tree($A$);$t^e_{p,k}$
      $t^s_{a,k}$; $y_a$ = non-projective-approx.($y_p$,$A$); $t^e_{a,k}$
      update $\overrightarrow{w}$, $\overrightarrow{v}$ according to $\Delta(y_p, y_i)$ and $\gamma$
$w = v/(E * I)$ // average
```

dren $\phi_{h,d,g}$ where $h, d, g,$ and s are the indexes of the words included in x_i. Finally, the method stores the feature vectors on the hard disc.

The next two loops build the main part of the training algorithm. The outer loop iterates over the number of training epochs, while the inner loop iterates over all training examples. The online training algorithm considers a single training example in each iteration. The first function in the loop reads the features and computes the weights A for the factors in the sentence x_i. A is a set of weight arrays.

$$A = \{\overrightarrow{w} * \overrightarrow{f}_{h,d}, \overrightarrow{w} * \overrightarrow{f}_{h,d,s}, \overrightarrow{w} * \overrightarrow{f}_{h,d,g}\}$$

The parsing algorithm uses the weight arrays to predict a projective dependency structure y_p. The non-projective approximation algorithm has as input the dependency structure and the weight arrays. It rearranges the edges and tries to increase the total score of the dependency structure. This algorithm builds a dependency structure y_a, which might be non-projective. The training algorithm updates $\overrightarrow{w}$ according to the difference between the predicted dependency structures y_a and the reference structure y_i. It updates $\overrightarrow{v}$ as well, whereby the algorithm additionally weights the updates by γ. Since the algorithm decreases γ in each round, the algorithm adapts the weights more aggressively at the beginning (Crammer et al., 2006). After all iterations, the algorithm computes the average of $\overrightarrow{v}$, which reduces the effect of overfitting (Collins, 2002).

We have inserted into the training algorithm functions to measure the start times t^s and the end times t^e for the procedures to compute and store the features, to read the features, to predict the projective parse, and to calculate the non-projective approximation. We calculate the average elapsed time per instance, as the average over all training examples and epochs:

$$t_x = \frac{\sum_{k=1}^{E*I} t^e_{x,k} - t^s_{x,k}}{E*I}.$$

We use the training set and the test set of the CoNLL shared task 2009 for our experiments. Table 1 shows the elapsed times in $\frac{1}{1000}$ seconds (milliseconds) of the selected languages for the procedure calls in the loops of Algorithm 1. We had to measure the times for the feature extraction in the parsing algorithm, since in the training algorithm, the time can only be measured together with the time for storing the features. The table contains additional figures for the total training time and parsing scores.[1]

The parsing algorithm itself only required, to our surprise, 12.5 ms (t_p) for a English sentence

[1] We use a Intel Nehalem i7 CPU 3.33 Ghz. With turbo mode on, the clock speed was 3.46 Ghz.

on average, while the feature extraction needs 1223 ms. To extract the features takes about 100 times longer than to build a projective dependency tree. The feature extraction is already implemented efficiently. It uses only numbers to represent features which it combines to a long integer number and then maps by a hash table[2] to a 32bit integer number. The parsing algorithm uses the integer number as an index to access the weights in the vectors $\overrightarrow{w}$ and $\overrightarrow{v}$.

The complexity of the parsing algorithm is usually considered the reason for long parsing times. However, it is not the most time consuming component as proven by the above analysis. Therefore, we investigated the question further, asking what causes the high time consumption of the feature extraction?

In our next experiment, we left out the mapping of the features to the index of the weight vectors. The feature extraction takes 88 ms/sentence without the mapping and 1223 ms/sentence with the mapping. The feature–index mapping needs 93% of the time to extract the features and 91% of the total parsing time. What causes the high time consumption of the feature–index mapping?

The mapping has to provide a number as an index for the features in the training examples and to filter out the features of examples built, while the parser predicts the dependency structures. The algorithm filters out negative features to reduce the memory requirement, even if they could improve the parsing result. We will call the features built due to the training examples positive features and the rest negative features. We counted 5.8 times more access to negative features than positive features.

We now look more into the implementation details of the used hash table to answer the previously asked question. The hash table for the feature–index mapping uses three arrays: one for the keys, one for the values and a status array to indicate the deleted elements. If a program stores a value then the hash function uses the key to calculate the location of the value. Since the hash function is a heuristic function, the predicted location might be wrong, which leads to so-called hash misses. In such cases the hash algorithm has to retry to find the value. We counted 87% hash misses including misses where the hash had to retry several times. The number of hash misses was high, because of the additional negative features. The CPU cache can only store a small amount of the data from the hash table. Therefore, the memory controller has frequently to transfer data from the main memory into the CPU. This procedure is relatively slow. We traced down the high time consumption to the access of the key and the access of the value. Successive accesses to the arrays are fast, but the relative random accesses via the hash function are very slow. The large number of accesses to the three arrays, because of the negative features, positive features and because of the hash misses multiplied by the time needed to transfer the data into the CPU are the reason for the high time consumption.

We tried to solve this problem with Bloom filters, larger hash tables and customized hash functions to reduce the hash misses. These techniques did not help much. However, a substantial improvement did result when we eliminated the hash table completely, and directly accessed the weight vectors $\overrightarrow{w}$ and $\overrightarrow{v}$ with a hash function. This led us to the use of Hash Kernels.

4 Hash Kernel

A Hash Kernel for structured data uses a hash function $h : J \rightarrow \{1...n\}$ to index ϕ, cf. Shi et al. (2009). ϕ maps the observations X to a feature space. We define $\phi(x, y)$ as the numeric feature representation indexed by J. Let $\overline{\phi}_k(x, y) = \phi_j(x, y)$ the hash based feature–index mapping, where $h(j) = k$. The process of parsing a sentence x_i is to find a parse tree y_p that maximizes a scoring function $\text{argmax}_y F(x_i, y)$. The learning problem is to fit the function F so that the errors of the predicted parse tree y are as low as possible. The scoring function of the Hash Kernel is

$$F(x, y) = \overrightarrow{w} * \overline{\phi}(x, y)$$

where $\overrightarrow{w}$ is the weight vector and the size of $\overrightarrow{w}$ is n.

Algorithm 2 shows the update function of the Hash Kernel. We derived the update function from the update function of MIRA (Crammer et

[2] We use the hash tables of the *trove* library: `http://sourceforge.net/projects/trove4j`.

Algorithm 2: Update of the Hash Kernel

```
// $y_p$ = arg max$_y F(x_i, y)$
**update(**$\overrightarrow{w}, \overrightarrow{v}, x_i, y_i, y_p, \gamma$**)**
  $\epsilon = \Delta(y_i, y_p)$ // number of wrong labeled edges
  **if** $\epsilon > 0$ **then**
    $\overrightarrow{u} \leftarrow (\overline{\phi}(x_i, y_i) - \overline{\phi}(x_i, y_p))$
    $\nu = \frac{\epsilon - (F(x_t, y_i) - F(x_i, y_p))}{||\overrightarrow{u}||^2}$
    $\overrightarrow{w} \leftarrow \overrightarrow{w} + \nu * \overrightarrow{u}$
    $\overrightarrow{v} \leftarrow \vec{v} + \gamma * \nu * \overrightarrow{u}$
**return** $\overrightarrow{w}, \overrightarrow{v}$
```

al., 2006). The parameters of the function are the weight vectors $\overrightarrow{w}$ and $\overrightarrow{v}$, the sentence x_i, the gold dependency structure y_i, the predicted dependency structure y_p, and the update weight γ. The function Δ calculates the number of wrong labeled edges. The update function updates the weight vectors, if at least one edge is labeled wrong. It calculates the difference $\overrightarrow{u}$ of the feature vectors of the gold dependency structure $\overline{\phi}(x_i, y_i)$ and the predicted dependency structure $\overline{\phi}(x_i, y_p)$. Each time, we use the feature representation ϕ, the hash function h maps the features to integer numbers between 1 and $|\overrightarrow{w}|$. After that the update function calculates the margin ν and updates $\overrightarrow{w}$ and $\overrightarrow{v}$ respectively.

Algorithm 3 shows the training algorithm for the Hash Kernel in pseudo code. A main difference to the baseline algorithm is that it does not store the features because of the required time which is needed to store the additional negative features. Accordingly, the algorithm first extracts the features for each training instance, then maps the features to indexes for the weight vector with the hash function and calculates the weight arrays.

Algorithm 3: Training – Hash Kernel

```
**for** n $\leftarrow$ 1 **to** $E$ // iteration over the training epochs
  **for** i $\leftarrow$ 1 **to** $I$ // iteration over the training exmaples
    $k \leftarrow (n-1) * I + i$
    $\gamma \leftarrow E * I - k + 2$ // passive-aggressive weight
    $t^s_{e,k}$; $A \leftarrow$ extr.-features-&-calc-arrays(i,$\overrightarrow{w}$) ; $t^e_{e,k}$
    $t^s_{p,k}$; $y_p \leftarrow$ predicte-projective-parse-tree($A$);$t^e_{p,k}$
    $t^s_{a,k}$; $y_a \leftarrow$ non-projective-approx.($y_p$,$A$); $t^e_{a,k}$
    update $\overrightarrow{w}$, $\overrightarrow{v}$ according to $\Delta(y_p, y_i)$ and $\gamma$
$w = v/(E * I)$ // average
```

For different j, the hash function $h(j)$ might generate the same value k. This means that the hash function maps more than one feature to the same weight. We call such cases collisions. Collisions can reduce the accuracy, since the weights are changed arbitrarily. This procedure is similar to randomization of weights (features), which aims to save space by sharing values in the weight vector (Blum., 2006; Rahimi and Recht, 2008). The Hash Kernel shares values when collisions occur that can be considered as an approximation of the kernel function, because a weight might be adapted due to more than one feature. If the approximation works well then we would need only a relatively small weight vector otherwise we need a larger weight vector to reduce the chance of collisions. In an experiments, we compared two hash functions and different hash sizes. We selected for the comparison a standard hash function (h_1) and a custom hash function (h_2). The idea for the custom hash function h_2 is not to overlap the values of the feature sequence number and the edge label with other values. These values are stored at the beginning of a long number, which represents a feature.

$$h_1 \leftarrow |(l\ xor(l \vee \text{0xffffffff00000000} >> 32))\%\ \text{size}|$$ [3]

$$\begin{aligned} h_2 \leftarrow |(l\ xor\ & ((l >> 13)\ \vee\ \text{0xfffffffffffe000})\ xor \\ & ((l >> 24)\ \vee\ \text{0xffffffffffff0000})\ xor \\ & ((l >> 33)\ \vee\ \text{0xfffffffffffc0000})\ xor \\ & ((l >> 40)\ \vee\ \text{0xfffffffffff00000}))\ \%\ \text{size}\ | \end{aligned}$$

vector size	h_1	#(h_1)	h_2	#(h_2)
411527	85.67	0.41	85.74	0.41
3292489	87.82	3.27	87.97	3.28
10503061	88.26	8.83	88.35	8.77
21006137	88.19	12.58	88.41	12.53
42012281	88.32	12.45	88.34	15.27
115911564*	88.32	17.58	88.39	17.34
179669557	88.34	17.65	88.28	17.84

Table 2: The labeled attachment scores for different weight vector sizes and the number of nonzero values in the feature vectors in millions. * Not a prime number.

Table 2 shows the labeled attachment scores for selected weight vector sizes and the number of nonzero weights. Most of the numbers in Table 2 are primes, since they are frequently used to obtain a better distribution of the content in hash ta-

[3] $>> n$ shifts n bits right, and % is the modulo operation.

bles. h_2 has more nonzero weights than h_1. Nevertheless, we did not observe any clear improvement of the accuracy scores. The values do not change significantly for a weight vector size of 10 million and more elements. We choose a weight vector size of 115911564 values for further experiments since we get more non zero weights and therefore fewer collisions.

	t_e	t_p	t_a	r	total	par.	trai.
Chinese	1308	-	200	3	1511	1184	93h
English	379	21.3	18.2	1.5	420	354	46h
German	209	12	15.3	1.7	238	126	24h
Spanish	1056	-	39	2	1097	1044	44h

Table 3: The time in milliseconds for the feature extraction, projective parsing, non-projective approximation, rest (r), the total training time per instance, the average parsing (par.) time in milliseconds for the test set and the training time in hours

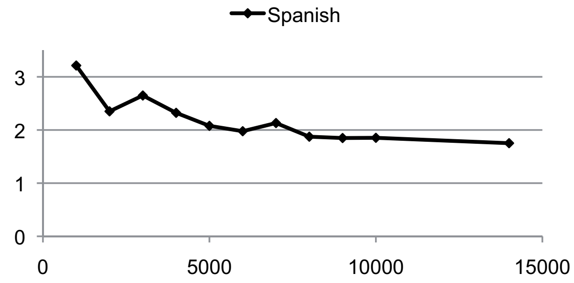


Figure 1: The difference of the labeled attachment score between the baseline parser and the parser with the Hash Kernel (y-axis) for increasing large training sets (x-axis).

Table 3 contains the measured times for the Hash Kernel as used in Algorithm 2. The parser needs 0.354 seconds in average to parse a sentence of the English test set. This is 3.5 times faster than the baseline parser. The reason for that is the faster feature mapping of the Hash Kernel. Therefore, the measured time t_e for the feature extraction and the calculation of the weight arrays are much lower than for the baseline parser. The training is about 19% slower since we could no longer use a file to store the feature indexes of the training examples because of the large number of negative features. We counted about twice the number of nonzero weights in the weight vector of the Hash Kernel compared to the baseline parser. For instance, we counted for English 17.34 Millions nonzero weights in the Hash Kernel and 8.47 Millions in baseline parser and for Chinese 18.28 Millions nonzero weights in the Hash Kernel and 8.76 Millions in the baseline parser. Table 6 shows the scores for all languages of the shared task 2009. The attachment scores increased for all languages. It increased most for Catalan and Spanish. These two corpora have the smallest training sets. We searched for the reason and found that the Hash Kernel provides an overproportional accuracy gain with less training data compared to MIRA. Figure 1 shows the difference between the labeled attachment score of the parser with MIRA and the Hash Kernel for Spanish. The decreasing curve shows clearly that the Hash Kernel provides an overproportional accuracy gain with less training data compared to the baseline. This provides an advantage for small training corpora.

However, this is probably not the main reason for the high improvement, since for languages with only slightly larger training sets such as Chinese the improvement is much lower and the gradient at the end of the curve is so that a huge amount of training data would be needed to make the curve reach zero.

5 Parallelization

Current CPUs have up to 12 cores and we will see soon CPUs with more cores. Also graphic cards provide many simple cores. Parsing algorithms can use several cores. Especially, the tasks to extract the features and to calculate the weight arrays can be well implemented as parallel algorithm. We could also successful parallelize the projective parsing and the non-projective approximation algorithm. Algorithm 4 shows the parallel feature extraction in pseudo code. The main method prepares a list of tasks which can be performed in parallel and afterwards it creates the threads that perform the tasks. Each thread removes from the task list an element, carries out the task and stores the result. This procedure is repeated until the list is empty. The main method waits until all threads are completed and returns the result. For the parallel algorithms, Table 5 shows the elapsed times depend on the number of

#	**Standard Features**	#	**Linear Features**		**Linear G. Features**		**Sibling Features**
1	l,h_f,h_p,d(h,d)	14	l,h_p,h+1_p,d_p,d(h,d)	44	l,g_p,d_p,d+1_p,d(h,d)	99	l,s_l,h_p,d(h,d)$\oplus$r(h,d)
2	l,h_f,d(h,d)	15	l,h_p,d-1_p,d_p,d(h,d)	45	l,g_p,d_p,d-1_p,d(h,d)	100	l,s_l,d_p,d(h,d)$\oplus$r(h,d)
3	l,h_p,d(h,d)	16	l,h_p,d_p,d+1_p,d(h,d)	46	l,g_p,g+1_p,d-1_p,d_p,d(h,d)	101	l,h_l,d_p,d(h,d)$\oplus$r(h,d)
4	l,d_f,d_p,d(h,d)	17	l,h_p,h+1_p,d-1_p,d_p,d(h,d)	47	l,g-1_p,g_p,d-1_p,d_p,d(h,d)	102	l,d_l,s_p,d(h,d)$\oplus$r(h,d)
5	l,h_p,d(h,d)	18	l,h-1_p,h+1_p,d-1_p,d_p,d(h,d)	48	l,g_p,g+1_p,d_p,d+1_p,d(h,d)	75	l,∀d_m,∀s_m,d(h,d)
6	l,d_p,d(h,d)	19	l,h_p,h+1_p,d_p,d+1_p,d(h,d)	49	l,g-1_p,g_p,d_p,d+1_p,d(h,d)	76	l,∀h_m,∀s_m,d(h,s)
7	l,h_f,h_p,d_f,d_p,d(h,d)	20	l,h-1_p,h_p,d_p,d-1_p,d(h,d)	50	l,g_p,g+1_p,h_p,d(h,d)		**Linear S. Features**
8	l,h_p,d_f,d_p,d(h,d)		**Grandchild Features**	51	l,g_p,g-1_p,h_p,d(h,d)	58	l,s_p,s+1_p,h_p,d(h,d)
9	l,h_f,d_f,d_p,d(h,d)	21	l,h_p,d_p,g_p,d(h,d,g)	52	l,g_p,h_p,h+1_p,d(h,d)	59	l,s_p,s-1_p,h_p,d(h,d)
10	l,h_f,h_p,d_f,d(h,d)	22	l,h_p,g_p,d(h,d,g)	53	l,g_p,h_p,h-1_p,d(h,d)	60	l,s_p,h_p,h+1_p,d(h,d)
11	l,h_f,d_f,h_p,d(h,d)	23	l,d_p,g_p,d(h,d,g)	54	l,g_p,g+1_p,h-1_p,h_p,d(h,d)	61	l,s_p,h_p,h-1_p,d(h,d)
12	l,h_f,d_f,d(h,d)	24	l,h_f,g_f,d(h,d,g)	55	l,g-1_p,g_p,h-1_p,h_p,d(h,d)	62	l,s_p,s+1_p,h-1_p,d(h,d)
13	l,h_p,d_p,d(h,d)	25	l,d_f,g_f,d(h,d,g)	56	l,g_p,g+1_p,h_p,h+1_p,d(h,d)	63	l,s-1_p,s_p,h-1_p,d(h,d)
77	l,h_l,h_p,d(h,d)	26	l,g_f,h_p,d(h,d,g)	57	l,g-1_p,g_p,h_p,h+1_p,d(h,d)	64	l,s_p,s+1_p,h_p,d(h,d)
78	l,h_l,d(h,d)	27	l,g_f,d_p,d(h,d,g)		**Sibling Features**	65	l,s-1_p,s_p,h_p,h+1_p,d(h,d)
79	l,h_p,d(h,d)	28	l,h_f,g_p,d(h,d,g)	30	l,h_p,d_p,s_p,d(h,d) $\oplus$r(h,d)	66	l,s_p,s+1_p,d_p,d(h,d)
80	l,d_l,d_p,d(h,d)	29	l,d_f,g_p,d(h,d,g)	31	l,h_p,s_p,d(h,d)$\oplus$r(h,d)	67	l,s_p,s-1_p,d_p,d(h,d)
81	l,d_l,d(h,d)	91	l,h_l,g_l,d(h,d,g)	32	l,d_p,s_p,d(h,d)$\oplus$r(h,d)	68	s_p,d_p,d+1_p,d(h,d)
82	l,d_p,d(h,d)	92	l,d_p,g_p,d(h,d,g)	33	l,p_f,s_f,d(h,d)$\oplus$r(h,d)	69	s_p,d_p,d-1_p,d(h,d)
83	l,d_l,h_p,d_p,h_l,d(h,d)	93	l,g_l,h_p,d(h,d,g)	34	l,p_p,s_f,d(h,d)$\oplus$r(h,d)	70	s_p,s+1_p,d-1_p,d_p,d(h,d)
84	l,d_l,h_p,d_p,d(h,d)	94	l,g_l,d_p,d(h,d,g)	35	l,s_f,p_p,d(h,d)$\oplus$r(h,d)	71	s-1_p,s_p,d-1_p,d_p,d(h,d)
85	l,h_l,d_l,d_p,d(h,d)	95	l,h_l,g_p,d(h,d,g)	36	l,s_f,d_p,d(h,d)$\oplus$r(h,d)	72	s_p,s+1_p,d_p,d+1_p,d(h,d)
86	l,h_l,h_p,d_p,d(h,d)	96	l,d_l,g_p,d(h,d,g)	37	l,s_f,d_p,d(h,d)$\oplus$r(h,d)	73	s-1_p,s_p,d_p,d+1_p,d(h,d)
87	l,h_l,d_l,h_p,d(h,d)	74	l,∀d_m,∀g_m,d(h,d)	38	l,d_f,s_p,d(h,d)$\oplus$r(h,d)		**Special Feature**
88	l,h_l,d_l,d(h,d)		**Linear G. Features**	97	l,h_l,s_l,d(h,d)$\oplus$r(h,d)	39	∀l,h_p,d_p,$x_p$between h,d
89	l,h_p,d_p,d(h,d)	42	l,g_p,g+1_p,d_p,d(h,d)	98	l,d_l,s_l,d(h,d)$\oplus$r(h,d)		
41	l,∀h_m,∀d_m,d(h,d)	43	l,g_p,g-1_p,d_p,d(h,d)				

Table 4: Features Groups. *l* represents the label, *h* the head, d the dependent, *s* a sibling, and *g* a grandchild, **d**(x,y,[,z]) the order of words, and **r**(x,y) the distance.

used cores. The parsing time is 1.9 times faster on two cores and 3.4 times faster on 4 cores. Hyper threading can improve the parsing times again and we get with hyper threading 4.6 faster parsing times. Hyper threading possibly reduces the overhead of threads, which contains already our single core version.

Algorithm 4: Parallel Feature Extraction

```
$A$ // weight arrays
extract-features-and-calc-arrays($x_i$)
  data-list ← {} // thread-save data list
  for $w_1$ ← 1 to $|x_i|$
    for $w_2$ ← 1 to $|x_i|$
      data-list ← data-list $\cup\{(w_1, w_2)\}$
  $c$ ← number of CPU cores
  for $t$ ← 1 to $c$
    $T_t$ ← create-array-thread($t, x_i$,data-list)
    start array-thread $T_t$// start thread t
  for $t$ ← 1 to $c$
    join $T_t$// wait until thread $t$ is finished
    $A$ ← $A \cup$ collect-result($T_t$)
return $A$
//
array-thread $T$
  $d$ ← remove-first-element(data-list)
  if $d$ is empty then end-thread
  ... // extract features and calculate part $d$ of $A$
```

Cores	t_e	t_p	t_a	rest	total	pars.	train.
1	379	21.3	18.2	1.5	420	354	45.8h
2	196	11.7	9.2	2.1	219	187	23.9h
3	138	8.9	6.5	1.6	155	126	16.6h
4	106	8.2	5.2	1.6	121	105	13.2h
4+4h	73.3	8.8	4.8	1.3	88.2	77	9.6h

Table 5: Elapsed times in milliseconds for different numbers of cores. The parsing time (pars.) are expressed in milliseconds per sentence and the training (train.) time in hours. The last row shows the times for 8 threads on a 4 core CPU with Hyper-threading. For these experiment, we set the clock speed to 3.46 Ghz in order to have the same clock speed for all experiments.

6 Non-Projective Approximation Threshold

For non-projective parsing, we use the Non-Projective Approximation Algorithm of McDonald and Pereira (2006). The algorithm rearranges edges in a dependency tree when they improve the score. Bohnet (2009) extended the algorithm by a threshold which biases the rearrangement of the edges. With a threshold, it is possible to gain a higher percentage of correct dependency links. We determined a threshold in experiments for Czech, English and German. In the experiment, we use the Hash Kernel and increase the thresh-

System	Average	Catalan	Chinese	Czech	English	German	Japanese	Spanish
Top CoNLL 09	85.77$^{(1)}$	**87.86**$^{(1)}$	**79.19**$^{(4)}$	80.38$^{(1)}$	89.88$^{(2)}$	87.48$^{(2)}$	**92.57**$^{(3)}$	87.64$^{(1)}$
Baseline Parser	85.10	85.70	76.88	76.93	90.14	87.64	92.26	86.12
this work	**86.33**	87.45	76.99	**80.96**	**90.33**	**88.06**	92.47	**88.13**

Table 6: Top LAS of the CoNLL 2009 of (1) Gesmundo et al. (2009), (2) Bohnet (2009), (3) Che et al. (2009), and (4) Ren et al. (2009); LAS of the baseline parser and the parser with Hash Kernel. The numbers in bold face mark the top scores. We used for Catalan, Chinese, Japanese and Spanish the projective parsing algorithm.

old at the beginning in small steps by 0.1 and later in larger steps by 0.5 and 1.0. Figure 2 shows the labeled attachment scores for the Czech, English and German development set in relation to the rearrangement threshold. The curves for all languages are a bit volatile. The English curve is rather flat. It increases a bit until about 0.3 and remains relative stable before it slightly decreases. The labeled attachment score for German and Czech increases until 0.3 as well and then both scores start to decrease. For English a threshold between 0.3 and about 2.0 would work well. For German and Czech, a threshold of about 0.3 is the best choice. We selected for all three languages a threshold of 0.3.

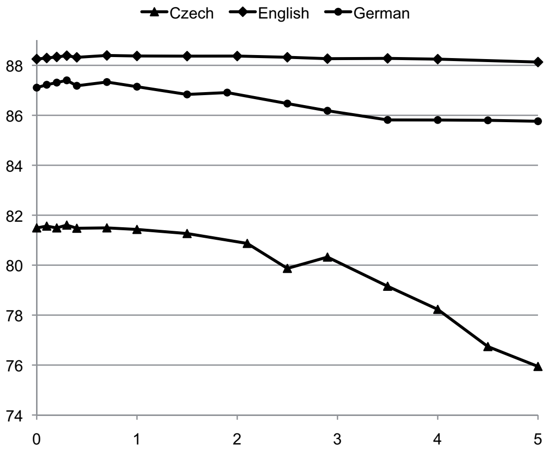


Figure 2: English, German, and Czech labeled attachment score (y-axis) for the development set in relation to the rearrangement threshold (x-axis).

7 Conclusion and Future Work

We have developed a very fast parser with excellent attachment scores. For the languages of the 2009 CoNLL Shared Task, the parser could reach higher accuracy scores on average than the top performing systems. The scores for Catalan, Chinese and Japanese are still lower than the top scores. However, the parser would have ranked second for these languages. For Catalan and Chinese, the top results obtained transition-based parsers. Therefore, the integration of both techniques as in Nivre and McDonald (2008) seems to be very promising. For instance, to improve the accuracy further, more global constrains capturing the subcategorization correct could be integrated as in Riedel and Clarke (2006). Our faster algorithms may make it feasible to consider further higher order factors.

In this paper, we have investigated possibilities for increasing parsing speed without any accuracy loss. The parsing time is 3.5 times faster on a single CPU core than the baseline parser which has an typical architecture for a maximum spanning tree parser. The improvement is due solely to the Hash Kernel. The Hash Kernel was also a prerequisite for the parallelization of the parser because it requires much less memory bandwidth which is nowadays a bottleneck of parsers and many other applications.

By using parallel algorithms, we could further increase the parsing time by a factor of 3.4 on a 4 core CPU and including hyper threading by a factor of 4.6. The parsing speed is 16 times faster for the English test set than the conventional approach. The parser needs only 77 millisecond in average to parse a sentence and the speed will scale with the number of cores that become available in future. To gain even faster parsing times, it may be possible to trade accuracy against speed. In a pilot experiment, we have shown that it is possible to reduce the parsing time in this way to as little as 9 milliseconds. We are convinced that the Hash Kernel can be applied successful to transition based dependency parsers, phrase structure parsers and many other NLP applications. [4]

[4] We provide the Parser and Hash Kernel as open source for download from http://code.google.com/p/mate-tools.

References

Attardi, G. 2006. Experiments with a Multilanguage Non-Projective Dependency Parser. In *Proceedings of CoNLL*, pages 166–170.

Blum., A. 2006. Random Projection, Margins, Kernels, and Feature-Selection. In *LNCS*, pages 52–68. Springer.

Bohnet, B. 2009. Efficient Parsing of Syntactic and Semantic Dependency Structures. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009)*.

Carreras, X. 2007. Experiments with a Higher-order Projective Dependency Parser. In *EMNLP/CoNLL*.

Che, W., Li Z., Li Y., Guo Y., Qin B., and Liu T. 2009. Multilingual Dependency-based Syntactic and Semantic Parsing. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009)*.

Collins, M. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *EMNLP*.

Crammer, K., O. Dekel, S. Shalev-Shwartz, and Y. Singer. 2003. Online Passive-Aggressive Algorithms. In *Sixteenth Annual Conference on Neural Information Processing Systems (NIPS)*.

Crammer, K., O. Dekel, S. Shalev-Shwartz, and Y. Singer. 2006. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585.

Eisner, J. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pages 340–345, Copenhaen.

Eisner, J., 2000. *Bilexical Grammars and their Cubic-time Parsing Algorithms*, pages 29–62. Kluwer Academic Publishers.

Gesmundo, A., J. Henderson, P. Merlo, and I. Titov. 2009. A Latent Variable Model of Synchronous Syntactic-Semantic Parsing for Multiple Languages. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009)*, Boulder, Colorado, USA., June 4-5.

Hajič, J., M. Ciaramita, R. Johansson, D. Kawahara, M. Antònia Martí, L. Màrquez, A. Meyers, J. Nivre, S. Padó, J. Štěpánek, P. Straňák, M. Surdeanu, N. Xue, and Y. Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the 13th CoNLL-2009, June 4-5*, Boulder, Colorado, USA.

Johansson, R. and P. Nugues. 2008. Dependency-based Syntactic–Semantic Analysis with PropBank and NomBank. In *Proceedings of the Shared Task Session of CoNLL-2008*, Manchester, UK.

McDonald, R. and F. Pereira. 2006. Online Learning of Approximate Dependency Parsing Algorithms. In *In Proc. of EACL*, pages 81–88.

McDonald, R., K. Crammer, and F. Pereira. 2005. Online Large-margin Training of Dependency Parsers. In *Proc. ACL*, pages 91–98.

Nivre, J. and R. McDonald. 2008. Integrating Graph-Based and Transition-Based Dependency Parsers. In *ACL-08*, pages 950–958, Columbus, Ohio.

Nivre, J., J. Hall, and J. Nilsson. 2004. Memory-Based Dependency Parsing. In *Proceedings of the 8th CoNLL*, pages 49–56, Boston, Massachusetts.

Nivre, J. 2003. An Efficient Algorithm for Projective Dependency Parsing. In *8th International Workshop on Parsing Technologies*, pages 149–160, Nancy, France.

Nivre, J. 2009. Non-Projective Dependency Parsing in Expected Linear Time. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 351–359, Suntec, Singapore.

Rahimi, A. and B. Recht. 2008. Random Features for Large-Scale Kernel Machines. In Platt, J.C., D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. MIT Press, Cambridge, MA.

Ren, H., D. Ji Jing Wan, and M. Zhang. 2009. Parsing Syntactic and Semantic Dependencies for Multiple Languages with a Pipeline Approach. In *Proceedings of the 13th Conference on Computational Natural Language Learning (CoNLL-2009)*, Boulder, Colorado, USA., June 4-5.

Riedel, S. and J. Clarke. 2006. Incremental Integer Linear Programming for Non-projective Dependency Parsing. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 129–137, Sydney, Australia, July. Association for Computational Linguistics.

Shi, Q., J. Petterson, G. Dror, J. Langford, A. Smola, and S.V.N. Vishwanathan. 2009. Hash Kernels for Structured Data. In *Journal of Machine Learning*.

Titov, I. and J. Henderson. 2007. A Latent Variable Model for Generative Dependency Parsing. In *Proceedings of IWPT*, pages 144–155.

Yamada, H. and Y. Matsumoto. 2003. Statistical Dependency Analysis with Support Vector Machines. In *Proceedings of IWPT*, pages 195–206.

Broad Coverage Multilingual Deep Sentence Generation with a Stochastic Multi-Level Realizer

Bernd Bohnet[1], **Leo Wanner**[1,2], **Simon Mille**[1], **Alicia Burga**[1]
[1]Department of Information and Communication Technologies
Pompeu Fabra University
[2]Institució Catalana de Recerca i Estudis Avançats (ICREA)
{first-name.last-name}@upf.edu

Abstract

Most of the known stochastic sentence generators use syntactically annotated corpora, performing the projection to the surface in one stage. However, in full-fledged text generation, sentence realization usually starts from semantic (predicate-argument) structures. To be able to deal with semantic structures, stochastic generators require semantically annotated, or, even better, multilevel annotated corpora. Only then can they deal with such crucial generation issues as sentence planning, linearization and morphologization. Multilevel annotated corpora are increasingly available for multiple languages. We take advantage of them and propose a multilingual deep stochastic sentence realizer that mirrors the state-of-the-art research in semantic parsing. The realizer uses an SVM learning algorithm. For each pair of adjacent levels of annotation, a separate decoder is defined. So far, we evaluated the realizer for Chinese, English, German, and Spanish.

1 Introduction

Recent years saw a significant increase of interest in corpus-based natural language generation (NLG), and, in particular, in corpus-based (or stochastic) *sentence realization*, i.e., that part of NLG which deals with mapping of a formal (more or less abstract) sentence plan onto a chain of inflected words; cf., among others, (Langkilde and Knight, 1998; Oh and Rudnicky, 2000; Bangalore and Rambow, 2000; Wan et al., 2009). The advantage of stochastic sentence realization over traditional rule-based realization is mainly threefold: (i) it is more robust, (ii) it usually has a significantly larger coverage; (iii) it is *per se* language- and domain-independent. Its disadvantage is that it requires at least syntactically annotated corpora of significant size (Bangalore et al., 2001). Given the aspiration of NLG to start from numeric time series or conceptual or semantic structures, syntactic annotation even does not suffice: the corpora must also be at least semantically annotated. Up to date, deep stochastic sentence realization was hampered by the lack of multiple-level annotated corpora. As a consequence, available stochastic sentence generators either take syntactic structures as input (and avoid thus the need for multiple-level annotation) (Bangalore and Rambow, 2000; Langkilde-Geary, 2002; Filippova and Strube, 2008), or draw upon hybrid models that imply a symbolic submodule which derives the syntactic representation that is then used by the stochastic submodule (Knight and Hatzivassiloglou, 1995; Langkilde and Knight, 1998).

The increasing availability of multilevel annotated corpora, such as the corpora of the shared task of the Conference on Computational Natural Language Learning (CoNLL), opens new perspectives with respect to deep stochastic sentence generation—although the fact that these corpora have not been annotated with the needs of generation in mind, may require additional adjustments, as has been, in fact, in the case of our work.

In this paper, we present a Support Vector Machine (SVM)-based multilingual dependency-oriented stochastic deep sentence realizer that uses multilingual corpora of the CoNLL '09 shared task (Hajič, 2009) for training. The sentences of these corpora are annotated with shallow semantic structures, dependency trees, and lemmata; for some of the languages involved, they also contain morphological feature annotations. The multilevel annotation allows us to take into account all levels of representation needed for linguistic generation and to model the projection between pairs of adjacent levels by separate decoders, which, in its turn, facilitates the coverage of such critical generation tasks as sentence planning, linearization, and morphologization. The presented realizer is, in principle, language-independent in that it is trainable on any multilevel annotated corpus. In this paper, we discuss its performance for Chinese, English, German, and Spanish.

The remainder of the paper is structured as follows. In Section 2, we discuss how the shallow semantic annotation in the CoNLL '09 shared task corpora should be completed in order to be suitable for generation. Section 3 presents the training setup of our realizer. Section 4 shows the individual stages of sentence realization: from the semantic structure to the syntactic structure, from the syntactic structure to the linearized structure and from the linearized structure to a chain of inflected word forms (if applicable for the language in question). Section 5 outlines the experimental set up for the evaluation of our realizer and discusses the results of this evaluation. In Section 6, finally, some conclusions with respect to the characteristics of our realizer and its place in the research landscape are drawn.

The amount of the material which comes into play makes it impossible to describe all stages in adequate detail. However, we hope that the overview provided in what follows still suffices to fully assess our proposal.

2 Completing the Semantic Annotation

The semantic annotation of sentences in CoNLL '09 shared task corpora follows the PropBank annotation guidelines (Palmer et al., 2005). Problematic from the viewpoint of generation is that this annotation is not always a connected acyclic graph. As a consequence, in these cases no valid (connected) syntactic tree can be derived. The most frequent cases of violation of the connectivity principle are not attached adjectival modifiers, determiners, adverbs, and coordinations; sometimes, the verb is not connected with its argument(s). Therefore, prior to starting the training procedure, the semantic annotation must be completed: non-connected adjectival modifiers must be annotated as predicates with their syntactic heads as arguments, determiners must be "translated" into quantifiers, detached verbal arguments must be connected with their head, etc.

Algorithm 1 displays the algorithm that completes the semantic annotations of the corpora. Each sentence x_i of the corpus I, with $i = 1, \ldots, |I|$, is annotated with its dependency tree y_i and its shallow semantic graph s_i. The algorithm traverses y_i breath-first, and examines for each node n in y_i whether n's corresponding node in s_i is connected with the node corresponding to the parent of n. If not, the algorithm connects both by a directed labeled edge. The direction and the label of the edge are selected consulting a look up table in which default labels and the orientation of the edges between different node categories are specified.

Figure 1 shows the semantic representation of a sample English sentence obtained after the application of Algorithm 1. The solid edges are the edges available in the original annotation; the dashed edges have been introduced by the algorithm. The edge labels 'A0' and 'A1' stand for "first argument" and "second argument" (of the corresponding head), respectively, 'R-A0' for "A0 realized as a relative clause", and 'AM-MNR' for "manner modifier". As can be seen, 6 out of the total of 14 edges in the complete representation of this example have been added by Algorithm 1. We still did not finish the formal evaluation of the principal changes necessary to adapt the PropBank annotation for generation, nor the quality of our completion algorithm. However, the need of an annotation with generation in mind is obvious.

Algorithm 1: Complete semantic graph

```
//$s_i$ is a semantic graph and $y_i$ a dependency tree
// $s_i = \langle N_{s_i}, L_{s_i}, E_{s_i} \rangle$, where $N_{s_i}$ is the set of nodes
// $L_{s_i}$ the set of edge labels
// $E_{s_i} \subseteq N_s \times N_s \times L_s$ is the set of edges
for $i \leftarrow 1$ to $|I|$ // iteration over the training examples
  let $r_y \in y_i$ be the root node of the dependency tree
  // initialization of the queue
  $nodeQueue \leftarrow children(r_y)$
  while $nodeQueue \neq \emptyset$ do
    $n_y \leftarrow removeFirst$(nodeQueue)
    // breath first: add nodes at the end of the queue
    nodeQueue $\leftarrow$ nodeQueue $\cup$ $children(n_y)$
    $n_{y_s} \leftarrow sem(n_y)$; $p_{y_s} \leftarrow sem(parent(n_y))$
    //get the semantic equivalents of $n_y$ and of its parent
    if not exists path($n_{y_s}, p_{y_s}$) then
      $l \leftarrow label(n_y, parent(n_y))$
      $l_s \leftarrow look\text{-}up\text{-}sem\text{-}label(n_{y_s}, p_{y_s}, l)$
      if $look\text{-}up\text{-}sem\text{-}direction(n_{y_s}, p_{y_s}, l_s)$ = "$\rightarrow$" then
        // add the semantic edge
        $E_s \leftarrow E_s \cup (p_{y_s}, n_{y_s}, l_s)$
      else // direction of the edge "$\leftarrow$"
        // add the semantic edge
        $E_s \leftarrow E_s \cup (n_{y_s}, p_{y_s}, l_s)$
```

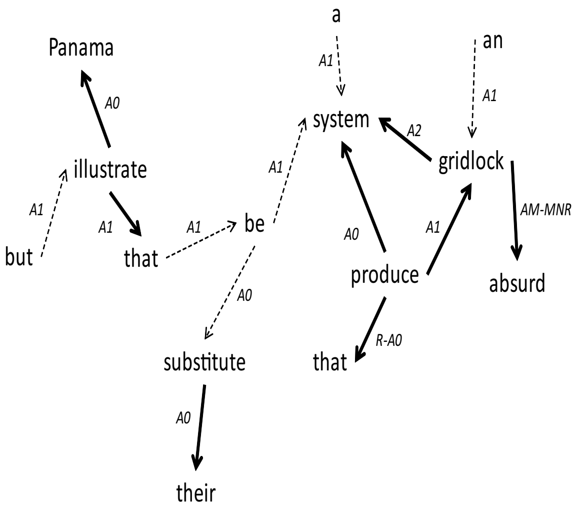


Figure 1: Semantic representation of the sentence *But Panama illustrates that their substitute is a system that produces an absurd gridlock.* after completion

3 Realizer Training Setup

Figure 2 shows the training setup of our realizer. For each level of annotation, an SVM feature extractor and for each pair of adjacent levels of annotation, an SVM decoder is defined. The Sem-Synt decoder constructs from a semantic graph the corresponding dependency tree. The Synt-Linearization decoder derives from a dependency tree a chain of lemmata, i.e., determines the word order within the sentence. The Linearization-Morph decoder generates the inflected word form for each lemma in the chain. Both the feature extractors and the decoders are language-independent, which makes the realizer applicable to any language for which multilevel-annotated corpora are available.

To compute the score of the alternative realizations by each decoder, we apply MIRA (Margin Infused Relaxed Algorithm) to the features provided by the feature extractors. MIRA is one of the most successful large-margin training techniques for structured data (Crammer et al., 2006). It has been used, e.g., for dependency parsing, semantic role labelling, chunking and tagging. Since we have similar feature sets (of comparable size) as those for which MIRA has proven to work well, we assume that it will also perform well for sentence realization. Unfortunately, due to the lack of space, we cannot present here the instantiation of MIRA for all stages of our model. For illustration, Algorithm 2 outlines it for morphological realization.

The morphologic realization uses the minimal string edit distance (Levenshtein, 1966) to map lemmata to word forms. As input to the MIRA-classifier, we use the lemmata of a sentence, its dependency tree and the already ordered sentence. The characters of the input strings are reversed since most of the changes occur at the end of the words and the string edit scripts work relatively to the beginning of the string. For example, to calculate the minimal string edit distance between the lemma *go* and the form *goes*, both are first reversed by the function **compute-edit-dist** and then the minimal string edit script between *og* and *seog* is computed. The resulting script is *Ie0Is0*. It translates into the operations 'insert *e* at the position 0 of the input string' and 'insert *s* at the position 0'.

Before MIRA starts, we compute all minimal edit distance scripts to be used as classes of MIRA. Only scripts that occur more often than twice are used. The number of the resulting edit scripts is language-dependent; e.g., we get about

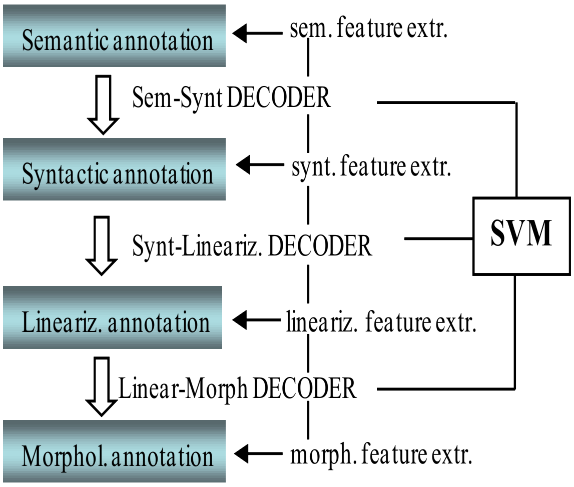


Figure 2: Realizer training scenario setup

1500 scripts for English and 2500 for German.

The training algorithms typically perform 6 iterations (*epochs*) over the training examples. For each training example, a minimal edit script is selected. If this script is different from the gold script, the features of the gold script are calculated and the weight vector of the SVM is adjusted according to the difference between the predicted vector and the *gold feature vector*. The classification task consists then in finding the classification script that maps the lemma to the correct word form. For this purpose, the classifier scores each of the minimal edit scripts according to the input, choosing the one with the highest score.

4 Sentence Generation

Sentence generation that starts from a given semantic structure as input consists in the application of the previously trained SVM decoders in sequence in order to realize the following sequence of mappings:

SemStr → *SyntStr* → *LinearStr* → *Surface*

4.1 Semantic Generation

Algorithm 3 shows the algorithm for semantic generation, i.e., the derivation of a dependency tree from a semantic structure. It is a beam search that creates a maximum spanning tree. In the first step, a seed tree consisting of one edge is built. In each of the subsequent steps, this tree is extended by one node. For the decision, which node is to be attached next and to which node, we consider the highest scoring options. This procedure works well since nodes that are close in the semantic structure are usually close in the syntactic tree as well. Therefore subtrees that contain those nodes are considered first.

Algorithm 2: Morphological realization training with MIRA

```
// $y_i, l_i$; $y_i$ is a dependency tree, $l_i$ lemmatized sentence
script-list ← {} //initialize the script-list
for $i \leftarrow 1$ to $|I|$ // iteration over the training examples
  for $l \leftarrow 1$ to $|l_i|$ do//// iteration over the lemmata of $l_i$
    lemma$_l$ ← lower-case ($l_i$,$l$)
    //ensure that all lemmata start with a lower case letter
    script ← compute-edit-dist-script(lemma$_l$, form($l_i$,$l$))
    if script $\notin$ script-list
      script-list ← script-list ∪ { script }
for $k \leftarrow 1$ to $E$ // $E$ = number of traininig epochs
  for $i \leftarrow 1$ to $|I|$ // iteration over the training examples
    for $l \leftarrow 1$ to $|l_i|$ do
      script$_p$ ← predict-script($l_i$,$y_i$,$l$)
      script$_g$ ← edit-dist-script(lemma$_l$, form($l_i$,$l$))
      if script$_p \neq$ script$_g$ then
      // update the weight vector $v$ and the vector $w$, which
      // averages over all collected weight vectors acc.
      // to diff. of the predicted and gold feature vector
      update $w, v$ according to $\Delta(\phi(\text{script}_p), \phi(\text{script}_g))$
      //with $\phi(\text{script}_p)$, $\phi(\text{script}_g)$ as feature vectors of
      //script$_p$ and script$_g$, respectively
```

Unlike the traditional n-gram based stochastic realizers such as (Langkilde and Knight, 1998), we use for the score calculation structured features composed of the following elements: (i) the lemmata, (ii) the **dist**ance between the starting node s and the target node t, (iii) the **dir**ection of the path (if the path has a direction), (iv) the sorted **bag** of in-going edges labels without repitition, (v) the **path** of edge labels between source and target node.

The composed structured features are:

- label+dist(s, t)+dir
- label+dist(s, t)+lemma$_s$+dir
- label+dist(s, t)+lemma$_t$+dir
- label+dist(s, t)+lemma$_s$+lemma$_t$+dir
- label+dist(s, t)+bag$_s$+dir
- label+dist(s, t)+bag$_t$+dir
- label+path(s, t)+dir

#	word-pairs(w$_1$,w$_2$)	#	n-grams
1	label$_{w1}$+label$_{w2}$	13	PoS$_1$+PoS$_2$+PoS$_3$
2	label$_{w1}$+lemma$_1$	14	PoS$_1$+PoS$_2$+PoS$_3$+dist
3	label$_{w1}$+lemma$_2$	15	lemma$_1$+lemma$_2$+lemma$_3$
4	label$_{w2}$+lemma$_1$	16	lemma$_1$+lemma$_2$+lemma$_3$+dist
5	label$_{w2}$+lemma$_2$	17	lemma$_1$+lemma$_3$+head(w1,w2,w3)
6	PoS$_1$+PoS$_2$	18	lemma$_1$+lemma$_3$+head(w1,w2,w3)+dist
7	PoS$_1$+PoS$_2$+head(w$_1$,w$_2$)	19	label$_1$+label$_2$+label$_3$+head(w1,w2,w3)
8	label$_{w1}$+label$_{w2}$+PoS$_1$+head(w$_1$,w$_2$)	20	label$_1$+label$_2$+label$_3$+head(w1,w2,w3)+dist
9	label$_{w1}$+label$_{w2}$+PoS$_2$+head(w$_1$,w$_2$)	21	label$_1$+label$_2$+label$_3$+lemma$_1$+PoS$_2$+head(w1,w2,w3)
10	label$_{w1}$+label$_{w2}$+PoS$_1$+PoS$_2$+head(w$_1$,w$_2$)	22	label$_1$+label$_2$+label$_3$+lemma$_1$+PoS$_2$+head(w1,w2,w3)+dist
11	label$_{w1}$+label$_{w2}$+PoS$_1$+#children$_2$+head(w$_1$,w$_2$)	23	label$_1$+label$_2$+label$_3$+lemma$_2$+PoS$_1$+head(w1,w2,w3)
12	label$_{w1}$+label$_{w2}$+PoS$_2$+#children$_1$+head(w$_1$,w$_2$)	24	label$_1$+label$_2$+label$_3$+lemma$_2$+PoS$_1$+head(w1,w2,w3)+dist
#	**global features for constituents**		
25	**if** $\lvert$constituent$\rvert$ > 1 **then** label$_{1st}$+label$_{last}$+label$_{last-1}$+PoS$_{first}$+PoS$_{last}$+PoS$_{head}$		
26	**if** $\lvert$constituent$\rvert$ > 2 **then** label$_{1st}$+label$_{2d}$+label$_{3d}$+PoS$_{last}$+PoS$_{last-1}$+PoS$_{head}$+contains-?		
27	**if** $\lvert$constituent$\rvert$ > 2 **then** label$_{1st}$+label$_{2d}$+label$_{3d}$+PoS$_{last}$+PoS$_{last-1}$+lemma$_{head}$+contains-?		
28	**if** $\lvert$constituent$\rvert$ > 3 **then** PoS$_{1st}$+PoS$_{2d}$+PoS$_{3d}$+PoS$_{4th}$+PoS$_{last}$+label$_{head}$+contains-?+pos-head		
29	**if** $\lvert$constituent$\rvert$ > 3 **then** PoS$_{last}$+PoS$_{last-1}$+PoS$_{last-2}$+PoS$_{last-3}$+PoS$_{first}$+label$_{head}$+contains-?+pos-head		
30	PoS$_{first}$+PoS$_{last}$+lemma$_{first}$+lemma$_{last}$+lemma$_{head}$+contains-?+pos-head		

Table 1: Feature schemas used for linearization (*label*$_w$ is the label of the in-going edge to a word w in the dependency tree; *lemma*$_w$ is the lemma of w, and *PoS*$_w$ is the part-of-speech tag of w; *head*(w$_1$,w$_2$, ...) is a function which is 1 if w_1 is the head, 2 if w_2 is the head, etc. and else 0; *dist* is the position within the constituent; *contains-?* is a boolean value which is true if the sentence contains a question mark and false otherwise; *pos-head* is the position of the head in the constituent)

4.2 Dependency Tree Linearization

Since we use unordered dependency trees as syntactic structures, our realizer has to find the optimal linear order for the lexemes of each dependency tree. Algorithm 4 shows our linearization algorithm. To order the dependency tree, we use a one classifier-approach for all languages—in contrast to, e.g., Filippova and Strube (2009), who use a two-classifier approach for German.[1]

The algorithm is again a beam search. It starts with an elementary list for each node of the dependency tree. Each elementary list is first extended by the children of the node in the list; then, the lists are extended stepwise by the children of the newly added nodes. If the number of lists during this procedure exceeds the threshold of 1000, the lists are sorted in accordance with their score, and the first 1000 are kept. The remaining lists are removed. Afterwards, the score of each list is adjusted according to a global score function which takes into account complex features such as the first word of a consitutent, last word, the head, and the edge label to the head (cf. Table 1 for the list of the features). Finally, the nodes of the dependency tree are ordered with respect to the highest ranked lists.

Only in a very rare case, the threshold of the beam search is exceeded. Even with a rich feature set, the procedure is very fast. The linearization takes about 3 milliseconds in average per dependency tree on a computer with a 2.8 Ghz CPU.

4.3 Morphological Realization

The morphological realization algorithm selects the edit script in accordance with the highest score for each lemma of a sentence obtained during training (see Algorithm 2 above) and applies then the scripts to obtain the word forms; cf. Algorithm 5.

Table 2 lists the feature schemas used for morphological realization.

5 Experiments

To evaluate the performance of our realizer, we carried out experiments on deep generation of Chinese, English, German and Spanish, starting from CoNLL '09 shared task corpora. The size of the test sets is listed in Table 3.[2]

[1] We decided to test at this stage of our work a uniform technology for all languages, even if the idiosyncrasies of some languages may be handled better by specific solutions.

[2] As in (Langkilde-Geary, 2002) and (Ringger et al., 2004), we used Section 23 of the WSJ corpus as test set for English.

Algorithm 3: Semantic generation

```
//$s_i$, $y$ semantic graph and its dependency tree
for i $\leftarrow$ 1 to $|I|$ // iteration over the training examples
  // build an initial tree
  for all $n_1 \in s_i$ do
  trees $\leftarrow$ {} // initialize the constructed trees list
    for all $n_2 \in s_i$ do
      if $n_1 \neq n_2$ then
        for all $l \in$ dependency-labels do
          trees = trees $\cup$ {(synt($n_1$),synt($n_2$),$l$)}
  trees $\leftarrow$ sort-trees-descending-to-score(trees)
  trees $\leftarrow$ look-forward(1000,sublist(trees,20))
  //assess at most 1000 edges of the 20 best trees
  tree $\leftarrow$ get-best-tree-due-to-score(trees)
  (s,t,l) $\leftarrow$ first-added-edge(tree)
  // create the best tree
  best-tree $\leftarrow$ (s,t,l)
  // compute the nodes that still need to be attached
  rest $\leftarrow$ nodes($s_i$) - $\{s, t\}$
  while rest $\neq \emptyset$ do
    trees $\leftarrow$ look-forward(1000,best-tree,rest)
    tree $\leftarrow$ get-best-tree-due-to-score(trees)
    (s,t,l) $\leftarrow$ first-added-edge(tree)
    best-tree $\leftarrow$ best-tree $\cup$ { (s,t,l) }
    if (root(s,best-tree)) then rest $\leftarrow$ rest - $\{s\}$
    else rest $\leftarrow$ rest - $\{t\}$
```

The performance of both the isolated stages and the realizer as a whole has been assessed.

5.1 Evaluation Metrics

In order to measure the correctness of the semantics to syntax mapping, we use the unlabeled and labeled attachment score as it commonly used in dependency parsing. The labeled attachment score (LAS) is the proportion of tokens that are assigned both the correct head and the correct edge label. The unlabeled attachment score (ULA) is the proportion of correct tokens that are assigned the correct head.

To assess the quality of linearization, we use three different evaluation metrics. The first metric is the per-phrase/per-clause accuracy (*acc snt.*), which facilitates the automatic evaluation of results:

$$acc = \frac{correct\ constituents}{all\ constituents}$$

As second evaluation metric, we use a metric related to the edit distance:

$$di = 1 - \frac{m}{total\ number\ of\ words}$$

(with m as the minimum number of deletions combined with insertions to obtain the correct order (Ringger et al., 2004)).

Algorithm 4: Dependency tree linearization

```
//$y_i$ a dependency tree
for i $\leftarrow$ 1 to $|I|$ // iteration over the training examples
  // iterate over all nodes of the dependency tree $y_i$
  for n $\leftarrow$ 1 to $|y_i|$ do
    $subtree_n$ $\leftarrow$ children(n) $\cup$ {n}
    ordered-lists$_n$ $\leftarrow$ {} // initialize
    for all m $\in$ $subtree_n$ do
      beam $\leftarrow$ {}
      for all $l \in$ ordered-lists do
        beam $\leftarrow$ beam  $\cup$ { append(clone($l$),m)}
      for all $l \in$ ordered-lists do
        score($l$) $\leftarrow$ compute-score-for-word-list($l$)
      sort-lists-descending-to-score(beam,score)
      if | beam | > beam-size then
        beam $\leftarrow$ sublist(0,1000,beam)
      ordered-lists$_n$ $\leftarrow$ beam
    $score_g(l)$ $\leftarrow$ $score(l)$ + compute-global-score($l$)
    sort-lists-descending-in-score(beam,score$_g$)
```

Algorithm 5: Morphological realization

```
// $y_i$ a dependency tree, and $l_i$ an ordered list of lemmata
for $l$ $\leftarrow$ 1 to $|l_i|$ do
  script$_p$ $\leftarrow$ predict-script($l_i$,$y_i$,$l$)
  form$_l$ $\leftarrow$ apply-edit-dist-script(lemma$_l$, script$_p$)
```

To be able to compare our results with (He et al., 2009) and (Ringger et al., 2004), we use the BLEU score as a third metric.

For the asessment of the quality of the word form generation, we use the accuracy score. The accuracy is the ratio between correctly generated word forms and the entire set of generated word forms.

For the evaluation of the sentence realizer as a whole, we use the BLEU metric.

5.2 Experimental Results

Table 4 displays the results obtained for the isolated stages of sentence realization and of the realization as a whole, with reference to a baseline and to some state-of-the-art works. The baseline is the deep sentence realization over all stages starting from the original semantic annotation in the CoNLL ’09 shared task corpora.

Note, that our results are not fully comparable with (He et al., 2009; Filippova and Strube, 2009) and (Ringger et al., 2004), respectively, since the data are different. Furthermore, Filippova and Strube (2009) linearize only English sentences

#	**features**
1	es+lemma
2	es+lemma+m.feats
3	es+lemma+m.feats+POS
4	es+lemma+m.feats+POS+position
5	es+lemma+(lemma+1)+m.feats
6	es+lemma+(lemma+1)+POS
7	es+lemma+(m.feats-1)+(POS-1)
8	es+lemma+(m.feats-1)+(POS-1)+position
9	es+m.feats+(m.feats-1)
10	es+m.feats+(m.feats+1)
11	es+lemma+(m.feats-1)
12	es+m.feats+(m.feats-1)+(m.feats-2)
13	es+m.feats+POS
14	es+m.feats+(m.feats+1)
15	es+m.feats+(m.feats+1)+lemma
16	es+m.feats
17	es+e0+e1+m.feats
18	es+e0+e1+e2+m.feats
19	es+e0+e1+e2+e3+m.feats
20	es+e0+e1+e2+e3+e4+m.feats
21	es+e0+m.feats

Table 2: Feature schemas used for morphological realization

Chinese	English	German	Spanish
2556	2400	2000	1725

Table 3: The number of sentences in the test sets used in the experiments

that do not contain phrases that exceed 20,000 linearization options—which means that they filter out about 1% of the phrases.

For Spanish, to the best of our knowledge, no linearization experiments have been carried out so far. Therefore, we cannot contrast our results with any reference work.

As far as morphologization is concerned, the performance achieved by our realizer for English is somewhat lower than in (Minnen et al., 2001) (97.8% vs. 99.8% of accuracy). Note, however, that Minnen et al. describe a combined analyzer-generator, in which the generator is directly derived from the analyzer, which makes both approaches not directly comparable.

5.3 Discussion

The overall performance of our SVM-based deep sentence generator ranges between 0.611 (for German) and 0.688 (for Chinese) of the BLEU score. HALogen's (Langkilde-Geary, 2002) scores range between 0.514 and 0.924, depending on the completeness of the input. The figures are not directly comparable since HALogen takes as input syntactic structures. However, it gives us an idea where our generator is situated.

Traditional linearization approaches are rule-based; cf., e.g., (Bröker, 1998; Gerdes and Kahane, 2001; Duchier and Debusmann, 2001), and (Bohnet, 2004). More recently, statistic language models have been used to derive word order, cf. (Ringger et al., 2004; Wan et al., 2009) and (Filippova and Strube, 2009). Because of its partially free order, which is more difficult to handle than fixed word order, German has often been worked with in the context of linearization. Filippova and Strube (2009) adapted their linearization model originally developed for German to English. They use two classifiers to determine the word order in a sentence. The first classifier uses a trigram LM to order words within constituents, and the second (which is a maximum entropy classifier) determines the order of constituents that depend on a finite verb. For English, we achieve with our SVM-based classifier a better performance. As mentioned above, for German, Filippova and Strube (2009)'s two classifier approach pays off because it allows them to handle non-projective structures for the *Vorfeld* within the field model. It is certainly appropriate to optimize the performance of the realizer for the languages covered in a specific application. However, our goal has been so far different: to offer an off-the-shelf language-independent solution.

The linearization error analysis, first of all of German and Spanish, reveals that the annotation of coordinations in corpora of these languages as ‘X ← *and*/*or*/. . . → Y’ is a source of errors. The “linear” annotation used in the PropBank (‘X → *and*/*or*/. . . → Y’) appears to facilitate higher quality linearization. A preprocessing stage for automatic conversion of the annotation of coordinations in the corpora would have certainly contributed to a higher quality. We refrained from doing this because we did not want to distort the figures.

The morphologization error analysis indicates a number of error sources that we will address in the process of the improvement of the model. Among those sources are: quotes at the beginning of a sentence, acronyms, specific cases of starting capital letters of proper nouns (for English and Spanish), etc.

	Chinese	English	German	Spanish
Semantics-Syntax (ULA/LAS)	95.71/86.29	94.77/89.76	95.46/82.99	98.39/93.00
Syntax-Topology (di/acc)	0.88/64.74	0.91/74.96	0.82/50.5	0.83/52.77
Syntax-Topology (BLEU)	0.85	0.894	0.735	0.78
Topology-Morphology (accuracy=correct words/all words)	–	97.8	97.49	98.48
All stages (BLEU)	0.688	0.659	0.611	0.68
Baseline (BLEU)	0.12	0.18	0.11	0.14
Syntax-Topology (He et al., 2009) (di/acc)	0.89/–	–	–	–
Syntax-Topology (He et al., 2009) (BLEU)	0.887	–	–	–
Syntax-Topology (Filippova and Strube, 2009) (di/acc)	–	0.88/67	0.87/61	–
Syntax-Topology (Ringger et al., 2004) (BLEU)	–	0.836	–	–

Table 4: Quality figures for the isolated stages of deep sentence realization and the complete process.

As far as the contrastive evaluation of the quality of our morphologization stage is concerned, it is hampered by the fact that for the traditional manually crafted morphological generators, it is difficult to find thorough quantitative evaluations, and stochastic morphological generators are rare.

As already repeatedly pointed out above, so far we intentionally refrained from optimizing the individual realization stages for specific languages. Therefore, there is still quite a lot of room for improvement of our realizer when one concentrates on a selected set of languages.

6 Conclusions

We presented an SVM-based stochastic deep multilingual sentence generator that is inspired by the state-of-the-art research in semantic parsing. It uses similar techniques and relies on the same resources. This shows that there is a potential for stochastic sentence realization to catch up with the level of progress recently achieved in parsing technologies.

The generator exploits recently available multilevel-annotated corpora for training. While the availability of such corpora is a condition for deep sentence realization that starts, as is usually the case, from semantic (predicate-argument) structures, we discovered that current annotation schemata do not always favor generation such that additional preprocessing is necessary. This is not surprising since stochastic generation is a very young field. An initiative of the generation community would be appropriate to influence future multilevel annotation campaigns or to feed back the enriched annotations to the "official" resources.[3]

The most prominent features of our generator are that it is *per se* multilingual, it achieves an extremely broad coverage, and it starts from abstract semantic structures. The last feature allows us to cover a number of critical generation issues: sentence planning, linearization and morphological generation. The separation of the semantic, syntactic, linearization and morphological levels of annotation and their modular processing by separate SVM decoders also facilitates a subsequent integration of other generation tasks such as referring expression generation, ellipsis generation, and aggregation. As a matter of fact, this generator instantiates the Reference Architecture for Generation Systems (Mellish et al., 2006) for linguistic generation.

A more practical advantage of the presented deep stochastic sentence generator (as, in principle, of all stochastic generators) is that, if trained on a representative corpus, it is domain-independent. As rightly pointed out by Belz (2008), traditional wide coverage realizers such as KPML (Bateman et al., 2005), FUF/SURGE (Elhadad and Robin, 1996) and RealPro (Lavoie and Rambow, 1997), which were also intended as off-the-shelf plug-in realizers still tend to require a considerable amount of work for integration and fine-tuning of the grammatical and lexical resources. Deep stochastic sentence realizers have the potential to become real off-the-shelf modules. Our realizer is freely available for download at `http://www.recerca.upf.edu/taln`.

[3] We are currently working on a generation-oriented multilevel annotation of corpora for a number of languages. The corpora will be made available to the community.

Acknowledgments

Many thanks to the three anonymous reviewers for their very valuable comments and suggestions.

References

Bangalore, S. and O. Rambow. 2000. Exploiting a Probabilistic Hierarchical Model for Generation. In *Proceedings of COLING '00*, pages 42–48.

Bangalore, S., J. Chen, and O. Rambow. 2001. Impact of Quality and Quantity of Corpora on Stochastic Generation. In *Proceedings of the EMNLP Conference*, pages 159–166.

Bateman, J.A., I. Kruijff-Korbayová, and G.-J. Kruijff. 2005. Multilingual Resource Sharing Across Both Related and Unrelated Languages: An Implemented, Open-Source Framework for Practical Natural Language Generation. *Research on Language and Computation*, 15:1–29.

Belz, A. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*, 14(4):431–455.

Bohnet, B. 2004. A graph grammar approach to map between dependency trees and topological models. In *Proceedings of the IJCNLP*, pages 636–645.

Bröker, N. 1998. Separating Surface Order and Syntactic Relations in a Dependency Grammar. In *Proceedings of the COLING/ACL '98*.

Crammer, K., O. Dekel, S. Shalev-Shwartz, and Y. Singer. 2006. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585.

Duchier, D. and R. Debusmann. 2001. Topological dependency trees: A constraint-based account of linear precedence. In *Proceedings of the ACL*.

Elhadad, M. and J. Robin. 1996. An overview of SURGE: A reusable comprehensive syntactic realization component. Technical Report TR 96-03, Department of Mathematics and Computer Science, Ben Gurion University.

Filippova, K. and M. Strube. 2008. Sentence fusion via dependency graph compression. In *Proceedings of the EMNLP Conference*.

Filippova, K. and M. Strube. 2009. Tree linearization in English: Improving language model based approaches. In *Proceedings of the NAACL '09 and HLT, Short Papers*, pages 225–228.

Gerdes, K. and S. Kahane. 2001. Word order in German: A formal dependency grammar using a topological hierarchy. In *Proceedings of the ACL*.

Hajič, J. et al. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of the CoNLL*.

He, W., H. Wang, Y. Guo, and T. Liu. 2009. Dependency based chinese sentence realization. In *Proceedings of the ACL and of the IJCNLP of the AFNLP*, pages 809–816.

Knight, K. and V. Hatzivassiloglou. 1995. Two-level, many paths generation. In *Proceedings of the ACL*.

Langkilde, I. and K. Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the COLING/ACL*, pages 704–710.

Langkilde-Geary, I. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the Second INLG Conference*, pages 17–28.

Lavoie, B. and O. Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the 5th Conference on ANLP*.

Levenshtein, V.I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics*, 10:707–710.

Mellish, C., D. Scott, L. Cahill, D. Paiva, R. Evans, and M. Reape. 2006. A reference architecture for natural language generation systems. *Natural Language Engineering*, 12(1):1–34.

Minnen, G., J. Carroll, and D. Pearce. 2001. Applied morphological processing for English. *Natural Language Engineering*, 7(3):207–223.

Oh, A.H. and A.I. Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of the ANL/NAACL Workshop on Conversational Systems*, pages 27–32.

Palmer, M., D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105.

Ringger, E., M. Gamon, R.C. Moore, D. Rojas, M. Smets, and S. Corston-Oliver. 2004. Linguistically informed statistical models of constituent structure for ordering in sentence realization. In *Proceedings of COLING*, pages 673–679.

Wan, S., M. Dras, Dale R., and C. Paris. 2009. Improving Grammaticality in Statistical Sentence Generation: Introducing a Dependency Spanning Tree Algorithm with an Argument Satisfaction Model. In *Proceedings of the EACL '09*, pages 852–860.

Towards an optimal weighting of context words based on distance

Bernard Brosseau-Villeneuve*#, Jian-Yun Nie*, Noriko Kando#
* Université de Montréal, Email: {brosseab, nie}@iro.umontreal.ca
# National Institute of Informatics, Email: {bbrosseau, kando}@nii.ac.jp

Abstract

Word Sense Disambiguation (WSD) often relies on a context model or vector constructed from the words that co-occur with the target word within the same text windows. In most cases, a fixed-sized window is used, which is determined by trial and error. In addition, words within the same window are weighted uniformly regardless to their distance to the target word. Intuitively, it seems more reasonable to assign a stronger weight to context words closer to the target word. However, it is difficult to manually define the optimal weighting function based on distance. In this paper, we propose a unsupervised method for determining the optimal weights for context words according to their distance. The general idea is that the optimal weights should maximize the similarity of two context models of the target word generated from two random samples. This principle is applied to both English and Japanese. The context models using the resulting weights are used in WSD tasks on Semeval data. Our experimental results showed that substantial improvements in WSD accuracy can be obtained using the automatically defined weighting schema.

1 Introduction

The meaning of a word can be defined by the words that accompany it in the text. This is the principle often used in previous studies on Word Sense Disambiguation (WSD) (Ide and Véronis, 1998; Navigli, 2009). In general, the accompanying words form a context vector of the target word, or a probability distribution of the context words. For example, under the unigram bag-of-words assumption, this means building $p(x|t) = \frac{count(x,t)}{\sum_{x'} count(x',t)}$, where $count(x,t)$ is the count of co-occurrences of word x with the target word t under a certain criterion. In most studies, x and t should co-occur within a window of up to k words or sentences. The bounds are usually selected in an ad-hoc fashion to maximize system performance. Occurrences inside the window often weight the same without regard to their position. This is counterintuitive. Indeed, a word closer to the target word generally has a greater semantic constraint on the target word than a more distant word. It is however difficult to define the optimal weighting function manually. To get around this, some systems add positional features for very close words. In information retrieval, to model the strength of word relations, some studies have proposed non-uniform weighting methods of context words, which decrease the importance of more distant words in the context vector. However, the weighting functions are defined manually. It is unclear that these functions can best capture the impact of the context words on the target word.

In this paper, we propose an unsupervised method to automatically learn the optimal weight of a word according to its distance to the target word. The general principle used to determine such weight is that, if we randomly determine two sets of windows containing the target word from the same corpus, the meaning – or mixture of meanings for polysemic words – of the target word in the two sets should be similar. As the context model – a probability distribution for the context words – determines the meaning of the target word, the context models generated from the two sets should also be similar. The weights of context words at different distance are therefore determined so as to maximize the similarity of context models generated from the two sets of samples. In this paper, we propose a gradient descent method to find the optimal weights. We will see that the optimal weighting functions are different from those used in previous studies. Experimentation on Semeval-2007 English and Semeval-2010 Japanese lexical sample task data shows that improvements can be attained using the resulting weighting functions on simple Naïve Bayes (NB) systems in comparison to manually selected functions. This result validates the general principle we propose in this paper.

The remainder of this paper is organized as follows: typical uses of text windows and related work are presented in Section 2. Our method is presented in Section 3. In Section 4 to 6, we show experimental results on English and Japanese WSD. We conclude in Section 7 with discussion and further possible extensions.

2 Uses of text windows

Modeling the distribution of words around one target word, which we call context model, has many uses. For instance, one can use it to define a co-occurrence-based stemmer (Xu and Croft, 1998), which uses window co-occurrence statistics to calculate the best equivalence classes for a group of word forms. In the study of Xu and Croft, they suggest using windows of up to 100 words. Context models are also widely used in WSD. For example, top performing systems on English WSD tasks in Semeval-2007, such as NUS-ML (Cai et al., 2007), all made use of bag-of-words features around the target word. In this case, they found that the best results can be achieved using a window size of 3.

Both systems limit the size of their windows for different purposes. The former uses a large size in order to model the topic of the documents containing the word rather than the word's meaning. The latter would limit the size because bag-of-words features further from the target word would not be sufficiently related to its meaning (Ide and Véronis, 1998). We see that there is a compromise between taking fewer, highly related words, or taking more, lower quality words. However, there is no principled way to determine the optimal size of windows. The size is determined by trial and error.

A more questionable aspect in the above systems is that for bag-of-words features, all words in a window are given equal weights. This is counterintuitive. One can easily understand that a context word closer to the target word *generally* imposes a stronger constraint on the meaning of the latter, than a more distant context word. It is then reasonable to define a weighting function that decreases along with distance. Several studies in information retrieval (IR) have proposed such functions to model the strength of dependency between words. For instance, Gao et al. (2002) proposed an exponential decay function to capture the strength of dependency between words. This function turns out to work better than the uniform weighting in the IR experiments.

Song and Bruza (2003) used a fixed-size sliding window to determine word co-occurrences. This is equivalent to define a linear decay function for context words. The context vectors defined this way are used to estimate similarity between words. A use of the resulting similarity in query expansion in IR turned out to be successful (Bai et al., 2005).

In a more recent study, Lv and Zhai (2009) evaluated several kernel functions to determine the weights of context words according to distance, including Gaussian kernel, cosine kernel, and so on. As for the exponential and linear decaying functions, all these kernel functions have fixed shapes, which are determined manually.

Notice that the above functions have only been tested in IR experiments. It is not clear how these functions perform in WSD. More importantly, all the previous studies have investigated only a limited number of weighting functions for context words. Although some improvements using these functions have been observed in IR, it is not clear whether the functions can best capture the true impact of the context words on the meaning of the target word. Although the proposed functions comply with the general principle that closer words are more important than more distant words, no principled way has been proposed to determine the particular shape of the function for different languages and collections.

In this paper, we argue that there is indeed a hidden weighting function that best capture the impact of context words, but the function cannot be defined manually. Rather, the best function should be the one that emerges naturally from the data. Therefore, we propose an unsupervised method to discover such a function based on the following principle: the context models for a target word generated from two random samples should be similar. In the next section, we will define in detail how this principle is used.

3 Computing weights for distances

In this section, we present our method for choosing how much a word occurrence should count in the context model according to its distance to the target word. In this study, for simplicity, we assume that all word occurrences at a given distance count equally in the context model. That is, we ignore other features such as POS-tags, which are used in other studies on WSD.

Let $\mathcal{C}$ be a corpus, W a set of text windows for the target word w, $c_{W,i,x}$ the count of occurrences of word x at distance i in W, $c_{W,i}$ the sum of these counts, and α_i the weight put on one word occurrence at distance i. Then,

$$P_{ML,W}(x) = \frac{\sum_i \alpha_i c_{W,i,x}}{\sum_i \alpha_i c_{W,i}} \qquad (1)$$

is the maximum likelihood estimator for x in the context model of w. To counter the zero probability problem, we apply Dirichlet smoothing with the collection language model as a prior:

$$P_{Dir,W}(x) = \frac{\sum_i \alpha_i c_{W,i,x} + \mu_W P(x|\mathcal{C})}{\sum_i \alpha_i c_{W,i} + \mu_W} \qquad (2)$$

The pseudo-count μ_W can be a constant, or can be found by using Newton's method, maximizing the log likelihood via leave-one-out estimation:

$$\mathcal{L}_{-1}(\mu|W,\mathcal{C}) = \\ \sum_i \sum_{x \in V} \alpha_i c_{W,i,x} \log \frac{\alpha_i c_{W,i,x} - \alpha_i + \mu P(x|\mathcal{C})}{\sum_j \alpha_j c_{W,j} - \alpha_i + \mu}$$

The general process, which we call automatic Dirichlet smoothing, is similar to that described in (Zhai and Lafferty, 2002).

To find the best weights for our model we propose the following process:

- Let T be the set of all windows containing the target word. We randomly split this set into two sets A and B.

- We want to find $\alpha^\star$ that maximizes the similarity of the models obtained from the two sets, by minimizing their mutual cross entropy:

$$\begin{aligned} l(\alpha) &= H(P_{ML,A}, P_{Dir,B}) + \\ & \quad H(P_{ML,B}, P_{Dir,A}) \end{aligned} \qquad (3)$$

In other words, we want α_i to represent how much an occurrence at distance i models the context better than the collection language model, whose counts are weighted by the Dirichlet parameter. We hypothesize that target words occur in limited contexts, and as we get farther from them, the possibilities become greater, resulting in sparse and less related counts. Since two different sets of the same word are essentially noisy samples of the same distribution, the weights maximizing their mutual generation probabilities should model this phenomenon.

One may wonder why we do not use a distribution similarity metric such as Kullback–Leibler (KL) divergence or Information Radius (IRad). The reason is that with enough word occurrences (big windows or enough samples), the most similar distributions are found with uniform weights, when all word counts are used. KL divergence is especially problematic as, since it requires smoothing, the weights will converge to the degenerate weights $\alpha = 0$, where only the identical smoothing counts remain. Entropy minimization is therefore needed in the objective function.

To determine the optimal weight of α_i, we propose a simple gradient descent minimizing (3) over α. The following are the necessary derivatives:

$$\begin{aligned} \frac{\partial l}{\partial \alpha_i} &= \frac{\partial H(P_{ML,A}, P_{Dir,B})}{\partial \alpha_i} + \\ & \quad \frac{\partial H(P_{ML,B}, P_{Dir,A})}{\partial \alpha_i} \end{aligned}$$

$$\frac{\partial H\left(P_{ML,W}, P_{Dir,(T-W)}\right)}{\partial \alpha_i} =$$

$$- \sum_{x \in V} \left[\frac{\partial P_{ML,W}(x)}{\partial \alpha_i} \log P_{Dir,(T-W)}(x) + \frac{\partial P_{Dir,(T-W)}(x)}{\partial \alpha_i} \times \frac{P_{ML,W}(x)}{P_{Dir,(T-W)}(x)} \right]$$

$$\frac{\partial P_{ML,W}(x)}{\partial \alpha_i} = \frac{c_{W,i,x} - P_{ML,W}(x) c_{W,i}}{\sum_j \alpha_j c_{W,j}}$$

$$\frac{\partial P_{Dir,W}(x)}{\partial \alpha_i} = \frac{c_{W,i,x} - P_{Dir,W}(x) c_{W,i}}{\sum_j \alpha_j c_{W,j} + \mu_W}$$

We use stochastic gradient descent: one word is selected randomly, it's gradient is computed, a small gradient step is done and the process is repeated. A pseudo-code of the process can be found in Algorithm 1.

Algorithm 1 LearnWeight($\mathcal{C}, \eta, \epsilon$)

```
$\alpha \leftarrow 1^k$
**repeat**
  $T$ ←{Get windows for next word}
  $(A, B)$ ←RandomPartition($T$)
  **for** $W$ in $A, B$ **do**
    $P_{ML,W}$ ←MakeML($W$,$\alpha$)
    $\mu_W$ ←ComputePseudoCount($W$,$\mathcal{C}$)
    $P_{Dir,W}$ ←MakeDir( $P_{ML,W}, \mu_W, \mathcal{C}$)
  **end for**
  $grad \leftarrow \nabla H(P_{ML,A}, P_{Dir,B}) + \nabla H(P_{ML,B}, P_{Dir,A})$
  $\alpha \leftarrow \alpha - \eta \frac{grad}{\|grad\|}$
**until** $\exists \alpha_i < \epsilon$
**return** $\alpha / \max\{\alpha_i\}$
```

Now, as the objective function would eventually go towards putting nearly all weight on α_1, we hypothesize that the farthest distances should have a near-zero contribution, and determine the stop criterion as having one weight go under a small threshold. Alternatively, a control set of held out words can be used to observe the progress of the objective function or the gradient length. When more and more weight is put on the few closest positions, the objective function and gradient depends on less counts and will become less stable. This can be used as a stop criterion.

The above weight learning process is applied on an English collection and a Japanese collection with $\eta = \epsilon = 0.001$, and $\mu = 1000$. In the next sections, we will describe both resulting weighting functions in the context of WSD experiments.

4 Classifiers for supervised WSD tasks

Since we use the same systems for both English and Japanese experiments, we will briefly discuss the used classifiers in this section. In both tasks, the objective is to maximize WSD accuracy on held-out data, given that we have a set of training text passages containing a sense-annotated target word.

The first of our baselines, the *Most Frequent Sense* (MFS) system always selects the most frequent sense in the training set. It gives us a lower bound on system accuracies.

Naïve Bayes (NB) classifiers score classes using the Bayes formula under a feature independence assumption. Let w be the target word in a given window sample to be classified, the scoring formula for sense class S is:

$$Score(w, S) = P(S) P_{Tar}(w|S)^{\lambda_{Tar}} \times \prod_{x_i \in context(w)} P_{Con}(x_i|S)^{\lambda_{Con} \alpha_{dist(x_i)}}$$

where $dist(x_i)$ is the distance between the context word x_i and the target word w. The target word being an informative feature present in all samples, we use it in a target word language model P_{Tar}. The surrounding words are summed in the context model P_{Con} as shown in equation (1). As we can see with the presence of α in the equation, the scoring follows the same weighting scheme as we do when accumulating counts, since the samples to classify follow the same distribution as the training ones. Also, when a language model uses automatic Dirichlet smoothing, the impact of the features against the prior is controlled with the manual parameters λ_{Tar} or λ_{Con}. When a manual smoothing parameter is used, it also handles impact control. Our systems use the following weight functions:

Uniform: $\alpha_i = \mathbf{1}_{1 \le i \le \delta}$, where δ is a window size and $\mathbf{1}$ the indicator function.

Linear: $\alpha_i = \max\{0, 1 - (i-1)\delta\}$, where δ is the decay rate.

Exponential: $\alpha_i = e^{-(i-1)\delta}$, where δ is the exponential parameter.

Learned: α_i is the weight learned as shown previously.

The parameters for NB systems are identical for all words of a task and were selected by exhaustive search, maximizing leave-one-out accuracy on the training set. For each language model, we tried Laplace, manual Dirichlet and automatic Dirichlet smoothing.

For the sake of comparison, also we provide a *Support Vector Machine* (SVM) classifier, which produces the best results in Semeval 2007. We used libSVM with a linear kernel, and regularization parameters were selected via grid search maximizing leave-one-out accuracy on the training set. We tested the following windows limits: all words in sample, current sentence, and various fixed window sizes. We used the same features as the NB systems, testing Boolean, raw count, log-of-counts and counts from weight functions representations. Although non-Boolean features had good leave-one-out precision on the training data, since SVM does not employ smoothing, only Boolean features kept good results on test data, so our SVM baseline uses Boolean features.

5 WSD experiments on Semeval-2007 English Lexical Sample

The Semeval workshop holds WSD tasks such as the English Lexical Sample (ELS) (Pradhan et al., 2007). The task is to maximize WSD accuracy on a selected set of polysemous words, 65 verbs and 35 nouns, for which passages were taken from the WSJ Tree corpus. Passages contain a couple of sentences around the target word, which is manually annotated with a sense taken from OntoNotes (Hovy et al., 2006). The sense inventory is quite coarse, with an average of 3.6 senses per word. Instances count are listed in Table 1.

	Train	Test	Total
Verb	8988	2292	11280
Noun	13293	2559	15852
Total	22281	4851	

Table 1: Number of instances in the ELS data

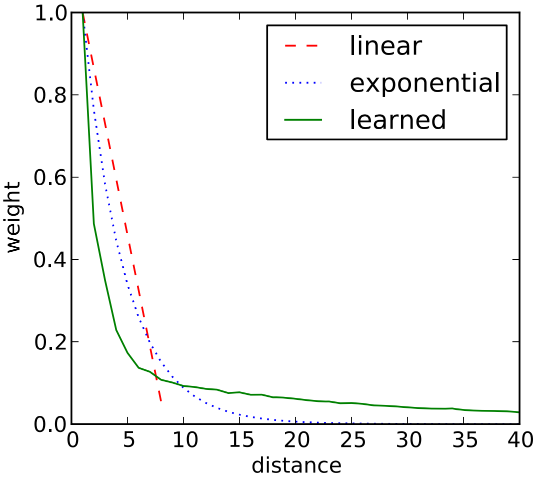


Figure 1: Weight curve for AP88-90

Since there are only 100 target words and instances are limited in the Semeval collection, we do not have sufficient samples to estimate the optimal weights for context words. Therefore, we used the AP88-90 corpus of the TREC collection (CD 1 & 2) in our training process. The AP collection contains 242,918 documents. Since our classifiers use word stems, the collection was also stemmed with the Porter stemmer and sets of windows were built for all word stems. To get near-uniform counts in all distances, only full windows with a size of 100, which was considered big enough without any doubt, were kept. In order to get more samples, windows to the right and to the left were separated. For each target word, we used 1000 windows. A stoplist of the top 10 frequent words was used, but place holders were left in the windows to preserve the distances. Multiple consecutive stop words (ex: “of the”) were merged, and the target word stem, being the same for all samples of a set, was ignored in the construction of context models. The AP collection results in 32,650 target words containing 5,870,604 windows. The training process described in Section 3 is used to determine the best weights of context words. Figure 1 shows the first 40 elements of the resulting weighting function curve.

As we can see, the curve is neither exponential, linear, or any of the forms used by Lv and Zhai. Its form is rather similar to $x^{-\delta}$, or rather $\log^{-1}(\delta + x)$ minus some constant. The decrease

System	Cross-Val (%)	Test set (%)
MFS	78.66	77.76
Uniform NB	86.04	84.52
SVM	85.53	85.03
Linear NB	86.89	85.71
Exp. NB	87.80	86.23
Learned NB	88.46	86.70

Table 2: WSD accuracy on Semeval-2007 ELC

rate is initially very high and then reduces as it becomes closer to zero. This long tail is not present in any of the previously suggested functions. The large difference between the above optimal weighting function and the functions used in previous studies would indicate that the latter are suboptimal. Also, as we can see, the relation between context words and the target word is mostly gone after a few words. This would motivate the commonly used very small windows when using a uniform weights, since using a bigger window would further widen the gap between the used weight and the optimal ones.

Now for the system settings, the context words were processed the same way as the external corpus. The target word was used without stemming but had the case stripped. The NB systems used the concatenation of the AP collection and the Semeval data for the collection language model. This is motivated by the fact that the Semeval data is not balanced: it contains only a small number of passages containing the target words. This makes words related to them unusually frequent. The class priors used an absolute discounting of 0.5 on class counts. *Uniform NB* uses a window of size 4, a Laplace smoothing of 0.65 on P_{Tar} and an automatic Dirichlet with $\lambda_{Con} = 0.7$ on P_{Con}. *Linear NB* has $\delta = 0.135$, uses a Laplace smoothing of 0.85 on P_{Tar} and an automatic Dirichlet with $\lambda_{Con} = 0.985$ on P_{Con}. *Exp NB* has $\delta = 0.27$, uses a Laplace smoothing of 2.8 on P_{Tar} and an automatic Dirichlet with $\lambda_{Con} = 1.01$ on P_{Con}. The *SVM* system uses a window of size 3. Our system, *Learned NB* uses a Laplace smoothing of 1.075 on P_{Tar}, and an automatic Dirichlet with $\lambda_{Con} = 1.025$ on P_{Con}. The results on WSD are listed in Table 2. WSD accuracy is measured by the proportion of correctly disambiguated words among all the word samples. The cross-validation is performed on the training data with leave-one-out and is shown as a hint of the capacity of the models. A randomization test comparing *Exponential NB* and *Learned NB* gives a p-value of 0.0508, which is quite good considering the extensive trials used to select the exponential parameter in comparison to a single curve computed from a different corpus. This performance is comparable to the current state of the art. It outperforms most of the systems participating in the task (Pradhan et al., 2007). Out of 14 systems, the best results had accuracies of 89.1*, 89.1*, 88.7, 86.9 and 86.4 (* indicates post-competition submissions). Notice that most previous systems used SVM with additional features such as local collocations, positional word features and POS tags. Our approach only uses bag-of-words in a Naïve Bayes classifier. Therefore, the performance of our method is sub-optimal. With additional features and better classification methods, we can expect that better performance can be obtained. In future work, we will investigate the applications of SVM with our new term weighting scheme, together with additional types of features.

6 WSD experiments on Semeval-2010 Japanese Lexical Sample

The Semeval-2010 Japanese WSD task (Okumura et al., 2010) consists of 50 polysemous words for which examples were taken from the BCCWJ corpus (Maekawa, 2008). It was manually segmented, POS-tagged, and annotated with senses taken from the Iwanami Kokugo dictionary. The selected words have 50 samples for both the training and test set. The task is identical to the ELS of the previous experiment.

Since the data was again insufficient to compute the optimal weighting curve, we used the Mainichi-2005 corpus of NTCIR-8. We tried to reproduce the same kind of segmentation as the training data by using the Chasen parser with UniDic, which nevertheless results in different word segments as the training data. For the corpus and Semeval data, conjugations (setsuzoku-to, jodôshi, etc.), particles (all jo-shi), symbols (blanks, kigô, etc.), and numbers were stripped. When a

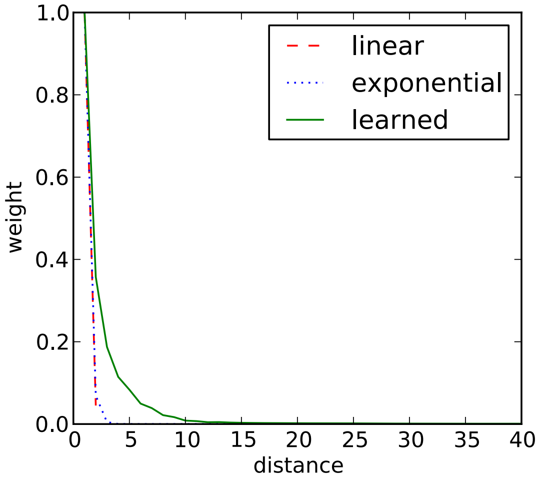


Figure 2: Weight curve for Mainichi 2005

System	Cross-Val (%)	Test set (%)
MFS	75.23	68.96
SVM	82.55	74.92
Uniform NB	82.47	76.16
Linear NB	82.63	76.48
Exp. NB	82.68	76.44
Learned NB	82.67	76.52

Table 3: WSD accuracy on Semeval-2010 JWSD

base-form reading was present (for verbs and adjectives), the token was replaced by the Kanjis (Chinese characters) in the word writing concatenated with the base-form reading. This treatment is somewhat equivalent to the stemming+stop list of the ELS tasks. The resulting curve can be seen in Figure 2.

As we can see, the general form of the curve is similar to that of the English collection, but is steeper. This suggests that the meaning of Japanese words can be determined using only the closest context words. Words further than a few positions away have very small impact on the target word. This can be explained by the grammatical structure of the Japanese language. While English can be considered a Subject-Verb-Complement language, Japanese is considered Subject-Complement-Verb. Verbs, mostly found at the end of a sentence, can be far apart from their subject, and vice versa. The window distance is therefore less useful to capture the relatedness in Japanese than in English since Japanese has more non-local dependencies.

The Semeval Japanese test data being part of a balanced corpus, untagged occurrences of the target words are plenty, so we can benefit from using the collection-level counts for smoothing. *Uniform NB* uses a window of size 1, manual Dirichlet smoothing of 4 for P_{Tar} and 90 for the P_{Con}. *Linear NB* has $\delta = 0.955$, uses a manual Dirichlet smoothing of 6.25 on P_{Tar} and manual Dirichlet smoothing with $\lambda_{Con} = 65$ on P_{Con}. *Exp NB* has $\delta = 2.675$, uses a manual Dirichlet smoothing of 6.5 on P_{Tar} and a manual Dirichlet of 70 on P_{Con}. The *SVM* system uses a window size of 1 and Boolean features. *Learned NB* used a manual Dirichlet smoothing of 4 for P_{Tar} and automatic Dirichlet smoothing with $\lambda_{Con} = 0.6$ for P_{Con}. We believe this smoothing is beneficial only on this system because it uses more words (the long tail), that makes the estimation of the pseudo-count more accurate. Results on WSD are listed in Table 3. As we can see, the difference between the NB models is less substantial than for English. This may be due to differences in the segmentation parameters of our external corpus: we used the human-checked segmentation found in the Semeval data for classification, but used a parser to segment our external corpus for weight learning. We are positive that the Chasen parser with the UniDic dictionary was used to create the initial segmentation in the Semeval data, but there may be differences in versions and the initial segmentation results were further modified manually.

Another reason for the results could be that the systems use almost the same weights: *Uniform NB* and *SVM* both used windows of size 1, and the Japanese curve is steeper than the English one, making the context model account to almost only immediately adjacent words. So, even if our context model contains more context words at larger distances, their weights are very low. This makes all context model quite similar. Nevertheless, we still observe some gain in WSD accuracy. These results show that the curves work as expected even in different languages. However, the weighting curve is strongly language-dependent. It could also be collection-dependent – we will investigate this aspect in the future, using different collections.

7 Conclusions

The definition of context vector and context model is critical in WSD. In previous studies in IR, decaying weight along with distance within a text window have been proposed. However, the decaying functions are defined manually. Although some of the functions produced better results than the uniform weighting, there is no evidence showing that these functions best capture the impact of the context words on the meaning of the target word. This paper proposed an unsupervised method for finding optimal weights for context words according to their distance to the target word. The general idea was to find the weights that best fit the data, in such a way that the context models for the same target word generated from two random windows samples become similar. It is the first time that this general principle is used for this purpose. Our experiments on WSD in English and Japanese suggest the validity of the principle.

In this paper, we limited context models to bag-of-words features, excluding additional features such as POS-tags. Despite this simple type of feature and the use of a simple Naïve Bayes classifier, the WSD accuracy we obtained can rival the other state-of-the-art systems with more sophisticated features and classification algorithms. This result indicates that a crucial aspect in WSD is the definition of an appropriate context model, and our weighting method can generate more reasonable weights of context words than using a predefined decaying function.

Our experiments also showed that the optimal weighting function is language-dependent. We obtained two different functions for English and Japanese, although their general shapes are similar. In fact, the optimal weighting function reflects the linguistic properties: as dependent words in Japanese can be further away from the target word due to its linguistic structure, the optimal weighting quickly decays, meaning that we can rely less on distant context words. This also shows a limitation of this study: distance is not the sole criterion to determine the impact of a context word. Other factors, such as POS-tag and syntactic dependency, can play an important role in the context model. These additional factors are complementary to the distance criterion and our approach can be extended to include such additional features. This extension is part of our future work.

Another limitation of straight window distance is that all words introduce the same distance, regardless of their nature. In our experiments, to make the distance a more sensible metric, we merged consecutive stop words in one placeholder token. The idea behind this it that some words, such as stop words, should introduce less distance than others. On the opposite, we can easily understand that tokens such as commas, full stops, parentheses and paragraph should introduce a bigger distance than regular words. We could therefore use a *congruence* score for a word, an indicator showing on average how much what comes before is similar to what comes after the word.

Also, we have combined our weighting schema with NB classifier. Other classifiers such as SVM could lead to better results. The utilization of our new weighting schema with SVM is another future work.

Finally, the weights computed with our method has been used in WSD tasks. The weights could be seen as the expected strength of relation between two words in a document according to their distance. The consideration of word relationships in documents and queries is one of the endeavors in current research in IR. The new weighting schema could be easily integrated with a dependency model in IR. We plan to perform such integration in the future.

Acknowledgments

The authors would like to thank Florian Boudin and Satoko Fujisawa for helpful comments on this work. This work is partially supported by Japanese MEXT Grant-in-Aid for Scientific Research on Info-plosion (#21013046) and the Japanese MEXT Research Student Scholarship program.

References

Bai, Jing, Dawei Song, Peter Bruza, Jian-Yun Nie, and Guihong Cao. 2005. Query expansion using term relationships in language models for information retrieval. In *CIKM '05 Proceedings*, pages 688–695, New York, NY, USA. ACM.

Cai, Jun Fu, Wee Sun Lee, and Yee Whye Teh. 2007. Nus-ml: improving word sense disambiguation using topic features. In *SemEval '07 Proceedings*, pages 249–252, Morristown, NJ, USA. Association for Computational Linguistics.

Cheung, Percy and Pascale Fung. 2004. Translation disambiguation in mixed language queries. *Machine Translation*, 18(4):251–273.

Gao, Jianfeng, Ming Zhou, Jian-Yun Nie, Hongzhao He, and Weijun Chen. 2002. Resolving query translation ambiguity using a decaying co-occurrence model and syntactic dependence relations. In *SIGIR '02 Proceedings*, pages 183–190, New York, NY, USA. ACM.

Ide, Nancy and Jean Véronis. 1998. Introduction to the special issue on word sense disambiguation: the state of the art. *Comput. Linguist.*, 24(1):2–40.

Lv, Yuanhua and ChengXiang Zhai. 2009. Positional language models for information retrieval. In *SIGIR '09 Proceedings*, pages 299–306, New York, NY, USA. ACM.

Maekawa, Kikuo. 2008. Compilation of the balanced corpus of contemporary written japanese in the kotonoha initiative (invited paper). In *ISUC '08 Proceedings*, pages 169–172, Washington, DC, USA. IEEE Computer Society.

Navigli, Roberto. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2):1–69.

Okumura, Manabu, Kiyoaki Shirai, Kanako Komiya, and Hikaru Yokono. 2010. Semeval-2010 task: Japanese wsd. In *SemEval '10 Proceedings*. Association for Computational Linguistics.

Pradhan, Sameer S., Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. Semeval-2007 task 17: English lexical sample, srl and all words. In *SemEval '07 Proceedings*, pages 87–92, Morristown, NJ, USA. Association for Computational Linguistics.

Song, D. and P. D. Bruza. 2003. Towards context sensitive information inference. *Journal of the American Society for Information Science and Technology*, 54(4):321–334.

Xu, Jinxi and W. Bruce Croft. 1998. Corpus-based stemming using cooccurrence of word variants. *ACM Trans. Inf. Syst.*, 16(1):61–81.

Zhai, ChengXiang and John Lafferty. 2002. Two-stage language models for information retrieval. In *SIGIR '02 Proceedings*, pages 49–56, New York, NY, USA. ACM.

Measuring the Non-compositionality of Multiword Expressions

Fan Bu and **Xiaoyan Zhu**
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Sci. and Tech., Tsinghua University
buf08@mails.tsinghua.edu.cn and zxy-dcs@tsinghua.edu.cn

Ming Li
David R. Cheriton School of Computer Science
University of Waterloo
mli@uwaterloo.ca

Abstract

Multiword Expressions (MWEs) appear frequently and ungrammatically in the natural languages. Identifying MWEs in free texts is a very challenging problem.

This paper proposes a knowledge-free, training-free, and language-independent Multiword Expression Distance (MED). The new metric is derived from an accepted physical principle, measures the distance from an n-gram to its semantics, and outperforms other state-of-the-art methods on MWEs in two applications: question answering and named entity extraction.

1 Introduction

A Multiword Expression (MWE) is a sequence of neighboring words "whose exact and unambiguous meaning or connotation cannot be derived from the meaning or connotation of its components" (Choueka, 1988). In the paper, MWEs refer to non-compositional lexical units including idioms, terminologies and name entities. As Jackendoff (1997) notes, the magnitude of MWEs is far greater than what has traditionally been realized within linguistics. He estimates that the number of MWEs in a speaker's lexicon is of the same order of magnitude as the number of single words. In WordNet 1.7 (Fellbaum, 1998), 41 percent of the entries are multi-words. Some specialized domain vocabulary, such as terminology, overwhelmingly consists of MWEs. Automatic extraction of MWEs is indispensable to many tasks such as machine translation, name entity extraction, information retrieval and question answering.

Due to their non-compositionality, many MWEs cannot be directly identified using grammatical rules, which poses a major challenge to automatic analysis. Moreover, existing resources like dictionaries can never have adequate and timely coverage. Therefore people turn to statistical method to characterize MWEs.

Since Church and Hanks (1990) proposed Pointwise Mutual Information (PMI), a variety of measures, such as Log-likelihood, Symmetrical Conditional Probability (SCP) and Mutual Expectation (Dias et al., 2000), have been introduced to measure word association. Their basic ideas are very similar: the whole n-gram is separated into two parts and the association is determined by the joint probability and the probability of each part. Pecina (2006) compared 84 bi-gram association measures and found PMI has the best performance in Czech data. When applying these measures to the n-grams for $n > 2$, it is not clear how can the association between the deliberately separated two parts represent the non-compositionality of the whole n-gram. Different policies have been studied to extend these measures into arbitrary n-grams (Silva and Lopes, 1999; Schone and Jurafsky, 2001; Dias et al., 2000). Is there a fundamental, less arbitrary, and general approach to this problem? That is,

- Can we actually derive a MWE metric for n-grams from the first principles, instead of making a seemingly sensible, but really arbitrary, proposal?
- Will such a theoretically justified new metric actually works better than other heuristic

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 116–124,
Beijing, August 2010

measures for general MWEs?

This paper will answer above questions positively. We derive an optimal distance metric Multiword Expression Distance (MED). MED defines the semantic function for n-grams and the information distance (Bennett et al., 1998) from the n-grams to their semantics. Unlike previous methods it ensures the cohesion of the n-gram directly hence applicable to MWEs of any length.

The MED is naturally generalized to its conditional version. The extension is based on the observation that many MWEs are domain dependent. It is true that some MWEs are only used in certain domains, but they are domain free. For example, we know that “polymerase chain reaction” is some sort of terminology even if many of us do not know what it is exactly. However that is not always the case. For those who do not watch movies, the sentence “catch me if you can” will probably be taken as a non-MWE, instead of a movie name. The non-compositionality of this sentence appears only in the movies domain. The experimental results show that given appropriate phrases as conditions, the conditional MED performs better than MED.

We also investigate the efficacy of MED on post-processing of Question Answering (QA) and complex named entity extraction. The experimental results show that our method outperforms state of art methods (Zhang et al., 2009; Downey et al., 2007) in these two applications. Moreover, MED is a pure statistical metric which can be easily combined with other methods.

The remainder of this paper is organized as follows: In the next section we review the related work on Multiword Expression and information distance. Section 3 gives a preliminary introduction to Kolomogorov complexity and information distance. Section 4 proposes the formal definition of MED. In Section 5 we discuss the difference between MED and Pointwise Mutual Information. We apply MED to QA post-processing and complex named entity extraction in Section 6 and evaluate their performance in Section 7. In the last section we conclude this work.

2 Related Work

Researchers have explored various techniques for identifying MWEs. These approaches could be broadly classified into three types: linguistic methods, sequential tagging based methods and statistical methods.

The mostly used linguistic information for MWE extraction is words’ Part-Of-Speech tags. Justeson and Katz (1995) extracted technical terminologies from documents using a regular expression on POS-tags of a word sequence, together with some frequency constraints. Argamon et al. (1998) separated the POS sequence of a multi-word into small POS tiles, counted tile frequency in the MWE and non-MWE training sets and identify new MWEs by these counts. Although linguistic methods perform well in term extraction on specific domains, it cannot be generalized to identify arbitrary MWEs.

Several supervised learning methods have been used previously for extracting Name Entities including Hidden Markov Models, Maximum Entropy Markov Models and Conditional Random Field (CRF) models (McCallum and Li, 2003). In order to allow tractable computation, these models can only use local features in a small window. Although the approximate inference methods have been incorporated into sequential tagging model to capture non-local information (Finkel et al., 2005), these models are not capable of recognizing complex named entities, especially those containing conjunctions and prepositions. Experimental results in (Downey et al., 2007) show that statistical methods substantially outperform sequential tagging based methods on identifying complex named entities.

In statistical methods for MWE extraction, Church and Hanks (1990) first presented Pointwise Mutual Information (PMI) as an objective measure for estimating word association. Since then, many methods has been proposed to measure bi-gram association, such as Log-likelihood (Dunning, 1993) and Symmetrical Conditional Probability (Silva and Lopes, 1999). Pecina (2006) compared 84 bi-gram association measures and concluded that PMI had the best performance in Czech data. When it comes to measure the non-compositionality for arbitrary n-grams, policies were taken to separate n-gram into two parts X and Y so that it can be measured by existing bi-gram methods (such as PMI). Silva and Lopes (1999) and Dias et al. (2000) calculated the arithmetic average of every possible separation. Schone and Jurafsky (2001) define X and Y to be the word sequences $w_1 w_1 ... w_i$ and $w_{i+1} w_{i+2} ... w_n$, where i is chosen to maximize $P_x P_y$. Recently Zhang et al. (2009) proposed Enhanced Mutual Information (EMI) which measured the cohesion of n-gram by the frequency of itself and the frequency of each word.

The information distance is a universal distance measure between two information carrying entities (Bennett et al., 1998; Li et al., 2001; Li et al., 2004). The applications of information distance using compression were first introduced in (Li et al., 2001) and then in (Bennett et al., 2003; Chen et al., 2004). The experimental results in (Keogh et al., 2004) showed that information distance/compression based method was superior to 51 parameter-laden methods from seven major data mining conferences on their benchmark data. The web-based approximation of information distance was introduced by Cilibrasi and Vitányi (2007) to measure the semantic similarity of two words or concepts.

3 Preliminaries

3.1 Kolmogorov Complexity

Kolmogorov complexity defines randomness of an individual string. Fix a universal Turing machine U, the *Kolmogorov complexity* of a binary string x condition to another binary string y $K_U(x|y)$ is defined as the length of the shortest (prefix-free) program for U that outputs x with input y. It can be shown that for a different universal Turing machine U', for all x, y

$$K_U(x|y) = K_{U'}(x|y) + C, \qquad (1)$$

where the constant C depends only on U'. Thus, we can simply write $K_U(x|y)$ as $K(x|y)$ and $K(x|\epsilon)$ as $K(x)$, where ϵ is the empty string.

3.2 Information Distance

Between any two information carrying entities, is there an objective distance that is application-independent and unique, similar to the concept of distance in the physical world? From a commonly accepted physical principle of von Neumann and Landauer that irreversibly processing one bit of information costs 1KT of energy, Bennett et al. (1998) derived exactly such a distance: the Information Distance. Information Distance $E(x, y)$ between two objects x and y is the energy to convert between x and y. Bennett et al. (1998) proved:

Theorem 1 *Up to an additive logarithmic term,* $E(x, y) = \max\{K(x|y), K(y|x)\}$.

Thus, the max distance was defined below (Bennett et al., 1998):

$$D_{max}(x, y) = \max\{K(x|y), K(y|x)\}.$$

D_{max} was shown to satisfy distance requirements such as positivity, symmetricity and triangle inequality (Bennett et al., 1998). It was further shown that D_{max} is optimal in the sense that it is universal. That is, it minorizes (up to constant factors) all other nontrivial and computable distances. More precisely, a distance *D* is admissible if

$$\sum_y 2^{-D(x,y)} \leq 1. \qquad (2)$$

Thus, we exclude trivial distances such as $d(x, y) = 0$ for all x, y. It was proved in (Bennett et al., 1998) that for any admissible computable distance *D*, there is a constant *c*, for all *x*,*y*,

$$D_{max}(x, y) \leq D(x, y) + c.$$

In other words, if any such distance D discovers some similarity between x and y, so will D_{max}.

In order to deal with the information carrying objects of different sizes, the normalized information distance was proposed in (Li et al., 2001). In (Li et al., 2004), the normalized max distance was defined as:

$$d_{max}(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}}$$

d_{max} satisfies positivity, symmetricity, triangle inequality and some weak form of universality (Li et al., 2004).

4 A New Metric for MWE

4.1 The Semantics

When applying the Information Distance to identifying MWEs, how to encode n-grams and their semantics is the first thing to be considered. It is inappropriate to encode MWEs literally. For example, when referring to “kick the bucket”, the three words “kick”, “the” and “bucket” cannot represent all the semantics about this expression.

Inspired by Cilibrasi and Vitányi (2007), we define *context* of an n-gram as the set of all the web pages containing it. Also, *semantic* of an n-gram is defined as the set of all the web pages containing all the words appeared in that n-gram. For example, the semantic of “U.S. president” including not only the pages containing itself but also those containing “the president of U.S.” or “president Obama says that ... U.S. government...”.

4.2 Multiword Expression Distance

Let us denote the vocabulary set by S and the set of web pages by Ω. The cardinality of Ω is denoted by M=$|\Omega|$. Define $G \equiv S^+$ as the set of n-grams. A search term t is defined as an n-gram or the conjunction of search terms. Denote T as the set of search terms and we have $G \subset T$. Let $\phi : T \to 2^\Omega$ be the *context* function mapping each search term t to the web set which includes (and only includes) all the web pages containing all the n-grams in t. Let $\theta : G \to T$ be the function mapping each n-gram $g = w_1 w_2 ... w_n$ to $\bigwedge_i w_i$, the conjunction of the words in it. Finally we define the *semantic* function $\mu : G \to 2^\Omega$ as the composite function $\phi \circ \theta$. It is obvious that for any n-gram g, we have $\phi(g) \subseteq \mu(g)$. Given an n-gram g, we will encode $\phi(g)$ and $\mu(g)$ and calculate the distance between them.

While $K(x)$ is not computable, a simple heuristic, noticed by Cilibrasi and Vitányi (2007), is to use Shannon-Fano code to encode the probability (approximated by its internet frequency) of x. Assume that all web pages are equiprobable, with the probability of being returned by search engine being $\frac{1}{M}$. Let $p : \phi(T) \to [0, 1]$ be the *context* probability function in which $\phi(T) \equiv \{x | \exists y \in T, x = \phi(y)\}$. Since each context is a set of webpages, the probability of context c is defined as $p(c) = \frac{|c|}{N}$ where $N = \sum_{c \in \phi(T)} |c|$ ensures p is a valid probability function. The Shannon-Fano code (Li and Vitányi, 2008) length associated with p can then be regarded as an approximation of K,

$$K(x) \approx -\log p(x) \qquad (3)$$

$$K(x, y) \approx -\log p(x, y) \qquad (4)$$

According to (3),(4) and Theorem 1, D_{max} can be approximated as follows:

$$\begin{aligned} & D_{max}(x, y) \\ = \; & \max\{K(x|y), K(y|x)\} \\ = \; & K(x, y) - \min\{K(y), K(x)\} \\ \approx \; & \max\{\log |x|, \log |y|\} - \log |x \cap y| \end{aligned}$$

Similarly, we have

$$\begin{aligned} & D_{max}(x, y|c) \\ \approx \; & \max\{\log |x \bigcap c|, \log |y \bigcap c|\} - \log |x \bigcap y \bigcap c| \end{aligned}$$

Since $\phi(g) \subseteq \mu(g)$, the Multiword Expression Distance of an n-gram g can be defined as follows:

$$\begin{aligned} & \text{MED}(g) \\ \equiv \; & D_{max}(\phi(g), \mu(g)) \\ \approx \; & \max\{\log \frac{|\phi(g)|}{|\phi(g) \bigcap \mu(g)|}, \log \frac{|\mu(g)|}{|\phi(g) \bigcap \mu(g)|}\} \\ = \; & \log |\mu(g)| - \log |\phi(g)| \end{aligned}$$

Given a search term c as condition, the Conditional Multiword Expression Distance of an n-gram g is defined as follows:

$$\begin{aligned} \text{MED}(g|c) \; & \equiv \; D_{max}(\phi(g), \mu(g)|\phi(c)) \\ & \approx \; \log |\mu(g) \bigcap \phi(c)| - \log |\phi(g) \bigcap \phi(c)| \end{aligned}$$

Based normalized information distance, NMED and its conditional version can be derived as follows:

$$\text{NMED}(g) \approx \frac{\log |\mu(g)| - \log |\phi(g)|}{\log N - \log |\phi(g)|}$$

$$\text{NMED}(g|c) \approx \frac{\log |\mu(g) \bigcap \phi(c)| - \log |\phi(g) \bigcap \phi(c)|}{\log |\phi(c)| - \log |\phi(g) \bigcap \phi(c)|}$$

Where N can be estimated from the size of internet by some combinatorial methods.

To implement MED by a general search engine, we assume Ω to be the set of indexed webpages. Thus, $|\phi(g)|$ and $|\mu(g)|$ can be approximated by the hit numbers given g and the “logic and” of each word in g as queries. Yahoo Search is used in our experiments.

5 Relation with Pointwise Mutual Information

When $n = 2$, we denote $\mathrm{P}(w_1 w_2)$ the probability of a web page containing bi-gram $g = w_1 w_2$ and $\mathrm{P}(w_1 \bigwedge w_2)$ the probability of a web page containing w_1 and w_2. Assuming the occurrence of w_1 and w_2 are independent, we have

$$\begin{aligned} \mathrm{MED}_2(g) &= \log \frac{|\phi(w_1 \bigwedge w_2)|}{|\phi(w_1 w_2)|} \\ &= \log \frac{\mathrm{P}(w_1 \bigwedge w_2)}{\mathrm{P}(w_1 w_2)} \\ &\approx \log \frac{\mathrm{P}(w_1) P(w_2)}{\mathrm{P}(w_1 w_2)} \\ &\propto -\mathrm{PMI}(g) \end{aligned}$$

Thus, PMI is inversely proportional to MED under the independence assumption. This assumption is unadvisable for obvious reasons. PMI compares the probability of observing x and y within a given window w (w=2 when measuring collocation) with the probabilities of observing x and y independently. However, most of the word sequences in practice (both MWEs and non-MWEs) are far from being independent. Therefore the assumption potentially creates additional noises to MED, especially when $n > 2$. The internet contains billions of pages and thus we can count the pages containing specified words directly without making independent assumption to overcome data sparseness.

6 Applications

6.1 MWE for QA Systems

Some types of questions require a QA system to return phrases as the answers instead of sentences, such as Factoid and List. Given a question, we need to generate queries, obtain relevant pages from the internet, extract the candidate n-grams from relevant pages and finally rank all the candidates by their likelihood of being an answer.

Some previous work exploited web redundancy to estimate answer validity(Magnini et al., 2002; Zhang et al., 2008). No research, to our knowledge, has focused on checking the completeness of candidates. Most of texts on the internet are informal (e.g. they contain uncapitalized proper nouns and incomplete sentence structures). Parser and named entity recognizers trained on formal corpus are unpractical on recognize NP chunks or name entities on the web.

Observing that each candidate is n-gram and checking the completeness of a candidate is to measure its non-compositionality, we introduce a simple MWEs-based method to rank all candidates by their completeness and merge similar answers.

Given a question and a list of candidate answers:

1. Extract proper nouns from the question as conditions.
2. Calculate the conditional MED (or MED if no proper noun is found in question) for each candidate. Then for each pair of literally similar candidates, the one with larger MED distance is removed.
3. Rank the rest candidates by conditional MED.

This method is case insensitive and do not rely on context information. All of the statistics are performed on the internet thus no local corpus is needed.

6.2 Complex Named Entity Extraction

In many previous work (McCallum and Li, 2003; Finkel et al., 2005), named entity extraction is combined with classification, which is known as Name Entity Recognition (NER). Most of these NER technique are based on sequential tagging models and unsuitable to the task of locating complex named entities in Web text. In (Downey et al., 2007), the author treated named entity as a type of MWE and proposed the algorithm LEX++ to locate complex named entities.

Inspired by Downey's work, we propose a conditional MED based algorithm MWE++ to extract named entities. Given a sentence $S = \{S_1, S_2 ... S_n\}$ and parameters τ_1,τ_2 and δ, MWE++ proceeds as follows:

1. Initialize a sequence of names $N = (n_1, n_2, ..., n_M)$ equal to the maximal contiguous substrings of S that consist entirely of capitalized words. If the first word of S appears capitalized in the local corpus and it is at the beginning of a sentence more than δ of the times, it is omitted from N.

2. Until N does not change during last iteration:

   (a) Choose the *mergeable* pair of names (n_i, n_{i+1}) with minimum conditional MED.

   (b) Replace n_{min_i} and n_{min_i+1} with the single name $n_{min_i} w_{min_i} n_{min_i+1}$ where w_i is the uncapitalized words between n_i and n_{i+1}.

3. For every names n_i in N

   (a) Check common prefix and punctuation at boundary of n_i via local corpus.

   (b) Check number at boundary of n_i via internet.

In MWE++, We define two thresholds τ_1 and τ_2 to estimate the name entity confidence of a given n-gram. If $\mathrm{MED}(g|.)$ is lower than τ_1, between τ_1 and τ_2 or higher than τ_2, $\mathrm{conf}(g)$ will be 2 (Definitely), 1 (Probably) or 0 (Impossible). The confidence of all initialized capitalized words will be set to 1. If an n-gram contain unmatched brackets or quotation marks, its confidence will be set to 0. Also, The confidence of n-gram containing comma will be reduced by 1. We say a pair of names (n_i, n_{i+1}) is mergeable if and only if $\mathrm{conf}(n_i w_i n_{i+1}) \geq \max(\mathrm{conf}(n_i), \mathrm{conf}(n_{i+1}))$.

After iteration, we will check common prefixes, punctuations and numbers at boundary of each names. If a name n_i is immediately preceded by a single number t and $\mathrm{conf}(tn_i) \geq 1$, we replace n_i by tn_i. Similarly, a number t immediately following n_i is appended to n_i when $\mathrm{conf}(n_i t) \geq 1$. Due to the limitation of search engine, punctuation check and common prefix check modules are performed on local corpus just the same as LEX++.

7 Experiments and Analysis

7.1 Compositionality Measure

In this section, we evaluate how well can MED separate non-compositional phrases (idioms) from compositional ones. First we evaluate MED and other four metrics on English_VPC data published on the MWE 2008 shared task. The data set contains 3078 verb-noun bi-grams and 14 percent of them are annotated as idiomatic. The average precision of MED, PMI, SCP, t-score and EMI (Zhang et al., 2009) are 0.234, 0.233, 0.285, 0.274 and 0.205. The result shows that MED is not distinguished on bi-grams test. It is partly because most idiomatic verb-noun collocations are often used non-idiomatically. Their compositionality are not necessarily lower than non-idiomatic ones.

We also evaluate different metrics on n-grams of varied lengths. Since all published MWE data sets we find only contain bi-grams, we construct our test set as follows. We first collected common idioms from the lists of english idioms on Wikipedia. To get enough common but not idiomatic phrases, we collect common compositional phrases from UsingEnglish.com, englishspeak.com, Wikipedia and China Daily BBS. Since it is difficult for non-native speakers to pick up idioms from non-idiomatic ones, we do not manually check all compositional phrases. The test set contains 1529 idioms and 1798 compositional phrases. The n-gram frequencies are not significantly different between idioms and compositional phrases. The mean and standard deviation are 2.1×10^5 and 7.8×10^5 on idioms and 7.4×10^5 and 4.8×10^6 on compositional phrases. We employ different measures to rank all the phrases. Non-conditional MED and NMED are compared with AVG_SCP (Silva and Lopes, 1999), MAX_PMI (Schone and Jurafsky, 2001), EMI (Zhang et al., 2009) and the baseline n-gram frequency. T-score is not under evaluation because we do not find sound n-gram extension for it. The precision-recall curve is shown in Fig. 1. Since the performance of MED and NMED are very

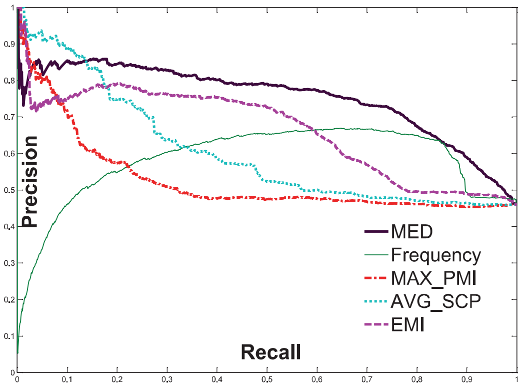


Figure 1: Precision-recall curves of five measures

	freq	MAX_PMI	AVG_SCP	EMI	NMED	MED	MED(.|.)
fairy tale	0.493	0.484	0.570	0.515	0.615	0.617	**0.657**
science fiction	0.500	0.470	0.558	0.525	0.596	0.599	**0.633**
action movie	0.695	0.523	0.723	0.703	0.763	0.768	**0.823**
animation	0.561	0.642	**0.693**	0.489	0.671	0.673	0.689
horror movie	0.595	0.528	0.647	0.633	0.667	0.670	**0.692**
documentary	0.525	0.549	0.626	0.512	0.596	0.598	**0.654**
hip hop	0.598	0.627	0.645	0.635	0.652	0.651	**0.712**
jazz	0.549	0.501	0.543	0.539	0.627	0.625	**0.716**
rock&roll	0.742	0.567	0.730	0.741	0.708	0.717	**0.836**
company	0.614	0.584	0.689	0.663	0.754	**0.756**	0.735
soccer player	0.945	0.648	0.904	**0.973**	0.911	0.918	0.941
novelists	0.772	0.701	**0.870**	0.866	0.821	0.828	0.864
PS3 game	0.603	0.675	0.740	0.535	0.742	**0.744**	0.727
overall	0.612	0.577	0.688	0.629	0.696	0.700	**0.726**

Table 1: Performance of different measures in each list

close, NMED is not displayed for clarity. From the result we can see that MED performs substantially better than all the other measures. Average precision(avp) of the top 3 measures MED, EMI and AVG_SCP are 0.75, 0.71 and 0.66.

7.2 QA Post-processing

It is difficult to evaluate the method introduced in Section 6.1 directly since QA benchmarks mainly focus on accuracy of the top one answer instead of the completeness of top-n candidates. Therefore, the experiment is designed as follows. We extract name lists on different domains from Wikipedia. For each name in each list, we put it into a search engine and get the context from a random selected snippet. For each name, We created two incomplete names by randomly adding (or removing) one or two words according to its context. It is guaranteed that the original name and its counterpart with noise must have at least two words in common. We tag the original names and the noise added ones in each list as positive and negative samples. A list can be regarded as the candidates and the list name (or its synonym) can be seen as the key phrase extracted from question.

The test set can be divided into six common categories: movie, book, music, person, organization and video game. Each category contains one to four lists. The test set contains 11080 samples in total. Still, we employ the measures in previous experiments to rank all the candidates to see if the complete names can be separated from the incomplete names. The results are listed in Table 1. The overall avp is the average of the avp of each lists weighted by their size.

It is shown that the performance of conditional MED is the best over all metrics, followed by MED. The reason why EMI and AVG_SCP get best results on soccer player and novelists is that they take more advantage of frequency. Since the length of people's name are short (2 to 3 words), most of negative samples are created by adding words, which makes frequency important.

7.3 Complex Named Entity Extraction

In this section we evaluate the named entity extraction performance of Algorithm MWE++. The experiment is done on the corpus, the training set and the test set provided by Downey et al. (2007). Four classes of entities (Actor, Book, Company and Film) were manually annotated on both training and test set. All sentences in the corpus contain named entities from the above four classes (but not annotated). The corpus consists of 183,726 sentences while the training and the test set contain 200 and 629 sentences, respectively. Furthermore, test sentences are separated into 100 difficult cases and 529 easy cases. All difficult cases contain complex name entities (entities containing uncapitalized words), such as "Procter and

Gamble" and "Gone with the Wind".

The conditional MED metric in this experiment is redefined as follows:

$$\mathrm{MED}(g|C) = \min_{c \in C}\{\mathrm{MED}(g|c)\},$$

where C={"IMDB","Amazon","corporation"}. "IMDB" is used as the condition of Actor and Film while "Amazon" and "corporation" are chosen to be the condition of Book and Company. We compute the conditional MED for all entities on training set. τ_1 is set to the median and τ_2 is set to the value larger than 90% entities on training set. δ is set to 0.5. MWE++ is performed on the 100 difficult cases. The results shown in Table 2 convincingly show that MWE++ significantly outperforms LEX++, supervised models (SVMCMM, CRF) and rule-based model (MAN) on identifying complex named entities. Compared to LEX++, MWE++ is not only more accurate but also more flexible. LEX++ relies on local corpus while MWE++ does not. When recognizing new entities, we just need to find appropriate condition words instead of preparing new corpus. For the sake of completeness, the F-score of MWE++ on easy cases is 91, which is lower than all the other methods. However this is irrelevant since this part can be made quite accurate by specialized databases and training by any known methods.

All test data in this paper can be downloaded from http://60.195.250.61:8080/download/.

8 Conclusion

We have derived an MWE metric MED from the first principles via Information Distance. The new metric measures the distance from an n-gram to its semantics. It is provably optimal (universal), overcomes several deficiencies of previous approaches, and convincingly outperforms the other methods.

	F_1	Recall	Precision
MAN	0.18	0.22	0.16
CRF	0.35	0.42	0.31
SVMCMM	0.42	0.48	0.37
LEX++	0.74	0.76	0.72
MWE++	**0.83**	**0.86**	**0.80**

Table 2: Named entity extraction on difficult cases

Also, we have taken advantage of the fact that some MWEs are domain dependent. This feature is important when recognizing named entities and terminologies. The conditional MED is better than MED when we know what we are looking for. Since MED is quite different from previous measures, it can be combined with others by machine learning approaches and enhance the overall performance. Further experiments are needed.

Acknowledgment

This work was supported mainly by Canada's IDRC Research Chair in Information Technology program, Project Number: 104519-006. It is also supported by the Chinese Natural Science Foundation grant No. 60973104, NSERC Grant OGP0046506, 863 Grant 2008AA02Z313 from China's Ministry of Science and Technology, Canada Research Chair program, MITACS, an NSERC Collaborative Grant, and Ontario's Premier's Discovery Award.

References

Shlomo Argamon, Ido Dagan and Yuval Krymolowski 1998. A memory-based approach to learning shallow natural language patterns. In *Proc. of COLING*, 1998, pp. 67-73.

Charles H. Bennett, Peter Gács, Ming Li, Paul M.B. Vitányi, and Wojciech H. Zurek 1998. Information distance. *IEEE Trans-IT* 44:4, 1998, pp. 1407-1423.

Charles H. Bennett, Ming Li and Bin Ma 2003. Chain letters and evolutionary histories. *Scientific American*, 288:6, (feature article), 76-81.

Xin Chen, Brent Francia, Ming Li, Brian Mckinnon, Amit Seker 2004. Shared information and program plagiarism detection. *IEEE Trans. Information Theory*, 50:7, 1545-1550.

Yaacov Choueka 1988. Looking for needles in a haystack or locating interesting collocation expressions in large textual databases In *Proc. of the RIAO*, 1988, pp. 38-43.

Kenneth W. Church and Patrick Hanks 1990. Word association norms, mutual information and lexicography. *Computational Linguistics*, Vol. 16, No. 1, pp. 22-29.

Rudi L. Cilibrasi and Paul M.B. Vitányi 2007. The Google similarity distance. *IEEE Trans-Knowledge and Data Engineering* 19:3, 2007, pp. 370-383.

Joaquim F. da Silva and Gabriel P. Lopes 1999. A local maxima method and a fair dispersion normalization for extracting multi-word units from corpora. In *Proc. of Sixth Meeting on Mathematics of Language*, pp. 369-381.

Gaël Dias, Sylvie Guilloré and José G.P. Lopes 2000. Mining textual associations in text corpora. In *Sixth ACM SIGKDD, Workshop on Text Mining*, pp. 92-95.

Ted Dunning 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*. Vol. 19, No. 1, 61-74.

Doug Downey, Matthew Broadhead and Oren Etzioni 2007. Locating complex named entities in web text. In *Proc. of IJCAI*, 2007, pp. 2733-2739.

Christine Fellbaum 1998. WordNet: an electronic lexical database. MIT Press, Cambrige, MA.

Jenny R. Finkel, Trond Grenager and Christopher Manning 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc. of ACL*, 2005, pp. 363-370.

Ray Jackendoff 1997. The architecture of the language faculty. MIT Press, Cambrige, MA.

John S. Justeson and Slava M. Katz 1995. Technical terminology: Some linguistic properties and an algorithm for indentification in text. *Natural Language Engineering* 1:1, 1995, 9-27.

Eamonn J. Keogh, Stefano Lonardi and Chotirat A. Ratanamahatana 2004. Towards parameter-free data mining. In *Proc. of ACM SIGKDD*,2004, pp. 206-215.

Ming Li, Jonathan H. Badger, Xin Chen, Sam Kwong, Paul Kearney, and Haoyong Zhang 2001. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17:2(2001), 149-154.

Ming Li, Xin Chen, Xin Li, Bin Ma and Paul M.B. Vitányi 2004. The similarity metric. *IEEE Trans-IT* 50:12, 2004, 3250-3264.

Ming Li and Paul M.B. Vitányi 2008. An introduction to kolmogorov complexity and its applications. Springer-Verlag, New York, 2008. Third edition.

Andrew McCallum and Wei Li 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proc. of the 7th conference on Natural language learning at HLT-NAACL*,2003 ,pp. 188-191.

Bernardo Magnini, Matteo Negri and Hristo Tanev 2002. Is it the right answer? Exploiting web redundancy for answer validation. In *Proc. of ACL*,2002, pp. 425-432.

Pavel Pecina 2006. An extensive empirical study of collocation extraction methods. In *Proc. of COLING-ACL*, 2006, pp. 953-960.

Patrick Schone and Daniel Jurafsky 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In *Proc. of EMNLP, 2001*.

Wen Zhang, Taketoshi Yoshida, Xijin Tang and Tu-Bao Ho 2009. Improving effectiveness of mutual information for substantival multiword expression extraction. *Expert Systems with Applications*, Volume 36, 10919-10930, Elsevier.

Xian Zhang, Yu Hao, Xiaoyan Zhu and Ming Li 2007. New information measure and its application in question answering system. *J. Comput. Sci. Tech.*, 23:4(2008), pp. 557-572. (Preliminary version appeared in SIGKDD 2007.)

A Utility-Driven Approach to Question Ranking in Social QA

Razvan Bunescu
School of EECS
Ohio University
bunescu@ohio.edu

Yunfeng Huang
School of EECS
Ohio University
yh324906@ohio.edu

Abstract

We generalize the task of finding question paraphrases in a question repository to a novel formulation in which known questions are ranked based on their utility to a new, reference question. We manually annotate a dataset of 60 groups of questions with a partial order relation reflecting the relative utility of questions inside each group, and use it to evaluate meaning and structure aware utility functions. Experimental evaluation demonstrates the importance of using structural information in estimating the relative usefulness of questions, holding the promise of increased usability for social QA sites.

1 Introduction

Open domain Question Answering (QA) is one of the most complex and challenging tasks in natural language processing. While building on ideas from Information Retrieval (IR), question answering is generally seen as a more difficult task due to constraints on both the input representation (natural language questions vs. keyword-based queries) and the form of the output (focused answers vs. entire documents). Recently, community-driven QA sites such as Yahoo! Answers and WikiAnswers have established a new approach to question answering in which the burden of dealing with the inherent complexity of open domain QA is shifted from the computer system to volunteer contributors. The computer is no longer required to perform a deep linguistic analysis of questions and generate corresponding answers, and instead acts as a mediator between users submitting questions and volunteers providing the answers. In most implementations of community-driven QA, the mediator system has a well defined strategy for enticing volunteers to post high quality answers on the website. In general, the overall objective is to minimize the response time and maximize the accuracy of the answers, measures that are highly correlated with user satisfaction. For any submitted question, one useful strategy is to search the QA repository for similar questions that have already been answered, and provide the corresponding ranked list of answers, if such a question is found. The success of this approach depends on the definition and implementation of the question-to-question similarity function. In the simplest solution, the system searches for previously answered questions based on exact string matching with the reference question. Alternatively, sites such as WikiAnswers allow the users to mark questions they think are rephrasings ("alternate wordings", or paraphrases) of existing questions. These question clusters are then taken into account when performing exact string matching, therefore increasing the likelihood of finding previously answered questions that are semantically equivalent to the reference question. Like the original question answering task, the solution to question rephrasing is also based on volunteer contributions. In order to lessen the amount of work required from the contributors, an alternative solution is to build a system that automatically finds rephrasings of questions, especially since question rephrasing seems to be computationally less demanding than question answering. The question rephrasing subtask has spawned a diverse set of approaches. (Herm-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 125–133,
Beijing, August 2010

jakob et al., 2002) derive a set of phrasal patterns for question reformulation by generalizing surface patterns acquired automatically from a large corpus of web documents. The focus of the work in (Tomuro, 2003) is on deriving reformulation patterns for the interrogative part of a question. In (Jeon et al., 2005), word translation probabilities are trained on pairs of semantically similar questions that are automatically extracted from an FAQ archive, and then used in a language model that retrieves question reformulations. (Jijkoun and de Rijke, 2005) describe an FAQ question retrieval system in which weighted combinations of similarity functions corresponding to questions, existing answers, FAQ titles and pages are computed using a vector space model. (Zhao et al., 2007) exploit the Encarta logs to automatically extract clusters containing question paraphrases and further train a perceptron to recognize question paraphrases inside each cluster based on a combination of lexical, syntactic and semantic similarity features. More recently, (Bernhard and Gurevych, 2008) evaluated various string similarity measures and vector space based similarity measures on the task of retrieving question paraphrases from the WikiAnswers repository.

According to previous work in this domain, a question is considered a rephrasing of a reference question Q_0 if it uses an alternate wording to express an identical information need. For example, Q_0 and Q_1 below may be considered rephrasings of each other, and consequently they are expected to have the same answer.

Q_0 What should I feed my turtle?

Q_1 What do I feed my pet turtle?

Community-driven QA sites are bound to face situations in which paraphrasings of a new question cannot be found in the QA repository. We believe that computing a ranked list of existing questions that partially address the original information need could be useful to the user, at least until other users volunteer to give an exact answer to the original, unanswered reference question. For example, in the absence of any additional information about the reference question Q_0, the expected answers to questions Q_2 and Q_3 above may be seen as partially overlapping in information content with the expected answer for the reference question. An answer to question Q_4, on the other hand, is less likely to benefit the user, even though it has a significant lexical overlap with the reference question.

Q_2 What kind of fish should I feed my turtle?

Q_3 What do you feed a turtle that is the size of a quarter?

Q_4 What kind of food should I feed a turtle dove?

In this paper, we propose a generalization of the question paraphrasing problem to a question ranking problem, in which questions are ranked in a partial order based on the relative information overlap between their expected answers and the expected answer of the reference question. The expectation in this approach is that the user who submits a reference question will find the answers of the highly ranked question to be more useful than the answers associated with the lower ranked questions. For the reference question Q_0 above, the system is expected to produce a partial order in which Q_1 is ranked higher than Q_2, Q_3 and Q_4, whereas Q_2 and Q_3 are ranked higher than Q_4. In Section 2 we give further details on the question ranking task and describe a dataset of questions that have been manually annotated with partial order information. Section 3 presents a set of initial approaches to question ranking, followed by their experimental evaluation in Section 4. The paper ends with a discussion of future work, and conclusion.

2 A Partially Ordered Dataset for Question Ranking

In order to enable the evaluation of question ranking approaches, we created a dataset of 60 groups of questions. Each group consists of a reference question (e.g. Q_0 above) that is associated with a partially ordered set of questions (e.g. Q_1 to Q_4 above). The 60 reference questions have been selected to represent a diverse set of question categories from Yahoo! Answers. For each reference questions, its corresponding partially ordered set is created from questions in Yahoo! Answers

REFERENCE QUESTION (Q_r)
Q_5 What's a good summer camp to go to in FL?
PARAPHRASING QUESTIONS ($\mathcal{P}$)
Q_6 What camps are good for a vacation during the summer in FL?
Q_7 What summer camps in FL do you recommend?
USEFUL QUESTIONS ($\mathcal{U}$)
Q_8 Does anyone know a good art summer camp to go to in FL?
Q_9 Are there any good artsy camps for girls in FL?
Q_{10} What are some summer camps for like singing in Florida?
Q_{11} What is a good cooking summer camp in FL?
Q_{12} Do you know of any summer camps in Tampa, FL?
Q_{13} What is a good summer camp in Sarasota FL for a 12 year old?
Q_{14} Can you please help me find a surfing summer camp for beginners in Treasure Coast, FL?
Q_{15} Are there any acting summer camps and/or workshops in the Orlando, FL area?
Q_{16} Does anyone know any volleyball camps in Miramar, FL?
Q_{17} Does anyone know about any cool science camps in Miami?
Q_{18} What's a good summer camp you've ever been to?
NEUTRAL QUESTIONS ($\mathcal{N}$)
Q_{19} What's a good summer camp in Canada?
Q_{20} What's the summer like in Florida?

Table 1: A question group.

and other online repositories that have a high cosine similarity with the reference question. Due to the significant lexical overlap between the questions, this is a rather difficult dataset, especially for ranking methods that rely exclusively on bag-of-words measures. Inside each group, the questions are manually annotated with a partial order relation, according to their utility with respect to the reference question. We shall use the notation $\langle Q_i \succ Q_j | Q_r \rangle$ to encode the fact that question Q_i is *more useful than* question Q_j with respect to the reference question Q_r. Similarly, $\langle Q_i = Q_j \rangle$ will be used to express the fact that questions Q_i and Q_j are reformulations of each other (the reformulation relation is independent of the reference question). The partial ordering among the questions Q_0 to Q_4 above can therefore be expressed concisely as follows: $\langle Q_0 = Q_1 \rangle$, $\langle Q_1 \succ Q_2 | Q_0 \rangle$, $\langle Q_1 \succ Q_3 | Q_0 \rangle$, $\langle Q_2 \succ Q_4 | Q_0 \rangle$, $\langle Q_3 \succ Q_4 | Q_0 \rangle$. Note that we do not explicitly annotate the relation $\langle Q_1 \succ Q_4 | Q_0 \rangle$, since it can be inferred based on the transitivity of the *more useful than* relation: $\langle Q_1 \succ Q_2 | Q_0 \rangle \wedge \langle Q_2 \succ Q_4 | Q_0 \rangle \Rightarrow \langle Q_1 \succ Q_4 | Q_0 \rangle$. Also note that no relation is specified between Q_2 and Q_3, and similarly no relation can be inferred between these two questions. This reflects our belief that, in the absence of any additional information regarding the user or the "turtle" referenced in Q_0, we cannot compare questions Q_2 and Q_3 in terms of their usefulness with respect to Q_0.

Table 1 shows another reference question Q_5 from our dataset, together with its annotated group of questions Q_6 to Q_{20}. In order to make the annotation process easier and reproducible, we divide it into two levels of annotation. During the first annotation stage (L_1), each question group is partitioned manually into 3 subgroups of questions:

- $\mathcal{P}$ is the set of *paraphrasing* questions.
- $\mathcal{U}$ is the set of *useful* questions.
- $\mathcal{N}$ is the set of *neutral* questions.

A question is deemed useful if its expected answer may overlap in information content with the expected answer of the reference question. The expected answer of a neutral question, on the other hand, should be irrelevant with respect to the reference question. Let Q_r be the reference question, $Q_p \in \mathcal{P}$ a paraphrasing question, $Q_u \in \mathcal{U}$ a useful question, and $Q_n \in \mathcal{N}$ a neutral question. Then the following relations are assumed to hold among these questions:

1. $\langle Q_p \succ Q_u | Q_r \rangle$: a *paraphrasing* question is more useful than a *useful* question.

2. $\langle Q_u \succ Q_n | Q_r \rangle$: a *useful* question is more useful than a *neutral* question.

We also assume that, by transitivity, the following ternary relations also hold: $\langle Q_p \succ Q_n | Q_r \rangle$, i.e. a *paraphrasing* question is more useful than a *neutral* question. Furthermore, if $Q_{p_1}, Q_{p_2} \in \mathcal{P}$ are two paraphrasing questions, this implies $\langle Q_{p_1} = Q_{p_2} | Q_r \rangle$.

For the vast majority of questions, the first annotation stage is straightforward and non-controversial. In the second annotation stage (L_2), we perform a finer annotation of relations between questions in the middle group $\mathcal{U}$. Table 1 shows two such relations (using indentation): $\langle Q_8 \succ Q_9 | Q_5 \rangle$ and $\langle Q_8 \succ Q_{10} | Q_5 \rangle$. Question Q_8 would have been a rephrasing of the reference question, were it not for the noun "art" modifying the focus noun phrase "summer camp". Therefore, the information content of the answer to Q_8 is strictly subsumed in the information content associated with the answer to Q_5. Similarly, in Q_9 the focus noun phrase is further specialized through the prepositional phrase "for girls". Therefore, (an answer to) Q_9 is less *useful* to Q_5 than (an answer to) Q_8, i.e. $\langle Q_8 \succ Q_9 | Q_5 \rangle$. Furthermore, the focus "art summer camp" in Q_8 conceptually subsumes the focus "summer camps for singing" in Q_{10}, therefore $\langle Q_8 \succ Q_{10} | Q_5 \rangle$.

Table 2 below presents the following statistics on the annotated dataset: the number of reference questions (Q_r), the total number of paraphrasings ($\mathcal{P}$), the total number of useful questions ($\mathcal{U}$), the total number of neutral questions ($\mathcal{N}$), and the total number of *more useful than* ordered pairs encoded in the dataset, either explicitly or through transitivity, in the two annotation levels L_1 and L_2.

Q_r	$\mathcal{P}$	$\mathcal{U}$	$\mathcal{N}$	L_1	L_2
60	177	847	427	7,378	7,639

Table 2: Dataset statistics.

3 Question Ranking Methods

An ideal question ranking method would take an arbitrary triplet of questions Q_r, Q_i and Q_j as input, and output an ordering between Q_i and Q_j with respect to the reference question Q_r, i.e. one of $\langle Q_i \succ Q_j | Q_r \rangle$, $\langle Q_i = Q_j | Q_r \rangle$, or $\langle Q_j \succ Q_i | Q_r \rangle$. One approach is to design a *usefulness* function $u(Q_i, Q_r)$ that measures how useful question Q_i is for the reference question Q_r, and define the *more useful than* ($\succ$) relation as follows:

$$\langle Q_i \succ Q_j | Q_r \rangle \Leftrightarrow u(Q_i, Q_r) > u(Q_j, Q_r)$$

If we define $I(Q)$ to be the information need associated with question Q, then $u(Q_i, Q_r)$ could be defined as a measure of the relative overlap between $I(Q_i)$ and $I(Q_r)$. Unfortunately, the information need is a concept that, in general, is defined only intensionally and therefore it is difficult to measure. For lack of an operational definition of the information need, we will approximate $u(Q_i, Q_r)$ directly as a measure of the similarity between Q_i and Q_r. The similarity between two questions can be seen as a special case of text-to-text similarity, consequently one possibility is to use a general text-to-text similarity function such as *cosine similarity* in the vector space model (Baeza-Yates and Ribeiro-Neto, 1999):

$$cos(Q_i, Q_r) = \frac{Q_i^T Q_r}{\|Q_i\| \|Q_r\|}$$

Here, Q_i and Q_r denote the corresponding *tf*×*idf* vectors. As a measure of question-to-question similarity, cosine has two major drawbacks:

1. As an exclusively lexical measure, it is oblivious to the meanings of words in each question.

2. Questions are treated as bags-of-words, and thus important structural information is missed.

3.1 Meaning Aware Measures

The three questions below illustrate the first problem associated with cosine similarity. Q_{22} and Q_{23} have the same cosine similarity with Q_{21}, they are therefore indistinguishable in terms of their usefulness to the reference question Q_{21}, even though we expect Q_{22} to be more useful than Q_{23} (a place that sells hydrangea often sells other types of plants too, possibly including cacti).

Q_{21} Where can I buy a hydrangea?

Q_{22} Where can I buy a cactus?

Q_{23} Where can I buy an iPad?

To alleviate the lexical chasm, we can redefine $u(Q_i, Q_r)$ to be the similarity measure proposed by (Mihalcea et al., 2006) as follows:

$$mcs(Q_i, Q_r) = \frac{\sum_{w \in \{Q_i\}} (maxSim(w, Q_r) * idf(w))}{\sum_{w \in \{Q_i\}} idf(w)} + \frac{\sum_{w \in \{Q_r\}} (maxSim(w, Q_i) * idf(w))}{\sum_{w \in \{Q_r\}} idf(w)}$$

Since scaling factors are immaterial for ranking, we have ignored the normalization constant contained in the original measure. For each word $w \in Q_i$, $maxSim(w, Q_r)$ computes the maximum semantic similarity between w and any word $w_r \in Q_r$. The similarity scores are then weighted by the corresponding *idf*'s, and normalized. A similar score is computed for each word $w \in Q_r$. The score computed by $maxSim$ depends on the actual function used to compute the word-to-word semantic similarity. In this paper, we evaluated four of the knowledge-based measures explored in (Mihalcea et al., 2006): *wup* (Wu and Palmer, 1994), *res* (Resnik, 1995), *lin* (Lin, 1998), and *jcn* (Jiang and Conrath, 1997). Since all these measures are defined on pairs of WordNet concepts, their analogues on word pairs (w_i, w_r) are computed by selecting pairs of WordNet synsets (c_i, c_r) such that w_i belongs to concept c_i, w_r belongs to concept c_r, and (c_i, c_r) maximizes the similarity function. The measure introduced in (Wu and Palmer, 1994) finds the *least common subsumer (LCS)* of the two input concepts in the WordNet hierarchy, and computes the ratio between its depth and the sum of the depths of the two concepts:

$$wup(c_i, c_r) = \frac{2 * depth(lcs(c_i, c_r))}{depth(c_i) + depth(c_r)}$$

Resnik's measure is based on the Information Content (IC) of a concept c defined as the negative log probability $-\log P(c)$ of finding that concept in a large corpus:

$$res(c_i, c_r) = IC(lcs(c_i, c_r))$$

Lin's similarity measure can be seen as a normalized version of Resnik's information content:

$$lin(c_i, c_r) = \frac{2 * IC(lcs(c_i, c_r))}{IC(c_i) + IC(c_r)}$$

Jiang & Conrath's measure is closely related to *lin* and is computed as follows:

$$jcn(c_i, c_r) = [IC(c_i) + IC(c_r) - 2 * IC(lcs(c_i, c_r))]^{-1}$$

3.2 Structure Aware Measures

Cosine similarity, henceforth referred as *cos*, treats questions as bags-of-words. The meta-measure proposed in (Mihalcea et al., 2006), henceforth called *mcs*, treats questions as bags-of-concepts. Consequently, both *cos* and *mcs* may miss important structural information. If we consider the question Q_{24} below as reference, question Q_{26} will be deemed more useful than Q_{25} when using *cos* or *mcs* because of the higher relative lexical and conceptual overlap with Q_{24}. However, this is contrary to the actual ordering $\langle Q_{25} \succ Q_{26} | Q_{24} \rangle$, which reflects that fact that Q_{25}, which expects the same answer type as Q_{24}, should be deemed more useful than Q_{26}, which has a different answer type.

Q_{24} What are some good thriller *movies*?

Q_{25} What are some thriller *movies* with happy ending?

Q_{26} What are some good *songs* from a thriller movie?

The analysis above shows the importance of using the answer type when computing the similarity between two questions. However, instead of relying exclusively on a predefined hierarchy of answer types, we have decided to identify the *question focus* of a question, defined as the set of maximal noun phrases in the question that corefer with the expected answer. Focus nouns such as *movies* and *songs* provide more discriminative information than general answer types such as *products*. We use answer types only for questions such as Q_{27} or Q_{28} below that lack an explicit question focus. In such cases, an artificial question focus is created from the answer type (e.g. *location* for Q_{27}, or *method* for Q_{28}) and added to the set of question words.

Q_{27} *Where* can I buy a good coffee maker?

Q_{28} *How* do I make a pizza?

Let *qsim* be a general bag-of-words question similarity measure (e.g. *cos* or *mcs*). Furthermore, let *wsim* by a generic word meaning similarity measure (e.g. *wup*, *res*, *lin* or *jcn*). The equation below describes a modification of *qsim* that makes it aware of the questions focus:

$$\begin{aligned} qsim_f(Q_i, Q_r) &= wsim(f_i, f_r) * \\ & \quad qsim(Q_i - \{f_i\}, Q_r - \{f_r\}) \end{aligned}$$

Here, Q_i and Q_r refer both to the questions and their sets of words, while f_i and f_r stand for the corresponding focus words. We define $qsim$ to return 1 if one of its arguments is an empty set, i.e. $qsim(\emptyset, \_) = qsim(\_, \emptyset) = 1$. The new similarity measure $qsim_f$ multiplies the semantic similarity between the two focus words with the bag-of-words similarity between the remaining words in the two questions. Consequently, the word “movie” in Q_{26} will not be compared with the word “movies” in Q_{24}, and therefore Q_{26} will receive a lower utility score than Q_{25}.

In addition to the question focus, the *main verb* of a question can also provide key information in estimating question-to-question similarity. We define the main verb to be the content verb that is highest in the dependency tree of the question, e.g. *buy* for Q_{27}, or *make* for Q_{28}. If the question does not contain a content verb, the main verb is defined to be the highest verb in the dependency tree, as for example *are* in Q_{24} to Q_{26}. The utility of a question's main verb in judging its similarity to other questions can be seen more clearly in the questions below, where Q_{29} is the reference:

Q_{29} How can I *transfer* music from iTunes to my iPod?

Q_{30} How can I *upload* music to my iPod?

Q_{31} How can I *play* music in iTunes?

The fact that *upload*, as the main verb of Q_{30}, is more semantically related to *transfer* (*upload* is a hyponym of *transfer* in WordNet) is essential in deciding that $\langle Q_{30} \succ Q_{31} | Q_{29} \rangle$, i.e. Q_{30} is more useful than Q_{31} to Q_{29}.

Like the focus word, the main verb can be incorporated in the question similarity function as follows:

$$\begin{aligned} qsim_{fv}(Q_i, Q_r) &= wsim(f_i, f_r) * wsim(v_i, v_r) * \\ & \quad qsim(Q_i - \{f_i, v_i\}, Q_r - \{f_r, v_r\}) \end{aligned}$$

The new measure $qsim_{fv}$ takes into account both the focus words and the main verbs when estimating the semantic similarity between questions. When decomposing the questions into focus words, main verbs and the remaining words, we have chosen to multiply the corresponding similarities instead of, for example, summing them. Consequently, a close to zero score in each of them would drive the entire similarity to zero. This reflects the belief that question similarity is sensitive to each component of a question.

4 Experimental Evaluation

We use the question ranking dataset described in Section 2 to evaluate the two similarity measures cos and mcs, as well as their structured versions cos_f, cos_{fv}, mcs_f, and mcs_{fv}. We report one set of results for each of the four word similarity measures *wup*, *res*, *lin* or *jcn*. Each question similarity measure is evaluated in terms of its accuracy on the set of ordered pairs for each of the two annotation levels described in Section 2. Thus, for the first annotation level (L_1) , we evaluate only over the set of relations defined across the three

Question similarity ($qsim$)	Word similarity ($wsim$)							
	wup		*res*		*lin*		*jcn*	
	L_1	L_2	L_1	L_2	L_1	L_2	L_1	L_2
cos	69.1	69.3	69.1	69.3	69.1	69.3	69.1	69.3
cos_f	69.9	70.1	**72.5**	**72.7**	71.0	71.2	69.6	69.8
cos_{fv}	69.9	70.1	**72.5**	**72.6**	71.0	71.2	69.6	69.8
mcs	62.6	62.5	65.0	65.0	65.6	65.7	66.8	66.9
mcs_f	64.2	64.4	68.5	68.5	**68.8**	**68.9**	67.2	67.4
mcs_{fv}	65.8	66.0	68.8	68.8	**69.7**	**69.8**	67.7	67.8

Table 3: Accuracy results, with and without meaning and structure information.

sets $\mathcal{R}$, $\mathcal{U}$, and $\mathcal{N}$. If $\langle Q_i \succ Q_j | Q_r \rangle$ is a relation specified in the annotation, we consider the tuple $\langle Q_i, Q_j, Q_r \rangle$ correctly classified if and only if $u(Q_i, Q_r) > u(Q_j, Q_r)$, where u is the question similarity measure (Section 3). For the second annotation level (L_2), we also consider the relations annotated between *useful* questions inside the group $\mathcal{U}$.

We used the NLTK [1] implementation of the four similarity measures *wup*, *res*, *lin* or *jcn*. The *idf* values for each word were computed from frequency counts over the entire Wikipedia. For each question, the *focus* is identified automatically by an SVM tagger trained on a separate corpus of 2,000 questions manually annotated with focus information. The SVM tagger uses a combination of lexico-syntactic features and a quadratic kernel to achieve a 93.5% accuracy in a 10-fold cross validation evaluation on the 2,000 questions. The *main verb* of a question is identified deterministically using a breadth first traversal of the dependency tree.

The overall accuracy results presented in Table 3 show that using the focus word improves the performance across all 8 combinations of question and word similarity measures. For cosine similarity, the best performing system uses the focus words and Resnik's similarity function to obtain a 3.4% increase in accuracy. For the meaning aware similarity *mcs*, the best performing system uses the focus words, the main verb and Lin's word similarity to achieve a 4.1% increase in accuracy. The improvement due to accounting for focus words is consistent, whereas adding the main verb seems to improve the performance only for *mcs*, although not by a large margin. The second level of annotation brings 261 more relations in the dataset, some of them more difficult to annotate when compared with the three groups in the first level. Nevertheless, the performance either remains the same (somewhat expected due to the relatively small number of additional relations), or is marginally better. The random baseline – assigning a random similarity value to each pair of questions – results in 50% accuracy. A somewhat unexpected result is that *mcs* does not perform better than *cos* on this dataset. After analysing the result in more detail, we have noticed that *mcs* seems to be less resilient than *cos* to variations in the length of the questions. The Microsoft paraphrase corpus was specifically designed such that "the length of the shorter of the two sentences, in words, is at least 66% that of the longer" (Dolan and Brockett, 2005), whereas in our dataset the two questions in a pair can have significantly different lengths [2].

The questions in each of the 60 groups have a high degree of lexical overlap, making the dataset especially difficult. In this context, we believe the results are encouraging. We expect to obtain further improvements in accuracy by allowing relations between all the words in a question to influence the overall similarity measure. For example, question Q_{19} has the same focus word as the reference question Q_5 (repeated below), yet the difference between the focus word prepositional modifiers makes it a neutral question.

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

[2] Our implementation of *mcs* did performed better than *cos* on the Microsoft dataset.

Q_5 What's a good summer camp to go to in FL?

Q_{19} What's a good summer camp in Canada?

Some of the questions in our dataset illustrate the need to design a word similarity function specifically tailored to reflect how words change the relative usefulness of a question. In the set of questions below, in deciding that Q_{33} and Q_{34} are more useful than Q_{36} for the reference question Q_{32}, an ideal question ranker needs to know that the "Mayflower Hotel" and the "Queensboro Bridge" are in the proximity of "Midtown Manhattan", and that proximity relations are relevant when asking for directions. A coarse measure of proximity can be obtained for the pair ("Manhattan", "Queensboro Bridge") by following the *meronymy* links connecting the two entities in WordNet. However, a different strategy needs to be devised for entities such as "Mayflower Hotel", "JFK", or "La Guardia" which are not covered in WordNet.

Q_{32} What is the best way to get to Midtown Manhattan from JFK?

Q_{33} What's the best way from JFK to Mayflower Hotel?

Q_{34} What's the best way from JFK to Queensboro Bridge?

Q_{35} How do I get from Manhattan to JFK airport by train?

Q_{36} What is the best way to get to LaGuardia from JFK?

Finally, to realize why question Q_{35} is useful one needs to know that, once directions on how to get by train from location X to location Y are known, then normally it suffices to reverse the list of stops in order to obtain directions on how to get from Y back to X.

5 Future Work

We plan to integrate the entire dependency structure of the question in the overall similarity measure, possibly by defining kernels between questions in a maximum margin model for ranking. We also plan to extend the word similarity functions to better reflect the types of relations that are relevant when measuring question utility, such as proximity relations between locations. Furthermore, we intend to take advantage of databases of interrogative paraphrases and paraphrase patterns that were created in previous research on question reformulation.

6 Conclusion

We presented a novel question ranking task in which previously known questions are ordered based on their relative utility with respect to a new, reference question. We created a dataset of 60 groups of questions [3] annotated with a partial order relation reflecting the relative utility of questions inside each group, and used it to evaluate the ranking performance of several meaning and structure aware utility functions. Experimental results demonstrate the importance of using structural information in judging the relative usefulness of questions. We believe that the new perspective on ranking questions has the potential to significantly improve the usability of social QA sites.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful suggestions.

References

Baeza-Yates, Ricardo and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. ACM Press, New York.

Bernhard, Delphine and Iryna Gurevych. 2008. Answering learners' questions by retrieving question paraphrases from social Q&A sites. In *EANL '08: Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pages 44–52, Morristown, NJ, USA. Association for Computational Linguistics.

Dolan, William B. and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, pages 9–16.

[3] The dataset will be made publicly available.

Hermjakob, Ulf, Abdessamad Echihabi, and Daniel Marcu. 2002. Natural language based reformulation resource and web exploitation for question answering. In *Proceedings of TREC-2002*.

Jeon, Jiwoon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM'05)*, pages 84–90, New York, NY, USA. ACM.

Jiang, J.J. and D.W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics*, pages 19–33.

Jijkoun, Valentin and Maarten de Rijke. 2005. Retrieving answers from frequently asked questions pages on the Web. In *Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM'05)*, pages 76–83, New York, NY, USA. ACM.

Lin, Dekang. 1998. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML '98)*, pages 296–304, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Mihalcea, Rada, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st national conference on Artificial intelligence (AAAI'06)*, pages 775–780. AAAI Press.

Resnik, Philip. 1995. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*, pages 448–453, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Tomuro, Noriko. 2003. Interrogative reformulation patterns and acquisition of question paraphrases. In *Proceedings of the Second International Workshop on Paraphrasing*, pages 33–40, Morristown, NJ, USA. Association for Computational Linguistics.

Wu, Zhibiao and Martha Palmer. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138, Morristown, NJ, USA. Association for Computational Linguistics.

Zhao, Shiqi, Ming Zhou, and Ting Liu. 2007. Learning question paraphrases for QA from Encarta logs. In *Proceedings of the 20th international joint conference on Artifical intelligence (IJCAI'07)*, pages 1795–1800, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Simultaneous Ranking and Clustering of Sentences: A Reinforcement Approach to Multi-Document Summarization

[1]Xiaoyan Cai, [1]Wenjie Li, [1]You Ouyang, [2]Hong Yan

[1]Department of Computing, The Hong Kong Polytechnic University
{csxcai,cswjli,csyouyang}@comp.polyu.edu.hk

[2]Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University
lgthyan@polyu.edu.hk

Abstract

Multi-document summarization aims to produce a concise summary that contains salient information from a set of source documents. In this field, sentence ranking has hitherto been the issue of most concern. Since documents often cover a number of topic themes with each theme represented by a cluster of highly related sentences, sentence clustering was recently explored in the literature in order to provide more informative summaries. Existing cluster-based ranking approaches applied clustering and ranking in isolation. As a result, the ranking performance will be inevitably influenced by the clustering result. In this paper, we propose a reinforcement approach that tightly integrates ranking and clustering by mutually and simultaneously updating each other so that the performance of both can be improved. Experimental results on the DUC datasets demonstrate its effectiveness and robustness.

1 Introduction

Automatic multi-document summarization has drawn increasing attention in the past with the rapid growth of the Internet and information explosion. It aims to condense the original text into its essential content and to assist in filtering and selection of necessary information. So far extractive summarization that directly extracts sentences from documents to compose summaries is still the mainstream in this field. Under this framework, sentence ranking is the issue of most concern.

Though traditional feature-based ranking approaches and graph-based approaches employed quite different techniques to rank sentences, they have at least one point in common, i.e., all of them focused on sentences only, but ignored the information beyond the sentence level (referring to Figure 1(a)). Actually, in a given document set, there usually exist a number of themes (or topics) with each theme represented by a cluster of highly related sentences (Harabagiu and Lacatusu, 2005; Hardy et al., 2002). These theme clusters are of different size and especially different importance to assist users in understanding the content in the whole document set. The cluster level information is supposed to have foreseeable influence on sentence ranking.

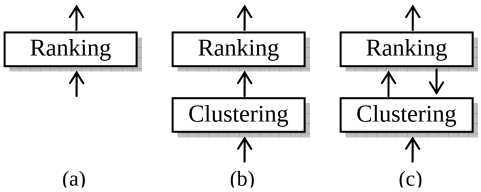


Figure 1. Ranking vs. Clustering

In order to enhance the performance of summarization, recently cluster-based ranking approaches were explored in the literature (Wan and Yang, 2006; Sun et al, 2007; Wang et al, 2008a,b; Qazvinian and Radev, 2008). Normally these approaches applied a clustering algorithm to obtain the theme clusters first and then ranked the sentences within each cluster or by exploring the interaction between sentences and obtained clusters (referring to Figure 1(b)). In other words, clustering and ranking are regarded as two independent processes in these approaches although the cluster-level information has been incorporated into the sentence ranking process. As a result,

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 134–142,
Beijing, August 2010

the ranking performance is inevitably influenced by the clustering result.

To help alleviate this problem, we argue in this paper that the quality of ranking and clustering can be both improved when the two processes are mutually enhanced (referring to Figure 1(c)). Based on it, we propose a reinforcement approach that updates ranking and clustering interactively and iteratively to multi-document summarization. The main contributions of the paper are three-fold: (1) Three different ranking functions are defined in a bi-type document graph constructed from the given document set, namely global, within-cluster and conditional rankings, respectively. (2) A reinforcement approach is proposed to tightly integrate ranking and clustering of sentences by exploring term rank distributions over the clusters. (3) Thorough experimental studies are conducted to verify the effectiveness and robustness of the proposed approach.

The rest of this paper is organized as follows. Section 2 reviews related work in cluster-based ranking. Section 3 defines ranking functions and explains reinforced ranking and clustering process and its application in multi-document summarization. Section 4 presents experiments and evaluations. Section 5 concludes the paper.

2 Related Work

Clustering has become an increasingly important topic with the explosion of information available via the Internet. It is an important tool in text mining and knowledge discovery. Its ability to automatically group similar textual objects together enables one to discover hidden similarity and key concepts, as well as to summarize a large amount of text into a small number of groups (Karypis et al., 2000).

To summarize a scientific paper, Qazvinian and Radev (2008) presented two sentence selection strategies based on the clusters which were generated by a hierarchical agglomeration algorithm applied in the citation summary network. One was called C-RR, which started with the largest cluster and extracted the first sentence from each cluster in the order they appeared until the summary length limit was reached. The other was called C-LexRank, which was similar to C-RR but adopted LexRank to rank the sentences within each cluster and chose the most salient one.

Meanwhile, Wan and Yang (2008) proposed two models to incorporate the cluster-level information into the process of sentence ranking for generic summarization. While the Cluster-based Conditional Markov Random Walk model (ClusterCMRW) incorporated the cluster-level information into the text graph and manipulated clusters and sentences equally, the Cluster-based HITS model (ClusterHITS) treated clusters and sentences as hubs and authorities in the HITS algorithm.

Besides, Wang et al. (2008) proposed a language model to simultaneously cluster and summarize documents. Nonnegative factorization was performed on the term-document matrix using the term-sentence matrix as the base so that the document-topic and sentence-topic matrices could be constructed, from which the document clusters and the corresponding summary sentences were generated simultaneously.

3 A Reinforcement Approach to Multi-document Summarization

3.1 Document Bi-type Graph

First of all, let's introduce the sentence-term bi-type graph model for a set of given documents *D*, based on which the algorithm of reinforced ranking and clustering is developed. Let $G = <V, E, W>$, where V is the set of vertices that consists of the sentence set $S = \{s_1, s_2, \ldots, s_n\}$ and the term set $T = \{t_1, t_2, \ldots, t_m\}$, i.e., $V = S \cup T$, E is the set of edges that connect the vertices, i.e., $E = \{< v_i, v_j > | v_i, v_j \in V\}$. W is the adjacency matrix in which the element w_{ij} represents the weight of the edge connecting v_i and v_j. Formally, W can be decomposed into four blocks, i.e., W_{SS}, W_{ST}, W_{TS} and W_{TT}, each representing a sub-graph of the textual objects indicated by the subscripts. W can be written as

$$W = \begin{pmatrix} W_{SS} & W_{ST} \\ W_{TS} & W_{TT} \end{pmatrix},$$

where $W_{ST}(i, j)$ is the number of times the term t_j appears in the sentence s_i. $W_{SS}(i,j)$ is the number of common terms in the sentences s_i and s_j . W_{TS} is equal to W_{ST}^T as the relationships between terms and sentences are symmetric. For simplification, in this study we assume there is no direct relationships between terms, i.e., $W_{TT} = 0$. In the future, we will explore effective ways to integrate term semantic relationships into the model.

3.2 Basic Ranking Functions

Recall that our ultimate goal is sentence ranking. As an indispensable part of the approach, the basic ranking functions need to be defined first.

3.2.1 Global Ranking (without Clustering)

Let $r(s_i)$ (*i*=1, 2, …, *n*) and $r(t_j)$ (*j*=1, 2, …, *m*) denote the ranking scores of the sentence s_i and the term t_j in the whole document set, respectively. Based on the assumptions that

> *"Highly ranked terms appear in highly ranked sentences, while highly ranked sentences contain highly ranked terms. Moreover, a sentence is ranked higher if it contains many terms that appear in many other highly ranked sentences."*

we define

$$r(s_i) = \lambda \cdot \sum_{j=1}^{m} W_{ST}(i,j) \cdot r(t_j) + (1-\lambda) \cdot \sum_{j=1}^{n} W_{SS}(i,j) \cdot r(s_j) \quad (1)$$

and

$$r(t_j) = \sum_{i=1}^{n} W_{TS}(j,i) \cdot r(s_i) . \quad (2)$$

For calculation purpose, $r(s_i)$ and $r(t_j)$ are normalized by

$$r(s_i) \leftarrow \frac{r(s_i)}{\sum_{i'=1}^{n} r(s_{i'})} \text{ and } r(t_j) \leftarrow \frac{r(t_j)}{\sum_{j'=1}^{m} r(t_{j'})} .$$

Equations (1) and (2) can be rewritten using the matrix form, i.e.,

$$\begin{cases} r(S) = \lambda \cdot \dfrac{W_{ST} \cdot r(T)}{\| W_{ST} \cdot r(T) \|} + (1-\lambda) \cdot \dfrac{W_{SS} \cdot r(S)}{\| W_{SS} \cdot r(S) \|} \\ r(T) = \dfrac{W_{TS} \cdot r(S)}{\| W_{TS} \cdot r(S) \|} \end{cases} . \quad (3)$$

We call $r(S)$ and $r(T)$ the "**global ranking functions**", because at this moment sentence clustering is not yet involved and all the sentences/terms in the whole document set are ranked together.

Theorem: The solution to $r(S)$ and $r(T)$ given by Equation (3) is the primary eigenvector of $\lambda \cdot W_{ST} \cdot W_{TS} + (1-\lambda) \cdot W_{SS}$ and $\lambda \cdot W_{TS}(I-(1-\lambda) \cdot W_{SS})^{-1} \cdot W_{ST}$, respectively.

Proof: Combine Equations (1) and (2), we get

$$r(S) = \lambda \cdot \frac{W_{ST} \cdot \dfrac{W_{TS} \cdot r(S)}{\| W_{TS} \cdot r(S) \|}}{\| W_{ST} \cdot \dfrac{W_{TS} \cdot r(S)}{\| W_{TS} \cdot r(S) \|} \|} + (1-\lambda) \cdot \frac{W_{SS} \cdot r(S)}{\| W_{SS} \cdot r(S) \|}$$

$$= \lambda \cdot \frac{W_{ST} \cdot W_{TS} \cdot r(S)}{\| W_{ST} \cdot W_{TS} \cdot r(S) \|} + (1-\lambda) \cdot \frac{W_{SS} \cdot r(S)}{\| W_{SS} \cdot r(S) \|}$$

As the iterative process is a power method, it is guaranteed that $r(S)$ converges to the primary eigenvector of $\lambda \cdot W_{ST} \cdot W_{TS} +$ $(1-\lambda) \cdot W_{SS}$. Similarly, $r(T)$ is guaranteed to converge to the primary eigenvector of $\lambda \cdot W_{TS}(I-(1-\lambda) \cdot W_{SS})^{-1} \cdot W_{ST}$. ■

3.2.2 Local Ranking (within Clusters)

Assume now K theme clusters have been generated by certain clustering algorithm, denoted as $C = \{C_1, C_2, \ldots, C_K\}$ where C_k (*k*=1, 2, …, *K*) represents a cluster of highly related sentences S_{C_k} ($\in C_k$) which contain the terms T_{C_k} ($\in C_k$) . The sentences and terms within the cluster C_k form a cluster bi-type graph with the adjacency matrix W_{C_k} . Let $r_{C_k}(S_{C_k})$ and $r_{C_k}(T_{C_k})$ denote the ranking scores of S_{C_k} and T_{C_k} within C_k . They are calculated by an equation similar to Equation (3) by replacing the document level adjacency matrix W with the cluster level adjacency matrix W_{C_k} . We call $r_{C_k}(S_{C_k})$ and $r_{C_k}(T_{C_k})$ the "**within-cluster ranking functions**" with respect to the cluster C_k . They are the local ranking functions, in contrast to $r(S)$ and $r(T)$ that rank all the sentences and terms in the whole document set *D*. We believe that it will benefit sentence overall ranking when knowing more details about the ranking results at the finer granularity of theme clusters, instead of at the coarse granularity of the whole document set.

3.2.3 Conditional Ranking (across Clusters)

To facilitate the discovery of rank distributions of terms and sentences over all the theme clusters, we further define two "**conditional ranking functions**" $r(S \mid C_k)$ and $r(T \mid C_k)$. These rank distributions are necessary for the parameter estimation during the reinforcement process introduced later. The conditional ranking score of the term t_j on the cluster C_k, i.e., $r(T \mid C_k)$ is directly derived from T_{C_k}, i.e., $r(t_j \mid C_k) = r_{C_k}(t_j)$ if $t_j \in C_k$, and $r(t_j \mid C_k) = 0$ otherwise. It is further normalized as

$$r(t_j \mid C_k) = \frac{r(t_j \mid C_k)}{\sum_{j=1}^{m} r(t_j \mid C_k)}. \qquad (4)$$

Then the conditional ranking score of the sentence s_i on the cluster C_k is deduced from the terms that are included in s_i, i.e.,

$$r(s_i \mid C_k) = \frac{\sum_{j=1}^{m} W_{ST}(i,j) \cdot r(t_j \mid C_k)}{\sum_{i=1}^{n} \sum_{j=1}^{m} W_{ST}(i,j) \cdot r(t_j \mid C_k)}. \qquad (5)$$

Equation (5) can be interpreted as that the conditional rank of s_i on C_k is higher if many terms in s_i are ranked higher in C_k. Now we have sentence and term conditional ranks over all the theme clusters and are ready to introduce the reinforcement process.

3.3 Reinforcement between Within-Cluster Ranking and Clustering

The conditional ranks of the term t_j across the K theme clusters can be viewed as a rank distribution. Then the rank distribution of the sentence s_i can be considered as a mixture model over K conditional rank distributions of the terms contained in the sentence s_i. And the sentence s_i can be represented as a K-dimensional vector in the new measure space, in which the vectors can be used to guide the sentence clustering update. Next, we will explain the mixture model of sentence and use EM algorithm (Bilmes, 1997) to get the component coefficients of the model. Then, we will present the similarity measure between sentence and cluster, which is used to adjust the clusters that the sentences belong to and in turn modify within-cluster ranking for the sentences in the updated clusters.

3.3.1 Sentence Mixture Model

For each sentence s_i, we assume that it follows the distribution $r(T \mid s_i)$ to generate the relationship between the sentence s_i and the term set *T*. This distribution can be considered as a mixture model over K component distributions, i.e. the term conditional rank distributions across K theme clusters. We use $\gamma_{i,k}$ to denote the probability that s_i belongs to C_k, then $r(T \mid s_i)$ can be modeled as:

$$r(T \mid s_i) = \sum_{k=1}^{K} \gamma_{i,k} \cdot r(T \mid C_k) \text{ and } \sum_{k=1}^{K} \gamma_{i,k} = 1. \qquad (6)$$

$\gamma_{i,k}$ can be explained as $p(C_k \mid s_i)$ and calculated by the Bayesian equation $p(C_k \mid s_i) \propto p(s_i \mid C_k) \cdot p(C_k)$, where $p(s_i \mid C_k)$ is assumed to be $r(s_i \mid C_k)$ obtained from the conditional rank of s_i on C_k as introduced before and $p(C_k)$ is the prior probability.

3.3.2 Parameter Estimation

We use EM algorithm to estimate the component coefficients $\gamma_{i,k}$ along with $\{p(C_k)\}$. A hidden variable C_z, $z \in \{1,2,\ldots,K\}$ is used to denote the cluster label that a sentence term pair (s_i, t_j) are from. In addition, we make the independent assumption that the probability of s_i belonging to C_k and the probability of t_j belonging to C_k are independent, i.e., $p(s_i, t_j \mid C_k) = p(s_i \mid C_k) \cdot p(t_j \mid C_k)$, where $p(s_i, t_j \mid C_k)$ is the probability of s_i and t_j both belonging to C_k. Similarly, $p(t_j \mid C_k)$ is assumed to be $r(t_j \mid C_k)$.

Let Θ be the parameter matrix, which is a $n \times K$ matrix $\Theta_{n \times K} = \{\gamma_{i,k}\}$ $(i = 1,\ldots,n;$ $k = 1,\ldots,K)$. The best Θ is estimated from the relationships observed in the document bi-type graph, i.e., W_{ST} and W_{SS}. The likelihood of generating all the relationships under the parameter Θ can be calculated as:

$$L'(\Theta \mid W_{ST}, W_{SS}) = p(W_{ST} \mid \Theta) \cdot p(W_{SS} \mid \Theta)$$
$$= \prod_{i=1}^{n} \prod_{j=1}^{m} p(s_i, t_j \mid \Theta)^{W_{ST}(i,j)} \cdot \prod_{i=1}^{n} \prod_{j=1}^{n} p(s_i, s_j \mid \Theta)^{W_{SS}(i,j)}$$

where $p(s_i,t_j|\Theta)$ is the probability that s_i and t_j both belong to the same cluster, given the current parameter. As $p(s_i,s_j|\Theta)$ does not contain variables from Θ, we only need to consider maximizing the first part of the likelihood in order to get the best estimation of Θ. Let $L(\Theta|W_{ST})$ be the first part of likelihood.

Taking into account the hidden variable C_z, the complete log-likelihood can be written as

$$\log L(\Theta|W_{ST},C_Z)=\log\prod_{i=1}^{n}\prod_{j=1}^{m}\left(p(s_i,t_j,C_z|\Theta)\right)^{W_{ST}(i,j)}$$

$$=\log\prod_{i=1}^{n}\prod_{j=1}^{m}\left(p(s_i,t_j|C_z,\Theta)\cdot p(C_z|\Theta)\right)^{W_{ST}(i,j)} .$$

$$=\sum_{i=1}^{n}\sum_{j=1}^{m}W_{ST}(i,j)\cdot\log\left(p_Z(s_i,t_j)\cdot p(C_z|\Theta)\right)$$

In the E-step, given the initial parameter Θ^0, which is set to $\gamma_{i,k}^0={}^1\!/_K$ for all *i* and *k*, the expectation of log-likelihood under the current distribution of C_Z is:

$$Q(\Theta,\Theta^0)=E_{f(C_Z|W_{ST},\Theta^0)}(\log L(\Theta|W_{ST},C_Z)$$

$$=\sum_{k=1}^{K}\sum_{i=1}^{n}\sum_{j=1}^{m}W_{ST}(i,j)\cdot\log(p_k(s_i,t_j))\cdot p(C_z=C_k|s_i,t_j,\Theta^0)+$$

$$\sum_{i=1}^{n}\sum_{k=1}^{K}\sum_{j=1}^{m}W_{ST}(i,j)\cdot\log(p(C_z=C_k|\Theta))\cdot p(C_z=C_k|s_i,t_j,\Theta^0)$$

The conditional distribution in the above equation, i.e., $p(C_z=C_k|s_i,t_j,\Theta^0)$, can be calculated using the Bayesian rule as follows:

$$\begin{aligned}&p(C_z=C_k|s_i,t_j,\Theta^0)\\&\propto p(s_i,t_j|C_z=C_k,\Theta^0)p(C_z=C_k|\Theta^0)\cdot\qquad(7)\\&\propto p^0(s_i|C_k)p^0(t_j|C_k)p^0(C_z=C_k)\end{aligned}$$

In the M-Step, we first get the estimation of $p(C_z=C_k)$ by maximizing the expectation $Q(\Theta,\Theta^0)$. By introducing a Lagrange multiplier λ, we get the equation below.

$$\frac{\partial}{\partial p(C_z=C_k)}[Q(\Theta,\Theta^0)+\lambda(\sum_{k=1}^{K}p(C_z=C_k)-1)]=0\Rightarrow$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m}W_{ST}(i,j)\frac{1}{p(C_z=C_k)}p(C_z=C_k|s_i,t_j,\Theta^0)+\lambda=0$$

Thus, the estimation of $p(C_z=C_k)$ given previous Θ^0 is

$$p(C_z=C_k)=\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}W_{ST}(i,j)p(C_z=C_k|s_i,t_j,\Theta^0)}{\sum_{i=1}^{n}\sum_{j=1}^{m}W_{ST}(i,j)}\cdot\quad(8)$$

Then, the parameters $\gamma_{i,k}$ can be calculated with the Bayesian rule as

$$\gamma_{i,k}=\frac{p(s_i|C_k)p(C_z=C_k)}{\sum_{l=1}^{K}p(s_i|C_l)p(C_z=C_l)}\cdot\quad(9)$$

By setting $\Theta^0=\Theta$, the whole process can be repeated. The updating rules provided in Equations (7)-(9) are applied at each iteration. Finally Θ will converge to a local maximum. A similar estimation process has been adopted in (Sun et al., 2009), which was used to estimate the component coefficients for author-conference networks.

3.3.3 Similarity Measure

After we get the estimations of the component coefficients $\gamma_{i,k}$ for s_i, s_i will be represented as a K dimensional vector $\vec{s_i}=(\gamma_{i,1},\gamma_{i,2},\ldots,\gamma_{i,K})$. The center of each cluster can thus be calculated accordingly, which is the mean of $\vec{s_i}$ for all s_i in the same cluster, i.e.,

$$\overrightarrow{Center}_{C_k}=\frac{\sum_{s_i\in C_k}\vec{s_i}}{|C_k|},$$

where $|C_k|$ is the size of C_k.

Then the similarity between each sentence and each cluster can be calculated as the cosine similarity between them, i.e.,

$$sim(s_i,C_k)=\frac{\sum_{l=1}^{K}\vec{s_i}(l)\overrightarrow{Center}_{C_k}(l)}{\sqrt{\sum_{l=1}^{K}\vec{s_i}(l))^2}\sqrt{\sum_{l=1}^{K}\overrightarrow{Center}_{C_k}(l))^2}}\cdot\quad(10)$$

Finally, each sentence is re-assigned to a cluster that is the most similar to the sentence. Based on the updated clusters, within-cluster ranking is updated accordingly, which triggers the next round of clustering refinement. It is expected that the quality of clusters should be improved during this iterative update process since the similar sentences under new attributes will be grouped together, and meanwhile the quality of ranking will be improved along with the better clusters and thus offers better attributes for further clustering.

3.4 Ensemble Ranking

The overall sentence ranking function f is defined as the ensemble of all the sentence conditional ranking scores on the K clusters.

$$f(s_i) = \sum_{k=1}^{K} \alpha_k \cdot r(s_i \mid C_k)\,, \tag{11}$$

where α_k is a coefficient evaluating the importance of C_k. It can be formulated as the normalized cosine similarity between a theme cluster and the whole document set for generic summarization, or between a theme cluster and a given query for query-based summarization. $\alpha_k \in [0,1]$ and $\sum_{k=1}^{K} \alpha_k = 1$.

Figure 2 below summarizes the whole process that determines the overall sentence ensemble ranking scores.

```
Input: The bi-type document graph $G = < S \cup T, E, W >$,
       ranking functions, the cluster number $K$, $\varepsilon = 1$,
       $Tre = 0.001$, $IterNum = 10$.
Output: sentence final ensemble ranking vector $f(S)$.
$t \leftarrow 0$;
Get the initial partition for $S$, i.e. $C_k^t$, $k = 1,2,\ldots K$,
  calculate cluster centers $\overrightarrow{Center}_{C_k^t}$ accordingly.
For ($t$=1; $t$<IterNum && $\varepsilon > Tre$; $t$++)
    Calculate the within-cluster ranking $r_{C_k}(T_{C_k})$,
    $r_{C_k}(S_{C_k})$ and the conditional ranking $r(s_i \mid C_k)$;
    Get new attribute $\vec{s}_i$ for each sentence $s_i$, and
    new attribute $\overrightarrow{Center}_{C_k^t}$ for each cluster $C_k^t$;
    For each sentence $s_i$ in $S$
        For $k$=1 to $K$
            Calculate similarity value $sim(s_i, C_k^t)$
        End For
        Assign $s_i$ to $C_{k_0}^{t+1}$, $k_0 = \arg\max_k sim(s_i, C_k^t)$
    End For
    $\varepsilon = \max_k |\overrightarrow{Center}_{C_k^{t+1}} - \overrightarrow{Center}_{C_k^t}|$
    $t \leftarrow t+1$
End For
For each sentence $s_i$ in $S$
    For $k$=1 to $K$
        $f(s_i) = \sum_{k=1}^{K} \alpha_k \cdot r(s_i \mid C_k)$
    End For
End For
```

Figure 2. The Overall Sentence Ranking Algorithm

3.5 Summary Generation

In multi-document summarization, the number of documents to be summarized can be very large. This makes information redundancy appears to be more serious in multi-document summarization than in single-document summarization. Redundancy control is necessary. We apply a simple yet effective way to choose summary sentences. Each time, we compare the current candidate sentence to the sentences already included in the summary. Only the sentence that is not too similar to any sentence in the summary (i.e., the cosine similarity between them is lower than a threshold) is selected into the summary. The iteration is repeated until the length of the sentences in the summary reaches the length limitation. In this paper, the threshold is set to 0.7 as always in our past work.

4 Experiments and Evaluations

We conduct the experiments on the DUC 2004 generic multi-document summarization dataset and the DUC 2006 query-based multi-document summarization dataset. According to task definitions, systems are required to produce a concise summary for each document set (without or with a given query description) and the length of summaries is limited to 665 bytes in DUC 2004 and 250 words in DUC 2006.

A well-recognized automatic evaluation toolkit ROUGE (Lin and Hovy, 2003) is used in evaluation. It measures summary quality by counting overlapping units between system-generated summaries and human-written reference summaries. We report two common ROUGE scores in this paper, namely ROUGE-1 and ROUGE-2, which base on Uni-gram match and Bi-gram match, respectively. Documents and queries are pre-processed by segmenting sentences and splitting words. Stop words are removed and the remaining words are stemmed using Porter stemmer.

4.1 Evaluation of Performance

In order to evaluate the performance of reinforced clustering and ranking approach, we compare it with the other three ranking approaches: (1) Global-Rank, which does not apply clustering and simply relies on the sentence global ranking scores to select summary sentences; (2) Local-Rank, which clusters sentences first and then rank sentences within each cluster. A summary is generated in the same way as presented in (Qazvinian and Radev, 2008). The clusters are ordered by decreasing size; (3) Cluster-HITS, which also clusters sentences first, but then regards clusters as hubs and sentences as authorities in the HITS algorithm and uses the obtained authority scores to rank and select sentences. The classical clustering algorithm K-means is used where necessary. For query-based summarization, the additional query-relevance (i.e. the cosine similarity between sentences and query) is involved to re-rank the candidate sentences chosen by the ranking approaches for generic summarization.

Note that K-means requires a predefined cluster number *K*. To avoid exhaustive search for a proper cluster number for each document set, we employ the spectra approach introduced in (Li et al., 2007) to predict the number of the expected clusters. Based on the sentence similarity matrix using the normalized 1-norm, for its eigenvalues λ_i (*i*=1,2, …, *n*), the ratio $\alpha_i = \lambda_{i+1} / \lambda_2 (\lambda \geq 1)$ is defined. If $\alpha_i - \alpha_{i+1} > 0.05$ and α_i is still close to 1, then set *K*=*i*+1. Tables 1 and 2 below compare the performance of the four approaches on DUC 2004 and 2006 according to the calculated *K*.

DUC 2004	ROUGE-1	ROUGE-2
Reinforced	0.37082	0.08351
Cluster-HITS	0.36463	0.07632
Local-Rank	0.36294	0.07351
Global-Rank	0.35729	0.06893

Table 1. Results on the DUC 2004 dataset

DUC 2006	ROUGE-1	ROUGE-2
Reinforced	0.39531	0.08957
Cluster-HITS	0.38315	0.08632
Local-Rank	0.38104	0.08841
Global-Rank	0.37478	0.08531

Table 2. Results on the DUC 2006 dataset

It is not surprised to find that "Global-Rank" shows the poorest performance, when it utilizes the sentence level information only whereas the other three approaches all integrate the additional cluster level information in various ways. In addition, as results illustrate, the performance of "Cluster-HITS" is better than the performance of "Local-Rank". This can be mainly credited to the ability of "Cluster-HITS" to consider not only the cluster-level information, but also the sentence-to-cluster relationships, which are ignored in "Local-Rank". It is happy to see that the proposed reinforcement approach, which simultaneously updates clustering and ranking of sentences, consistently outperforms the other three approaches.

4.2 Analysis of Cluster Quality

Our original intention to propose the reinforcement approach is to hope to generate more accurate clusters and ranking results by mutually refining within-cluster ranking and clustering. In order to check and monitor the variation trend of the cluster quality during the iterations, we define the following measure

$$quan = \sum_{k=1}^{K} \left(\frac{\min\limits_{s_i \in C_k} sim(s_i, C_k)}{\sum\limits_{l=1, l \neq k}^{K} \min\limits_{s_i \in C_k, s_j \in C_l} sim(s_i, s_j)} \right), \quad (12)$$

where $\min\limits_{s_i \in C_k} sim(s_i, C_k)$ denotes the distance between the cluster center and the border sentence in a cluster that is the farthest away from the center. The larger it is, the more compact the cluster is. $\min\limits_{s_i \in C_k, s_j \in C_l} sim(s_i, s_j)$, on the other hand, denotes the distance between the most distant pair of sentences, one from each cluster. The smaller it is, the more separated the two clusters are. The distance is measured by cosine similarity. As a whole, the larger *quan* means the better cluster quality. Figure 3 below plots the values of *quan* in each iteration on the DUC 2004 and 2006 datasets. Note that the algorithm converges in less than 6 rounds and 5 rounds on the DUC 2004 and 2006 datasets, respectively. The curves clearly show the increasment of *quan* and thus the improved cluster quality.

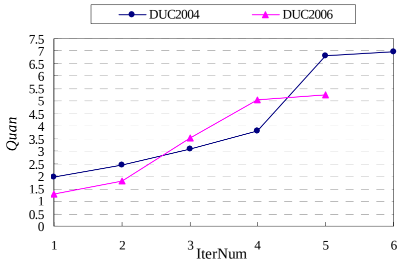


Figure 3. Cluster Quality on DUC 2004 and 2006

While *quan* directly evaluate the quality of the generated clusters, we are also quite interested in whether the improved clusters quality can further enhance the quality of sentence ranking and thus consequently raise the performance of summarization. Therefore, we evaluate the ROUGEs in each iteration as well. Figure 4 below illustrates the changes of ROUGE-1 and ROUGE-2 result on the DUC 2004 and 2006 datasets, respectively. Now, we have come to the positive conclusion.

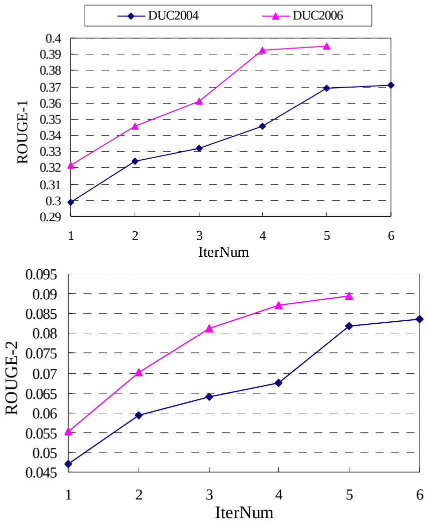


Figure 4. ROUGEs on DUC 2004 and 2006

4.3 Impact of Cluster Numbers

In previous experiments, the cluster number is predicted through the eigenvalues of 1-norm normalized sentence similarity matrix. This number is just the estimated number. The actual number is hard to predict accurately. To further examine how the cluster number influences summarization, we conduct the following additional experiments by varying the cluster number. Given a document set, we let S denote the sentence set in the document set, and set K in the following way:

$$K = \varepsilon \times |S|, \qquad (13)$$

where $\varepsilon \in (0,1)$ is a ratio controlling the expected cluster number. The larger ε is, the more clusters will be produced. ε ranges from 0.1 to 0.9 in the experiments. Due to page limitation, we only provide the ROUGE-1 and ROUGE-2 results of the proposed approach, “Cluster-HITS” and “Local-Rank” on the DUC 2004 dataset in Figure 5. The similar curves are also observed on the 2006 dataset.

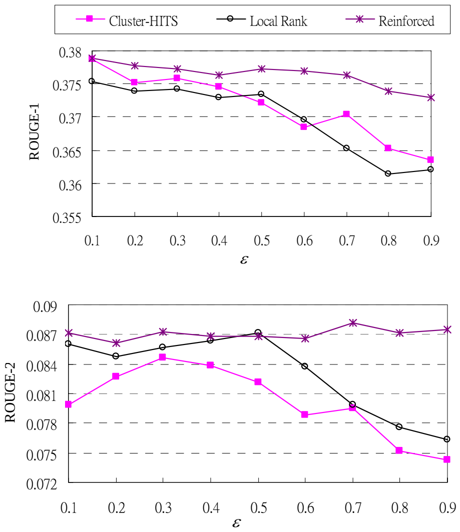


Figure 5. ROUGEs vs. ε on DUC 2004

It is shown that (1) the proposed approach outperforms “Cluster-HITS” and “Local-Rank” in almost all the cases no matter how the cluster number is set; (2) the performances of “Cluster-HITS” and “Local-Rank” are more sensitive to the cluster number and a large number of clusters appears to deteriorate the performances of both. This is reasonable. Actually when ε getting close to 1, “Local-Rank” approaches to “Global-Rank”. These results demonstrate the robustness of the proposed approach.

5 Conclusion

In this paper, we present a reinforcement approach that tightly integrates ranking and clustering together by mutually and simultaneously updating each other. Experimental results demonstrate the effectiveness and the robustness of the proposed approach. In the future, we will explore how to integrate term semantic relationships to further improve the performance of summarization.

Acknowledgement

The work described in this paper was supported by an internal grant from the Hong Kong Polytechnic University (G-YG80).

References

J. Bilmes. 1997. *A Gentle Tutorial on the em Algorithm and Its Application to Parameter Wstimation for Gaussian Mixture and Hidden Markov Models*. Technical Report ICSI-TR-97-02, University of Berkeley.

Brin, S., and Page, L. 1998. *The Anatomy of a Large-scale Hypertextual Web Search Engine*. In Proceedings of WWW1998..

Harabagiu S. and Lacatusu F. 2005. *Topic Themes for Multi-Document Summarization*. In Proceedings of SIGIR2005.

Hardy H., Shimizu N., Strzalkowski T., Ting L., Wise G. B., and Zhang X. 2002. *Cross-Document Summarization by Concept Classification*. In Proceedings of SIGIR2002.

Jon M. Kleinberg. 1999. *Authoritative Sources in a Hyperlinked Environment*. In Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms.

Karypis, George, Vipin Kumar and Michael Steinbach. 2000. *A Comparison of Document Clustering Techniques*. KDD workshop on Text Mining.

Lin, C. Y. and Hovy, E. 2000. *The Automated Acquisition of Topic Signature for Text Summarization*. In Proceedings of COLING2000.

Li W.Y., Ng W.K., Liu Y. and Ong K.L. 2007. *Enhancing the Effectiveness of Clustering with Spectra Analysis*. IEEE Transactions on Knowledge and Data Engineering (TKDE). 19(7): 887-902.

Li, F., Tang, Y., Huang, M., Zhu, X. 2009. *Answering Opinion Questions with Random Walks on Graphs*. In Proceedings of ACL2009.

Otterbacher J., Erkan G. and Radev D. 2005. *Using RandomWalks for Question-focused Sentence Retrieval*. In Proceedings of HLT/EMNLP 2005.

Qazvinian V. and Radev D. R. 2008. *Scientific paper summarization using citation summary networks*. In Proceedings of COLING2008.

Sun P., Lee J.H., Kim D.H., and Ahn C.M. 2007. *Multi-Document Using Weighted Similarity Between Topic and Clustering-Based Non-negative Semantic Feature*. APWeb/WAIM 2007.

Sun Y., Han J., Zhao P., Yin Z., Cheng H., and Wu T. 2009. *Rankclus: Integrating Clustering with Ranking for Heterogenous Information Network Analysis*. In Proceedings of EDBT 2009.

Wang D.D., Li T., Zhu S.H., Ding Chris. 2008a *Multi-Document Summarization via Sentence-Level Semantic Analysis and Symmetric Matrix Factorization*. In Proceedings of SIGIR2008.

Wang D.D., Zhu S.H., Li T., Chi Y., and Gong Y.H. 2008b. *Integrating Clustering and Multi-Document Summarization to Improve Document Understanding*. In Proceedings of CIKM 2008.

Wan X. and Yang J. 2006. *Improved Affinity Graph based Multi-Document Summarization*. In Proceedings of HLT-NAACL2006.

Zha H. 2002. *Generic Summarization and Key Phrase Extraction using Mutual Reinforcement Principle and Sentence Clustering*. In Proceedings of SIGIR2002.

End-to-End Coreference Resolution via Hypergraph Partitioning

Jie Cai and **Michael Strube**
Natural Language Processing Group
Heidelberg Institute for Theoretical Studies gGmbH
(jie.cai|michael.strube)@h-its.org

Abstract

We describe a novel approach to coreference resolution which implements a global decision via hypergraph partitioning. In constrast to almost all previous approaches, we do not rely on separate classification and clustering steps, but perform coreference resolution globally in one step. Our hypergraph-based global model implemented within an end-to-end coreference resolution system outperforms two strong baselines (Soon et al., 2001; Bengtson & Roth, 2008) using system mentions only.

1 Introduction

Coreference resolution is the task of grouping mentions of entities into sets so that all mentions in one set refer to the same entity. Most recent approaches to coreference resolution divide this task into two steps: (1) a classification step which determines whether a pair of mentions is coreferent or which outputs a confidence value, and (2) a clustering step which groups mentions into entities based on the output of step 1.

The classification steps of most approaches vary in the choice of the classifier (e.g. decision tree classifiers (Soon et al., 2001), maximum entropy classification (Luo et al., 2004), SVM classifiers (Rahman & Ng, 2009)) and the number of features used (Soon et al. (2001) employ a set of twelve simple but effective features while e.g., Ng & Cardie (2002) and Bengtson & Roth (2008) devise much richer feature sets).

The clustering step exhibits much more variation: Local variants utilize a closest-first decision (Soon et al., 2001), where a mention is resolved to its closest possible antecedent, or a best-first decision (Ng & Cardie, 2002), where a mention is resolved to its most confident antecedent (based on the confidence value returned by step 1). Global variants attempt to consider all possible clustering possibilites by creating and searching a *Bell tree* (Luo et al., 2004), by learning the optimal search strategy itself (Daumé III & Marcu, 2005), by building a graph representation and applying graph clustering techniques (Nicolae & Nicolae, 2006), or by employing integer linear programming (Klenner, 2007; Denis & Baldridge, 2009). Since these methods base their global clustering step on a local pairwise model, some global information which could have guided step 2 is already lost. The twin-candidate model (Yang et al., 2008) replaces the pairwise model by learning preferences between two antecedent candidates in step 1 and applies tournament schemes instead of the clustering in step 2.

There is little work which deviates from this two-step scheme. Culotta et al. (2007) introduce a first-order probabilistic model which implements features over sets of mentions and thus operates directly on entities.

In this paper we describe a novel approach to coreference resolution which avoids the division into two steps and instead performs a global decision in one step. We represent a document as a hypergraph, where the vertices denote mentions and the edges denote relational features between mentions. Coreference resolution is performed globally in one step by partitioning the hypergraph into subhypergraphs so that all mentions in one subhypergraph refer to the same entity. Our model out-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 143–151,
Beijing, August 2010

performs two strong baselines, Soon et al. (2001) and Bengtson & Roth (2008).

Soon et al. (2001) developed an end-to-end coreference resolution system for the MUC data, i.e., a system which processes raw documents as input and produces annotated ones as output. However, with the advent of the ACE data, many systems either evaluated only true mentions, i.e. mentions which are included in the annotation, the so-called key, or even received true information for mention boundaries, heads of mentions and mention type (Culotta et al., 2007, inter alia). While these papers report impressive results it has been concluded that this experimental setup simplifies the task and leads to an unrealistic surrogate for the coreference resolution problem (Stoyanov et al., 2009, p.657, p660). We argue that the field should move towards a realistic setting using system mentions, i.e. automatically determined mention boundaries and types. In this paper we report results using our end-to-end coreference resolution system, COPA, without relying on unrealistic assumptions.

2 Related Work

Soon et al. (2001) transform the coreference resolution problem straightforwardly into a pairwise classification task making it accessible to standard machine learning classifiers. They use a set of twelve powerful features. Their system is based solely on information of the mention pair anaphor and antecedent. It does not take any information of other mentions into account. However, it turned out that it is difficult to improve upon their results just by applying a more sophisticated learning method and without improving the features. We use a reimplementation of their system as first baseline. Bengtson & Roth (2008) push this approach to the limit by devising a much more informative feature set. They report the best results to date on the ACE 2004 data using true mentions. We use their system combined with our preprocessing components as second baseline.

Luo et al. (2004) perform the clustering step within a Bell tree representation. Hence their system theoretically has access to all possible outcomes making it a potentially global system. However, the classification step is still based on a pairwise model. Also since the search space in the Bell tree is too large they have to apply search heuristics. Hence, their approach loses much of the power of a truly global approach.

Culotta et al. (2007) introduce a first-order probabilistic model which implements features over sets of mentions. They use four features for their first-order model. The first is an enumeration over *pairs* of noun phrases. The second is the output of a *pairwise* model. The third is the cluster size. The fourth counts mention type, number and gender in each cluster. Still, their model is based mostly on information about pairs of mentions. They assume true mentions as input. It is not clear whether the improvement in results translates to system mentions.

Nicolae & Nicolae (2006) describe a graph-based approach which superficially resembles our approach. However, they still implement a two step coreference resolution approach and apply the global graph-based model only to step 2. They report considerable improvements over state-of-the-art systems including Luo et al. (2004). However, since they not only change the clustering strategy but also the features for step 1, it is not clear whether the improvements are due to the graph-based clustering technique. We, instead, describe a graph-based approach which performs classification and clustering in one step. We compare our approach with two competitive systems using the same feature sets.

3 COPA: Coreference Partitioner

The COPA system consists of learning modules which learn hyperedge weights from the training data, and resolution modules which create a hypergraph representation for the testing data and perform partitioning to produce subhypergraphs, each of which represents an entity. An example analysis of a short document involving the two entities, BARACK OBAMA and NICOLAS SARKOZY illustrates how COPA works.

> [US President Barack Obama] came to Toronto today.
> [Obama] discussed the financial crisis with [President Sarkozy].
> [He] talked to him [him] about the recent downturn of the European markets.
> [Barack Obama] will leave Toronto tomorrow.

A hypergraph (Figure (1a)) is built for this document based on three features. Two hyperedges denote the feature *partial string match*, {*US President Barack Obama, Barack Obama, Obama*} and {*US President Barack Obama, President Sarkozy*}. One hyperedge denotes the feature *pronoun match*, {*he, him*}. Two hyperedges denote the feature *all speak*, {*Obama, he*} and {*President Sarkozy, him*}.

On this initial representation, a spectral clustering technique is applied to find two partitions which have the strongest within-cluster connections and the weakest between-clusters relations. The cut found is called *Normalized Cut*, which avoids trivial partitions frequently output by the min-cut algorithm. The two output subhypergraphs (Figure (1b)) correspond to two resolved entities shown on both sides of the bold dashed line. In real cases, recursive cutting is applied to all the subhypergraphs resulting from previous steps, until a stopping criterion is reached.

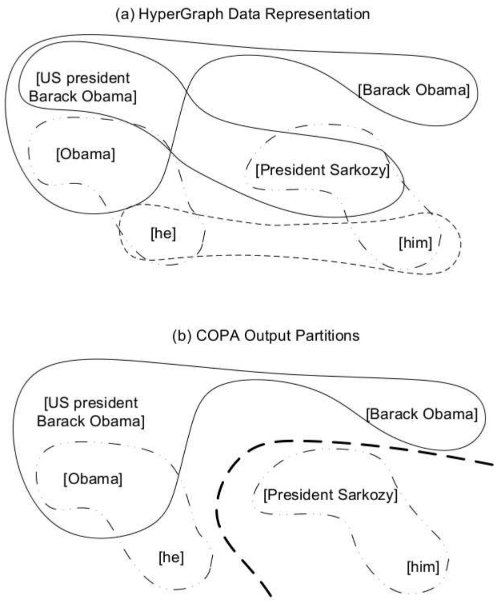

(a) HyperGraph Data Representation
[US president Barack Obama]
[Barack Obama]
[Obama]
[President Sarkozy]
[he]
[him]
(b) COPA Output Partitions
[US president Barack Obama]
[Barack Obama]
[Obama]
[President Sarkozy]
[he]
[him]

Figure 1: Hypergraph-based representation

3.1 HyperEdgeLearner

COPA needs training data only for computing the hyperedge weights. Hyperedges represent features. Each hyperedge corresponds to a feature instance modeling a simple relation between two or more mentions. This leads to initially overlapping sets of mentions. Hyperedges are assigned weights which are calculated based on the training data as the percentage of the initial edges (as illustrated in Figure (1a)) being in fact coreferent. The weights for some of Soon et al. (2001)'s features learned from the ACE 2004 training data are given in Table 1.

Edge Name	Weight
Alias	0.777
StrMatch_Pron	0.702
Appositive	0.568
StrMatch_Npron	0.657
ContinuousDistAgree	0.403

Table 1: Hyperedge weights for ACE 2004 data

3.2 Coreference Resolution Modules

Unlike pairwise models, COPA processes a document globally in one step, taking care of the preference information among all the mentions at the same time and clustering them into sets directly. A raw document is represented as a single hypergraph with multiple edges. The hypergraph resolver partitions the simple hypergraph into several subhypergraphs, each corresponding to one set of coreferent mentions (see e.g. Figure (1b) which contains two subhypergraphs).

3.2.1 HGModelBuilder

A single document is represented in a hypergraph with basic relational features. Each hyperedge in a graph corresponds to an instance of one of those features with the weight assigned by the *HyperEdgeLearner*. Instead of connecting nodes with the target relation as usually done in graph models, COPA builds the graph directly out of a set of low dimensional features without any assumptions for a distance metric.

3.2.2 HGResolver

In order to partition the hypergraph we adopt a spectral clustering algorithm. Spectral clustering techniques use information obtained from the eigenvalues and eigenvectors of the graph Laplacian to cluster the vertices. They are simple to implement and reasonably fast and have been shown to frequently outperform traditional clustering algorithms such as k-means. These techniques have

Algorithm 1 R2 partitioner

```
Note: { $L = I - {D_v}^{-\frac{1}{2}} H W {D_e}^{-1} H^T {D_v}^{-\frac{1}{2}}$ }
Note: { $Ncut(S) := vol\partial S(\frac{1}{volS} + \frac{1}{volS^c})$}
input: target hypergraph $HG$, predefined $\alpha^\star$
Given a $HG$, construct its $D_v$, $H$, $W$ and $D_e$
Compute $L$ for $HG$
Solve the $L$ for the second smallest eigenvector $V_2$
for each splitting point in $V_2$ do
    calculate $Ncut_i$
end for
Choose the splitting point with $\underset{i}{min}(Ncut_i)$
Generate two sub$HG$s
if $\underset{i}{min}(Ncut_i) < \alpha^*$ then
    for each sub$HG$ do
        Bi-partition the sub$HG$ with the R2 partitioner
    end for
else
    Output the current sub$HG$
end if
output: partitioned $HG$
```

many applications, e.g. image segmentation (Shi & Malik, 2000).

We adopt two variants of spectral clustering, *recursive 2-way partitioning (R2 partitioner)* and *flat-K partitioning*. Since flat-K partitioning did not perform as well we focus here on recursive 2-way partitioning. In contrast to flat-K partitioning, this method does not need any information about the number of target sets. Instead a stopping criterion $\alpha^\star$ has to be provided. $\alpha^\star$ is adjusted on development data (see Algorithm 1).

In order to apply spectral clustering to hypergraphs we follow Agarwal et al. (2005). All experimental results are obtained using symmetric Laplacians (L_{sym}) (von Luxburg, 2007).

Given a hypergraph *HG*, a set of matrices is generated. D_v and D_e denote the diagonal matrices containing the vertex and hyperedge degrees respectively. $|V| \times |E|$ matrix H represents the *HG* with the entries $h(v, e) = 1$ if $v \in e$ and 0 otherwise. H^T is the transpose of H. W is the diagonal matrix with the edge weights. S is one of the subhypergraphs generated from a cut in the *HG*, where $Ncut(S)$ is the cut's value.

Using Normalized Cut does not generate singleton clusters, hence a heuristic singleton detection strategy is used in COPA. We apply a threshold β to each node in the graph. If a node's degree is below the threshold, the node will be removed.

3.3 Complexity of HGResolver

Since edge weights are assigned using simple descriptive statistics, the time HGResolver needs for building the graph Laplacian matrix is insubstantial. For eigensolving, we use an open source library provided by the Colt project[1]which implements a Householder-QL algorithm to solve the eigenvalue decomposition. When applied to the symmetric graph Laplacian, the complexity of the eigensolving is given by $O(n^3)$, where n is the number of mentions in a hypergraph. Since there are only a few hundred mentions per document in our data, this complexity is not an issue (spectral clustering gets problematic when applied to millions of data points).

4 Features

The *HGModelBuilder* allows hyperedges with a degree higher than two to grow throughout the building process. This type of edge is *mergeable*. Edges with a degree of two describe pairwise relations. Thus these edges are *non-mergeable*. This way any kind of relational features can be incorporated into the hypergraph model.

Features are represented as types of hyperedges (in Figure (1b) the two hyperedges marked by "–··" are of the same type). Any realized edge is an instance of the corresponding edge type. All instances derived from the same type have the same weight, but they may get reweighted by the distance feature (Section 4.4).

In the following Subsections we describe the features used in our experiments. We use the entire set for obtaining the final results. We restrict ourselves to Soon et al. (2001)'s features when we compare our system with theirs in order to assess the impact of our model regardless of features (we use features 1., 2., 3., 6., 7., 11., 13.).

4.1 Hyperedges With a Degree > 2

High degree edges are the particular property of the hypergraph which allows to include all types of relational features into our model. The edges are built through pairwise relations and, if consistent, get incrementally merged into larger edges.

[1] `http://acs.lbl.gov/~hoschek/colt/`

High degree edges are not sensitive to positional information from the documents.

(1) StrMatch_Npron & (2) StrMatch_Pron: After discarding stop words, if the strings of mentions completely match and are not pronouns, they are put into edges of the *StrMatch_Npron* type. When the matched mentions are pronouns, they are put into the *StrMatch_Pron* type edges.

(3) Alias: After discarding stop words, if mentions are aliases of each other (i.e. proper names with partial match, full names and acronyms of organizations, etc.), they are put into edges of the *Alias* type.

(4) Synonym: If, according to WordNet, mentions are synonymous, they are put into an edge of the *Synonym* type.

(5) AllSpeak: Mentions which appear within a window of two words of a verb meaning *to say* form an edge of the *AllSpeak* type.

(6) Agreement: If mentions agree in *Gender*, *Number* and *Semantic Class* they are put in edges of the *Agreement* type. Because *Gender*, *Number* and *Semantic Class* are strong negative coreference indicators – in contrast to e.g. *StrMatch* – and hence weak positive features, they are combined into the one feature *Agreement*.

4.2 Hyperedges With a Degree = 2

Features which have been used by pairwise models are easily integrated into the hypergraph model by generating edges with only two vertices. Information sensitive to relative distance is represented by pairwise edges.

(7) Apposition & (8) RelativePronoun: If two mentions are in a appositive structure, they are put in an edge of type *Apposition*. If the latter mention is a relative pronoun, the mentions are put in an edge of type *RelativePronoun*.

(9) HeadModMatch: If the syntactic heads of two mentions match, and if their modifiers do not contradict each other, the mentions are put in an edge of type *HeadModMatch*.

(10) SubString: If a mention is the substring of another one, they are put into an edge of type *SubString*.

4.3 MentionType and EntityType

In our model **(11) mention type** can only reasonably be used when it is conjoined with other features, since mention type itself describes an attribute of single mentions. In COPA, it is conjoined with other features to form hyperedges, e.g. the *StrMatch_Pron* edge. We use the same strategy to represent **(12) entity type**.

4.4 Distance Weights

Our hypergraph model does not have any obvious means to encode distance information. However, the distance between two mentions plays an important role in coreference resolution, especially for resolving pronouns. We do not encode distance as feature, because this would introduce many two-degree-hyperedges which would be computationally very expensive without much gain in performance. Instead, we use distance to reweight two-degree-hyperedges, which are sensitive to positional information.

We experimented with two types of distance weights: One is **(13) sentence distance** as used in Soon et al. (2001)'s feature set, while the other is **(14) compatible mentions distance** as introduced by Bengtson & Roth (2008).

5 Experiments

We compare COPA's performance with two implementations of pairwise models. The first baseline is the BART (Versley et al., 2008) reimplementation of Soon et al. (2001), with few but effective features. Our second baseline is Bengtson & Roth (2008), which exploits a much larger feature set while keeping the machine learning approach simple. Bengtson & Roth (2008) show that their system outperforms much more sophisticated machine learning approaches such as Culotta et al. (2007), who reported the best results on true mentions before Bengtson & Roth (2008). Hence, Bengtson & Roth (2008) seems to be a reasonable competitor for evaluating COPA.

In order to report realistic results, we neither assume true mentions as input nor do we evaluate only on true mentions. Instead, we use an inhouse mention tagger for automatically extracting mentions.

5.1 Data

We use the MUC6 data (Chinchor & Sundheim, 2003) with standard training/testing divisions (30/30) as well as the MUC7 data (Chinchor, 2001) (30/20). Since we do not have access to the official ACE testing data (only available to ACE participants), we follow Bengtson & Roth (2008) for dividing the ACE 2004 English training data (Mitchell et al., 2004) into training, development and testing partitions (268/76/107). We randomly split the 252 ACE 2003 training documents (Mitchell et al., 2003) using the same proportions into training, development and testing (151/38/63). The systems were tuned on development and run only once on testing data.

5.2 Mention Tagger

We implement a classification-based mention tagger, which tags each NP chunk as ACE mention or not, with neccessary post-processing for embeded mentions. For the ACE 2004 testing data, we cover 75.8% of the heads with 73.5% accuracy.

5.3 Evaluation Metrics

We evaluate COPA with three coreference resolution evaluation metrics: the B^3-algorithm (Bagga & Baldwin, 1998), the *CEAF*-algorithm (Luo, 2005), and, for the sake of completeness, the *MUC*-score (Vilain et al., 1995).

Since the *MUC*-score does not evaluate singleton entities, it only partially evaluates the performance for ACE data, which includes singleton entities in the keys. The B^3-algorithm (Bagga & Baldwin, 1998) addresses this problem of the *MUC*-score by conducting calculations based on mentions instead of coreference relations. However, another problematic issue emerges when system mentions have to be dealt with: B^3 assumes the mentions in the key and in the response to be identical, which is unlikely when a mention tagger is used to create system mentions. The *CEAF*-algorithm aligns entities in key and response by means of a similarity metric, which is motivated by B^3's shortcoming of using one entity multiple times (Luo, 2005). However, although *CEAF* theoretically does not require to have the same number of mentions in key and response, the algorithm still cannot be directly applied to end-to-end coreference resolution systems, because the similarity metric is influenced by the number of mentions in key and response.

Hence, both the B^3- and *CEAF*-algorithms have to be extended to deal with system mentions which are not in the key and true mentions not extracted by the system, so called *twinless mentions* (Stoyanov et al., 2009). Two variants of the B^3-algorithm are proposed by Stoyanov et al. (2009), B^3_{all} and B^3_0. B^3_{all} tries to assign intuitive precision and recall to the twinless system mentions and twinless key mentions, while keeping the size of the system mention set and the key mention set unchanged (which are different from each other). For twinless mentions, B^3_{all} discards twinless key mentions for precision and twinless system mentions for recall. Discarding parts of the key mentions, however, makes the fair comparison of precision values difficult. B^3_0 produces counter-intuitive precision by discarding all twinless system mentions. Although it penalizes the recall of all twinless key mentions, so that the F-scores are balanced, it is still too lenient (for further analyses see Cai & Strube (2010)).

We devise two variants of the B^3- and *CEAF*-algorithms, namely B^3_{sys} and *CEAF*$_{sys}$. For computing precision, the algorithms put all twinless true mentions into the response even if they were not extracted. All twinless system mentions which were deemed not coreferent are discarded. Only twinless system mentions which were mistakenly resolved are put into the key. Hence, the system is penalized for resolving mentions not found in the key. For recall the algorithms only consider mentions from the original key by discarding all the twinless system mentions and putting twinless true mentions into the response as singletons (algorithm details, simulations and comparison of different systems and metrics are provided in Cai & Strube (2010)). For *CEAF*$_{sys}$, ϕ_3 (Luo, 2005) is used. B^3_{sys} and *CEAF*$_{sys}$ report results for end-to-end coreference resolution systems adequately.

5.4 Baselines

We compare COPA's performance with two baselines: *SOON* – the BART (Versley et al., 2008) reimplementation of Soon et al. (2001) – and

		SOON			COPA with R2 partitioner				
		R	P	F	R	P	F	$\alpha^\star$	β
MUC	MUC6	59.4	67.9	63.4	62.8	66.4	**64.5**	0.08	0.03
	MUC7	52.3	67.1	58.8	55.2	66.1	**60.1**	0.05	0.01
	ACE 2003	56.7	75.8	64.9	60.8	75.1	**67.2**	0.07	0.03
	ACE 2004	50.4	67.4	57.7	54.1	67.3	**60.0**	0.05	0.04
B^3_{sys}	MUC6	53.1	78.9	63.5	56.4	76.3	**64.1**	0.08	0.03
	MUC7	49.8	80.0	61.4	53.3	76.1	**62.7**	0.05	0.01
	ACE 2003	66.9	87.7	75.9	71.5	83.3	**77.0**	0.07	0.03
	ACE 2004	64.7	85.7	73.8	67.3	83.4	**74.5**	0.07	0.03
CEAF$_{sys}$	MUC6	56.9	53.0	54.9	62.2	57.5	**59.8**	0.08	0.03
	MUC7	57.3	54.3	55.7	58.3	54.2	56.2	0.06	0.01
	ACE 2003	71.0	68.7	69.8	71.1	68.3	69.7	0.07	0.03
	ACE 2004	67.9	65.2	66.5	68.5	65.5	67.0	0.07	0.03

Table 3: *SOON* vs. COPA R2 (*SOON* features, system mentions, bold indicates significant improvement in F-score over *SOON* according to a paired-t test with $p < 0.05$)

	SOON			*B&R*		
	R	P	F	R	P	F
B^3_{sys}	64.7	85.7	73.8	66.3	85.8	74.8

Table 2: Baselines on ACE 2004

B&R – Bengtson & Roth (2008)[2]. All systems share BART's preprocessing components and our in-house ACE mention tagger.

In Table 2 we report the performance of *SOON* and *B&R* on the ACE 2004 testing data using the BART preprocessing components and our in-house ACE mention tagger. For evaluation we use B^3_{sys} only, since Bengtson & Roth (2008)'s system does not allow to easily integrate *CEAF*.

B&R considerably outperforms *SOON* (we cannot compute statistical significance, because we do not have access to results for single documents in *B&R*). The difference, however, is not as big as we expected. Bengtson & Roth (2008) reported very good results when using true mentions. For evaluating on system mentions, however, they were using a too lenient variant of B^3 (Stoyanov et al., 2009) which discards all twinless mentions. When replacing this with B^3_{sys} the difference between *SOON* and *B&R* shrinks.

5.5 Results

In both comparisons, COPA uses the same features as the corresponding baseline system.

5.5.1 COPA vs. *SOON*

In Table 3 we compare the *SOON*-baseline with COPA using the R2 partitioner (parameters $\alpha^\star$ and β optimized on development data). Even though COPA and *SOON* use the same features, COPA consistently outperforms *SOON* on all data sets using all evaluation metrics. With the exception of the MUC7, the ACE 2003 and the ACE 2004 data evaluated with *CEAF*$_{sys}$, all of COPA's improvements are statistically significant. When evaluated using *MUC* and B^3_{sys}, COPA with the R2 partitioner boosts recall in all datasets while losing in precision. This shows that global hypergraph partitioning models the coreference resolution task more adequately than Soon et al. (2001)'s local model – even when using the very same features.

5.5.2 COPA vs. *B&R*

In Table 4 we compare the *B&R* system (using our preprocessing components and mention tagger), and COPA with the R2 partitioner using *B&R* features. COPA does not use the learned features from *B&R*, as this would have implied to embed a pairwise coreference resolution system in COPA. We report results for ACE 2003 and ACE 2004. The parameters are optimized on the ACE 2004 data. COPA with the R2 partitioner outperforms *B&R* on both datasets (we cannot compute statistical significance, because we do not have access to results for single documents in *B&R*). Bengtson & Roth (2008) developed their system on ACE 2004 data and never exposed it to ACE 2003 data. We suspect that the relatively poor result of *B&R* on ACE 2003 data is caused by overfitting to ACE

[2] `http://l2r.cs.uiuc.edu/~cogcomp/asoftware.php?skey=FLBJCOREF`

		B&R			COPA with R2 partitioner		
		R	P	F	R	P	F
B^3_{sys}	ACE 2003	56.4	97.3	71.4	70.3	86.5	77.5
	ACE 2004	66.3	85.8	74.8	68.4	84.4	75.6

Table 4: *B&R* vs. COPA R2 (*B&R* features, system mentions)

2004. Again, COPA gains in recall and loses in precision. This shows that COPA is a highly competetive system as it outperforms Bengtson & Roth (2008)'s system which has been claimed to have the best performance on the ACE 2004 data.

5.5.3 Running Time

On a machine with 2 AMD Opteron CPUs and 8 GB RAM, COPA finishes preprocessing, training and partitioning the ACE 2004 dataset in 15 minutes, which is slightly faster than our duplicated *SOON* baseline.

6 Discussion and Outlook

Most previous attempts to solve the coreference resolution task globally have been hampered by employing a local pairwise model in the classification step (step 1) while only the clustering step realizes a global approach, e.g. Luo et al. (2004), Nicolae & Nicolae (2006), Klenner (2007), Denis & Baldridge (2009), lesser so Culotta et al. (2007). It has been also observed that improvements in performance on true mentions do not necessarily translate into performance improvements on system mentions (Ng, 2008).

In this paper we describe a coreference resolution system, COPA, which implements a global decision in one step via hypergraph partitioning. COPA looks at the whole graph at once which enables it to outperform two strong baselines (Soon et al., 2001; Bengtson & Roth, 2008). COPA's hypergraph-based strategy can be taken as a general preference model, where the preference for one mention depends on information on all other mentions.

We follow Stoyanov et al. (2009) and argue that evaluating the performance of coreference resolution systems on true mentions is unrealistic. Hence we integrate an ACE mention tagger into our system, tune the system towards the real task, and evaluate only using system mentions. While Ng (2008) could not show that superior models achieved superior results on system mentions, COPA was able to outperform Bengtson & Roth (2008)'s system which has been claimed to achieve the best performance on the ACE 2004 data (using true mentions, Bengtson & Roth (2008) did not report any comparison with other systems using system mentions).

An error analysis revealed that there were some cluster-level inconsistencies in the COPA output. Enforcing this consistency would require a global strategy to propagate constraints, so that constraints can be included in the hypergraph partitioning properly. We are currently exploring constrained clustering, a field which has been very active recently (Basu et al., 2009). Using constrained clustering methods may allow us to integrate negative information as constraints instead of combining several weak positive features to one which is still weak (e.g. our *Agreement* feature). For an application of constrained clustering to the related task of database record linkage, see Bhattacharya & Getoor (2009).

Graph models cannot deal well with positional information, such as distance between mentions or the sequential ordering of mentions in a document. We implemented distance as weights on hyperedges which resulted in decent performance. However, this is limited to pairwise relations and thus does not exploit the power of the high degree relations available in COPA. We expect further improvements, once we manage to include positional information directly.

Acknowledgements. This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author has been supported by a HITS PhD. scholarship. We would like to thank Byoung-Tak Zhang for bringing hypergraphs to our attention and Èva Mújdricza-Maydt for implementing the mention tagger. Finally we would like to thank our colleagues in the HITS NLP group for providing us with useful comments.

References

Agarwal, Sameer, Jonwoo Lim, Lihi Zelnik-Manor, Pietro Perona, David Kriegman & Serge Belongie (2005). Beyond pairwise clustering. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05),* Vol. 2, pp. 838–845.

Bagga, Amit & Breck Baldwin (1998). Algorithms for scoring coreference chains. In *Proceedings of the 1st International Conference on Language Resources and Evaluation,* Granada, Spain, 28–30 May 1998, pp. 563–566.

Basu, Sugato, Ian Davidson & Kiri L. Wagstaff (Eds.) (2009). *Constrained Clustering: Advances in Algorithms, Theory, and Applications.* Boca Raton, Flo.: CRC Press.

Bengtson, Eric & Dan Roth (2008). Understanding the value of features for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Waikiki, Honolulu, Hawaii, 25-27 October 2008, pp. 294–303.

Bhattacharya, Indrajit & Lise Getoor (2009). Collective relational clustering. In S. Basu, I. Davidson & K. Wagstaff (Eds.), *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, pp. 221–244. Boca Raton, Flo.: CRC Press.

Cai, Jie & Michael Strube (2010). Evaluation metrics for end-to-end coreference resolution systems. In *Proceedings of the SIGdial 2010 Conference: The 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue,* Tokyo, Japan, 24–25 September 2010. To appear.

Chinchor, Nancy (2001). *Message Understanding Conference (MUC) 7.* LDC2001T02, Philadelphia, Penn: Linguistic Data Consortium.

Chinchor, Nancy & Beth Sundheim (2003). *Message Understanding Conference (MUC) 6.* LDC2003T13, Philadelphia, Penn: Linguistic Data Consortium.

Culotta, Aron, Michael Wick & Andrew McCallum (2007). First-order probabilistic models for coreference resolution. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics,* Rochester, N.Y., 22–27 April 2007, pp. 81–88.

Daumé III, Hal & Daniel Marcu (2005). A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing,* Vancouver, B.C., Canada, 6–8 October 2005, pp. 97–104.

Denis, Pascal & Jason Baldridge (2009). Global joint models for coreference resolution and named entity classification. *Procesamiento del Lenguaje Natural*, 42:87–96.

Klenner, Manfred (2007). Enforcing consistency on coreference sets. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing,* Borovets, Bulgaria, 27–29 September 2007, pp. 323–328.

Luo, Xiaoqiang (2005). On coreference resolution performance metrics. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing,* Vancouver, B.C., Canada, 6–8 October 2005, pp. 25–32.

Luo, Xiaoqiang, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla & Salim Roukos (2004). A mention-synchronous coreference resolution algorithm based on the Bell Tree. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics,* Barcelona, Spain, 21–26 July 2004, pp. 136–143.

Mitchell, Alexis, Stephanie Strassel, Shudong Huang & Ramez Zakhary (2004). *ACE 2004 Multilingual Training Corpus*. LDC2005T09, Philadelphia, Penn.: Linguistic Data Consortium.

Mitchell, Alexis, Stephanie Strassel, Mark Przybocki, JK Davis, George Doddington, Ralph Grishman, Adam Meyers, Ada Brunstain, Lisa Ferro & Beth Sundheim (2003). *TIDES Extraction (ACE) 2003 Multilingual Training Data*. LDC2004T09, Philadelphia, Penn.: Linguistic Data Consortium.

Ng, Vincent (2008). Unsupervised models for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing,* Waikiki, Honolulu, Hawaii, 25-27 October 2008, pp. 640–649.

Ng, Vincent & Claire Cardie (2002). Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics,* Philadelphia, Penn., 7–12 July 2002, pp. 104–111.

Nicolae, Cristina & Gabriel Nicolae (2006). BestCut: A graph algorithm for coreference resolution. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing,* Sydney, Australia, 22–23 July 2006, pp. 275–283.

Rahman, Altaf & Vincent Ng (2009). Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing,* Singapore, 6-7 August 2009, pp. 968–977.

Shi, Jianbo & Jitendra Malik (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

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

Stoyanov, Veselin, Nathan Gilbert, Claire Cardie & Ellen Riloff (2009). Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing,* Singapore, 2–7 August 2009, pp. 656–664.

Versley, Yannick, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang & Alessandro Moschitti (2008). BART: A modular toolkit for coreference resolution. In *Companion Volume to the Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics,* Columbus, Ohio, 15–20 June 2008, pp. 9–12.

Vilain, Marc, John Burger, John Aberdeen, Dennis Connolly & Lynette Hirschman (1995). A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Message Understanding Conference (MUC-6),* pp. 45–52. San Mateo, Cal.: Morgan Kaufmann.

von Luxburg, Ulrike (2007). A tutorial on spectral clustering. *Statistics and Computing,* 17(4):395–416.

Yang, Xiaofeng, Jian Su & Chew Lim Tan (2008). A twin-candidate model for learning-based anaphora resolution. *Computational Linguistics*, 34(3):327–356.

Exploiting Background Knowledge for Relation Extraction

Yee Seng Chan and **Dan Roth**
University of Illinois at Urbana-Champaign
{chanys,danr}@illinois.edu

Abstract

Relation extraction is the task of recognizing semantic relations among entities. Given a particular sentence supervised approaches to Relation Extraction employed feature or kernel functions which usually have a single sentence in their scope. The overall aim of this paper is to propose methods for using knowledge and resources that are external to the target sentence, as a way to improve relation extraction. We demonstrate this by exploiting background knowledge such as relationships among the target relations, as well as by considering how target relations relate to some existing knowledge resources. Our methods are general and we suggest that some of them could be applied to other NLP tasks.

1 Introduction

Relation extraction (RE) is the task of detecting and characterizing semantic relations expressed between entities in text. For instance, given the sentence "Cone, a Kansas City native, was originally signed by the Royals and broke into the majors with the team.", one of the relations we might want to extract is the *employment* relation between the pair of entity mentions "Cone" and "Royals". RE is important for many NLP applications such as building an ontology of entities, biomedical information extraction, and question answering.

Prior work have employed diverse approaches towards resolving the task. One approach is to build supervised RE systems using sentences annotated with entity mentions and predefined target relations. When given a new sentence, the RE system has to detect and disambiguate the presence of any predefined relations that might exist between each of the mention pairs in the sentence. In building these systems, researchers used a wide variety of features (Kambhatla, 2004; Zhou et al., 2005; Jiang and Zhai, 2007). Some of the common features used to analyze the target sentence include the words appearing in the sentence, their part-of-speech (POS) tags, the syntactic parse of the sentence, and the dependency path between the pair of mentions. In a related line of work, researchers have also proposed various kernel functions based on different structured representations (e.g. dependency or syntactic tree parses) of the target sentences (Bunescu and Mooney, 2005; Zhou et al., 2007; Zelenko et al., 2003; Zhang et al., 2006). Additionally, researchers have tried to automatically extract examples for supervised learning from resources such as Wikipedia (Weld et al., 2008) and databases (Mintz et al., 2009), or attempted *open* information extraction (IE) (Banko et al., 2007) to extract all possible relations.

In this work, we focus on supervised RE. In prior work, the feature and kernel functions employed are usually restricted to being defined on the various representations (e.g. lexical or structural) of the target sentences. However, in recognizing relations, humans are not thus constrained and rely on an abundance of implicit *world knowledge* or *background information*. What quantifies as world or background knowledge is rarely explored in the RE literature and we do not attempt to provide complete nor precise definitions in this paper. However, we show that by considering the relationship between our relations of interest, as

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 152–160,
Beijing, August 2010

well as how they relate to some existing knowledge resources, we improve the performance of RE. Specifically, the contributions of this paper are the following:

- When our relations of interest are clustered or organized in a hierarchical ontology, we show how to use this information to improve performance. By defining appropriate constraints between the predictions of relations at different levels of the hierarchy, we obtain globally coherent predictions as well as improved performance.

- Coreference is a generic relationship that might exists among entity mentions and we show how to exploit this information by assuming that co-referring mentions have no other interesting relations. We capture this intuition by using coreference information to constraint the predictions of a RE system.

- When characterizing the relationship between a pair of mentions, one can use a large encyclopedia such as Wikipedia to infer more knowledge about the two mentions. In this work, after probabilistically mapping mentions to their respective Wikipedia pages, we check whether the mentions are related. Another generic relationship that might exists between a pair of mentions is whether they have a *parent-child* relation and we use this as additional information.

- The sparsity of features (especially lexical features) is a common problem for supervised systems. In this work, we show that one can make fruitful use of unlabeled data, by using word clusters automatically gathered from unlabeled texts as a way of generalizing the lexical features.

- We combine the various relational predictions and background knowledge through a global inference procedure, which we formalize via an Integer Linear Programming (ILP) framework as a constraint optimization problem (Roth and Yih, 2007). This allows us to easily incorporate various constraints that encode the background knowledge.

Roth and Yih (2004) develop a relation extraction approach that exploits constraints among entity types and the relations allowed among them. We extend this view significantly, within a similar computational framework, to exploit relations among target relations, background information and world knowledge, as a way to improve relation extraction and make globally coherent predictions.

In the rest of this paper, we first describe the features used in our basic RE system in Section 2. We then describe how we make use of background knowledge in Section 3. In Section 4, we show our experimental results and perform analysis in Section 5. In Section 6, we discuss related work, before concluding in Section 7.

2 Relation Extraction System

In this section, we describe the features used in our basic relation extraction (RE) system. Given a pair of mentions m_1 and m_2 occurring within the same sentence, the system predicts whether any of the predefined relation holds between the two mentions. Since relations are usually asymmetric in nature, hence in all of our experiments, unless otherwise stated, we distinguish between the argument ordering of the two mentions. For instance, we consider m_1*:emp-org:*m_2 and m_2*:emp-org:*m_1 to be distinct relation types.

Most of the features used in our system are based on the work in (Zhou et al., 2005). In this paper, we propose some new collocation features inspired by word sense disambiguation (WSD). We give an overview of the features in Table 1. Due to space limitations, we only describe the collocation features and refer the reader to (Zhou et al., 2005) for the rest of the features.

2.1 Collocation Features

Following (Zhou et al., 2005), we use a single word to represent the head word of a mention. Since single words might be ambiguous or polysemous, we incorporate local collocation features which were found to be very useful for WSD. Given the head word hw_m of a mention *m*, the collocation feature $C_{i,j}$ refers to the sequence of tokens in the immediate context of hw_m. The offsets *i* and *j* denote the position (relative to hw_m)

Category	Feature
Lexical	hw of m_1
	hw of m_2
	hw of m_1, m_2
	BOW in m_1
	BOW in m_2
	single word between m_1, m_2
	BOW in between m_1, m_2
	bigrams in between m_1, m_2
	first word in between m_1, m_2
	last word in between m_1, m_2
Collocations	$C_{-1,-1}$, $C_{+1,+1}$
	$C_{-2,-1}$, $C_{-1,+1}$, $C_{+1,+2}$
Structural	m_1-in-m_2
	m_2-in-m_1
	#mentions between m_1, m_2
	any word between m_1, m_2
M-lvl	M-lvl of m_1, m_2
and	m_1, m_2 E-maintype
E-type	m_1, m_2 E-subtype
	m_1, m_2 M-lvl and E-maintype
	m_1, m_2 M-lvl and E-subtype
	m_1, m_2 E-subtype and m_1-in-m_2
	m_1, m_2 E-subtype and m_2-in-m_1
Dependency	path between m_1, m_2
	bag-of dep labels between m_1, m_2
	hw of m_1 and dep-parent
	hw of m_2 and dep-parent

Table 1: Features in the basic RE system. The abbreviations are as follows. hw: head word, M-lvl: mention level, E-type: entity type, dep-parent: the word's parent in the dependency tree.

of the first and last token of the sequence respectively. For instance, $C_{-1,+1}$ denotes a sequence of three tokens, consisting of the single token on the immediate left of hw_m, the token hw_m itself, and the single token on the immediate right of hw_m. For each mention, we extract 5 features: $C_{-1,-1}$, $C_{+1,+1}$, $C_{-2,-1}$, $C_{-1,+1}$, and $C_{+1,+2}$.

3 Using Background Knowledge

Now we describe how we inject additional knowledge into our relation extraction system.

3.1 Hierarchy of Relations

When our relations of interest are arranged in a hierarchical structure, one should leverage this information to learn more accurate relation predictors. For instance, assume that our relations are arranged in a two-level hierarchy and we learn two classifiers, one for disambiguating between the first level *coarse-grained* relations, and another for disambiguating between the second level *fine-grained* relations.

Since there are a lot more fine-grained relation types than coarse-grained relation types, we propose using the coarse-grained predictions which should intuitively be more reliable, to improve the fine-grained predictions. We show how to achieve this through defining appropriate constraints between the coarse-grained and fine-grained relations, which can be enforced through the Constrained Conditional Models framework (aka ILP) (Roth and Yih, 2007; Chang et al., 2008). Due to space limitations, we refer interested readers to the papers for more information on the CCM framework.

By doing this, not only are the predictions of both classifiers coherent with each other (thus obtaining better predictions from both classifiers), but more importantly, we are effectively using the (more reliable) predictions of the coarse-grained classifier to constrain the predictions of the fine-grained classifier. To the best of our knowledge, this approach for RE is novel.

In this paper, we work on the NIST Automatic Content Extraction (ACE) 2004 corpus. ACE defines several coarse-grained relations such as *employment/membership*, *geo-political entity (GPE) affiliation*, etc. Each coarse-grained relation is further refined into several fine-grained relations[1] and each fine-grained relation has a unique parent coarse-grained relation. For instance, the fine-grained relations *employed as ordinary staff*, *employed as an executive*, etc. are children relations of *employment/membership*.

Let m_i and m_j denote a pair of mentions i and j drawn from a document containing N mentions. Let $R_{i,j}$ denote a relation between m_i and m_j, and let $\mathcal{R} = \{R_{i,j}\}$, where $1{\leq}i, j{\leq}N; i{\neq}j$ denote the set of relations in the document. Also, we denote the set of predefined coarse-grained relation types and fine-grained relation types as $\mathcal{L}_{Rc}$ and $\mathcal{L}_{Rf}$ respectively. Since there could possibly be no relation between a mention pair, we add the *null* label to $\mathcal{L}_{Rc}$ and $\mathcal{L}_{Rf}$, allowing our classifiers to predict *null* for $R_{i,j}$. Finally, for a fine-grained relation type rf, let $\mathcal{V}(rf)$ denote its parent coarse-grained relation type.

[1] With the exception of the *Discourse* coarse-grained relation.

We learn two classifiers, one for disambiguating between the coarse-grained relations and one for disambiguating between the fine-grained relations. Let θ_c and θ_f denote the feature weights learned for predicting coarse-grained and fine-grained relations respectively. Let $p_R(rc) = \log P_c(rc|m_i, m_j; \theta_c)$ be the log probability that relation R is predicted to be of coarse-grained relation type rc. Similarly, let $p_R(rf) = \log P_f(rf|m_i, m_j; \theta_f)$ be the log probability that relation R is predicted to be of fine-grained relation type rf. Let $x_{\langle R,rc\rangle}$ be a binary variable which takes on the value of 1 if relation R is labeled with the coarse-grained label rc. Similarly, let $y_{\langle R,rf\rangle}$ be a binary variable which takes on the value of 1 if relation R is labeled with the fine-grained label rf. Our objective function is then:

$$\max \sum_{R\in\mathcal{R}} \sum_{rc\in\mathcal{L}_{Rc}} p_R(rc) \cdot x_{\langle R,rc\rangle} + \sum_{R\in\mathcal{R}} \sum_{rf\in\mathcal{L}_{Rf}} p_R(rf) \cdot y_{\langle R,rf\rangle} \quad (1)$$

subject to the following constraints:

$$\sum_{rc\in\mathcal{L}_{Rc}} x_{\langle R,rc\rangle} = 1 \quad \forall R \in \mathcal{R} \quad (2)$$

$$\sum_{rf\in\mathcal{L}_{Rf}} y_{\langle R,rf\rangle} = 1 \quad \forall R \in \mathcal{R} \quad (3)$$

$$x_{\langle R,rc\rangle} \in \{0,1\} \quad \forall R \in \mathcal{R}, rc \in \mathcal{L}_{Rc} \quad (4)$$

$$y_{\langle R,rf\rangle} \in \{0,1\} \quad \forall R \in \mathcal{R}, rf \in \mathcal{L}_{Rf} \quad (5)$$

Equations (2) and (3) require that each relation can only be assigned one coarse-grained label and one fine-grained label. Equations (4) and (5) indicate that $x_{\langle R,rc\rangle}$ and $y_{\langle R,rf\rangle}$ are binary variables. Two more constraints follow:

$$x_{\langle R,rc\rangle} \leq \sum_{\{rf\in\mathcal{L}_{Rf}|\mathcal{V}(rf)=rc\}} y_{\langle R,rf\rangle} \quad \forall R \in \mathcal{R}\ , rc \in \mathcal{L}_{Rc} \quad (6)$$

$$y_{\langle R,rf\rangle} \leq x_{\langle R,\mathcal{V}(rf)\rangle} \quad \forall R \in \mathcal{R}, rf \in \mathcal{L}_{Rf} \quad (7)$$

The logical form of Equation (6) can be written as: $x_{\langle R,rc\rangle} \Rightarrow y_{\langle R,rf_1\rangle} \vee y_{\langle R,rf_2\rangle} \cdots \vee y_{\langle R,rf_n\rangle}$, where $rf_1, rf_2, \ldots, rf_n$ are (child) fine-grained relations of the coarse-grained relation rc. This states that if we assign rc to relation R, then we must also assign to R a fine-grained relation rf which is a *child* of rc. The logical form of Equation (7) can be written as: $y_{\langle R,rf\rangle} \Rightarrow x_{\langle R,\mathcal{V}(rf)\rangle}$. This captures the inverse relation and states that if we assign rf to R, then we must also assign to R the relation type $\mathcal{V}(rf)$, which is the *parent* of rf. Together, Equations (6) and (7) constrain the predictions of the coarse-grained and fine-grained classifiers to be coherent with each other. Finally, we note that one could automatically translate logical constraints into linear inequalities (Chang et al., 2008).

art:	E_i ∈{gpe, org, per}, E_j ∈{fac, gpe, veh, wea}
emp-org:	E_i ∈{gpe, org, per}, E_j ∈{gpe, org, per}
gpe-aff:	E_i ∈{gpe, org, per}, E_j ∈{gpe, loc}
other-aff:	E_i ∈{gpe, org, per}, E_j ∈{gpe, loc}
per-soc:	E_i ∈{per}, E_j ∈{per}

Table 2: Entity type constraints.

This method is general and is applicable to other NLP tasks where a hierarchy exists, such as WSD and question answering. For instance, in WSD, one can predict coarse-grained and fine-grained senses using suitably defined sense inventories and then perform inference via ILP to obtain coherent predictions.

3.2 Entity Type Constraints

Each mention in ACE-2004 is annotated with one of seven coarse-grained entity types: person (per), organization (org), location (loc), geo-political entity (gpe), facility (fac), vehicle (veh), and weapon (wea).

Roth and Yih (2007) had shown that entity type information is useful for constraining the possible labels that a relation R can assume. For instance, both mentions involved in a *personal/social* relation must be of entity type *per*. In this work, we gather such information from the ACE-2004 documentation and inject it as constraints (on the coarse-grained relations) into our system. Due to space limitations, we do not state the constraint equations or objective function here, but we list the entity type constraints we imposed for each coarse-grained relation m_i-R-m_j in Table

2[2], where E_i (E_j) denotes the allowed set of entity types for mention m_i (m_j). Applying the entity type information improves the predictions of the coarse-grained classifier and this in turn could improve the predictions of the fine-grained classifier.

3.3 Using Coreference Information

We can also utilize the coreference relations among entity mentions. Assuming that we know mentions m_i and m_j are coreferent with each other, then there should be no relation between them[3]. Let $z_{\langle i,j \rangle}$ be a binary variable which takes on the value of 1 if mentions m_i and m_j are coreferent, and 0 if they are not. When $z_{\langle i,j \rangle}$=1, we capture the above intuition with the following constraints:

$$z_{\langle i,j \rangle} \leq x_{\langle R_{i,j}, null \rangle} \tag{8}$$

$$z_{\langle i,j \rangle} \leq y_{\langle R_{i,j}, null \rangle} \tag{9}$$

which can be written in logical form as: $z_{\langle i,j \rangle} \Rightarrow x_{\langle R_{i,j}, null \rangle}$, and $z_{\langle i,j \rangle} \Rightarrow y_{\langle R_{i,j}, null \rangle}$. We add the following to our objective function in Equation (1):

$$\sum_{m_i, m_j \in \mathbf{m}^2} co_{\langle i,j \rangle} \cdot z_{\langle i,j \rangle} + \bar{co}_{\langle i,j \rangle} \cdot (1 - z_{\langle i,j \rangle}) \tag{10}$$

where **m** is the set of mentions in a document, $co_{\langle i,j \rangle}$ and $\bar{co}_{\langle i,j \rangle}$ are the log probabilities of predicting that m_i and m_j are coreferent and not coreferent respectively. In this work, we assume we are given coreference information, which is available from the ACE annotation.

3.4 Using Knowledge from Wikipedia

We propose two ways of using Wikipedia to gather features for relation extraction. Wikipedia is a huge online encyclopedia and mainly contains articles describing entities or concepts.

The first intuition is that if we are able to correctly map a pair of mentions m_i and m_j to their corresponding Wikipedia article (assuming they are represented in Wikipedia), we could use the content on their Wikipedia pages to check whether they are related.

In this work, we use a *Wiki* system (Ratinov et al., 2010) which performs context-sensitive mapping of mentions to Wikipedia pages. In their work, the authors first identify phrases or mentions that could be mapped. The correct Wikipedia article for each mention is then probabilistically predicted using a combination of features based on Wikipedia hyperlink structure, semantic coherence, etc. The authors' own evaluation results indicate that the performance of their system ranges from 70–80%. When given a pair of mentions and the system returns the Wikipedia page for either one of the mentions, we introduce a feature:

$$w_1(m_i, m_j) = \begin{cases} 1, & \text{if } A_{m_i}(m_j) \\ & \text{or } A_{m_j}(m_i) \\ 0, & \text{otherwise} \end{cases}$$

where $A_{m_i}(m_j)$ returns true if the head extent of m_j is found (via simple string matching) in the predicted Wikipedia article of m_i. The interpretation of $A_{m_j}(m_i)$ is similar. We introduce a new feature into the RE system by combining $w_1(m_i, m_j)$ with m_i, m_j E-maintype (defined as in Table 1).

The second feature based on Wikipedia is as follows. It will be useful to check whether there is any *parent-child* relationship between two mentions. Intuitively, this will be useful for recognizing several relations such as *physical part-whole* (e.g. a city is part of a state), *subsidiary* (a company is a child-company of another), *citizenship* (a person is a citizen of a country), etc.

Given a pair of mentions m_i and m_j, we use a *Parent-Child* system (Do and Roth, 2010) to predict whether they have a *parent-child* relation. To achieve this, the system first gathers all Wikipedia articles that are related to m_i and m_j. It then uses the words in these pages and the category ontology of Wikipedia to make its *parent-child* predictions, while respecting certain defined constraints. In this work, we use its prediction as follows:

$$w_2(m_i, m_j) = \begin{cases} 1, & \text{if } \textit{parent-child}(m_i, m_j) \\ 0, & \text{otherwise} \end{cases}$$

[2] We do not impose entity type constraints on the coarse-grained relations *disc* and *phys*.

[3] In this work, we assume that no relations are reflexive. After the experiments in this paper are performed, we verified that in the ACE corpus we used, less than 1% of the relations are reflexive.

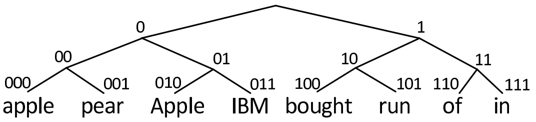


Figure 1: An example of Brown word cluster hierarchy from (Koo et al., 2008).

where we combine $w_2(m_i, m_j)$ with m_i, m_j E-maintype, introducing this as a new feature into our RE system.

3.5 Using Word Clusters

An inherent problem faced by supervised systems is that of data sparseness. To mitigate such issues in the lexical features, we use word clusters which are automatically generated from unlabeled texts. In this work, we use the Brown clustering algorithm (Brown et al., 1992), which has been shown to improve performance in various NLP applications such as dependency parsing (Koo et al., 2008), named entity recognition (Ratinov and Roth, 2009), and relation extraction (Boschee et al., 2005). The algorithm performs a hierarchical clustering of the words and represents them as a binary tree.

Each word is uniquely identified by its path from the root and every path is represented with a bit string. Figure 1 shows an example clustering where the maximum path length is 3. By using path prefixes of different lengths, one can obtain clusterings at different granularity. For instance, using prefixes of length 2 will put *apple* and *pear* into the same cluster, *Apple* and *IBM* into the same cluster, etc. In our work, we use clusters generated from New York Times text and simply use a path prefix of length 10. When Brown clusters are used in our system, all lexical features consisting of single words will be duplicated. For instance, for the feature *hw of m1*, one new feature which is the length-10 bit string path representing the original lexical head word of m1, will be introduced and presented to the classifier as a string feature.

4 Experiments

We used the ACE-2004 dataset (catalog LDC2005T09 from the Linguistic Data Consortium) to conduct our experiments. ACE-2004 defines 7 coarse-grained relations and 23 fine-grained relations. In all of our experiments, unless otherwise stated, we explicitly model the argument order (of the mentions) when asked to disambiguate the relation between a pair of mentions. Hence, we built our coarse-grained classifier with 15 relation labels to disambiguate between (two for each coarse-grained relation type and a *null* label when the two mentions are not related). Likewise, our fine-grained classifier has to disambiguate between 47 relation labels. In the dataset, relations do not cross sentence boundaries.

For our experiments, we trained regularized averaged perceptrons (Freund and Schapire, 1999), implemented within the Sparse Network of Winnow framework (Carlson et al., 1999), one for predicting the coarse-grained relations and another for predicting the fine-grained relations. Since the dataset has no split of training, development, and test sets, we followed prior work (Jiang and Zhai, 2007) and performed 5-fold cross validation to obtain our performance results. For simplicity, we used 5 rounds of training and a regularization parameter of 1.5 for the perceptrons in all our experiments. Finally, we concentrate on the evaluation of fine-grained relations.

4.1 Performance of the Basic RE system

As a gauge on the performance of our basic relation extraction system BasicRE using only the features described in Section 2, we compare against the state-of-the-art feature-based RE system of Jiang and Zhai (2007). However, we note that in that work, the authors performed their evaluation using *undirected* coarse-grained relations. That is, they do not distinguish on argument order of mentions and the classifier has to decide among 8 relation labels (7 coarse-grained relation types and a *null* label). Performing 5-fold cross validation on the news wire (nwire) and broadcast news (bnews) corpora in the ACE-2004 dataset, they reported a F-measure of 71.5 using a maximum entropy classifier[4]. Evaluating BasicRE on the same setting,

[4] After they heuristically performed feature selection and applied the heuristics giving the best evaluation performance, they obtained a result of 72.9.

	All nwire			10% of nwire		
Features	Rec%	Pre%	F1%	Rec%	Pre%	F1%
BasicRE	49.9	51.0	50.5	33.2	29.0	31.0
+Hier	+1.3	+1.3	+1.3	+1.1	+1.2	+1.1
+Hier+relEntC	+1.5	+2.0	+1.8	+3.3	+3.5	+3.4
+Coref	∼	+1.4	+0.7	−0.1	+1.0	+0.5
+Wiki	+0.2	+1.9	+1.0	+1.5	+2.5	+2.0
+Cluster	−0.2	+3.2	+1.4	−0.7	+3.9	+1.7
+ALL	+1.5	+6.7	+3.9	+4.7	+10.2	+7.6

Table 3: BasicRE gives the performance of our basic RE system on predicting fine-grained relations, obtained by performing 5-fold cross validation on only the news wire corpus of ACE-2004. Each subsequent row +Hier, +Hier+relEntC, +Coref, +Wiki, and +Cluster gives the *individual* contribution from using each knowledge. The bottom row +ALL gives the performance improvements from adding +Hier+relEntC+Coref+Wiki+Cluster. ∼ indicates no change in score.

we obtained a competitive F-measure of 71.2[5].

4.2 Experimental Settings for Evaluating Fine-grained Relations

Two of our knowledge sources, the Wiki system described in Section 3.4 and the word clusters described in Section 3.5, assume inputs of mixed-cased text. We note that the bnews corpus of ACE-2004 is entirely in lower-cased text. Hence, we use only the nwire corpus for our experiments here, from which we gathered 28,943 relation instances and 2,226 of those have a valid (non-null) relation[6].

We also propose the following experimental setting. First, since we made use of coreference information, we made sure that while performing our experiments, all instances from the same document are either all used as training data or all used as test data. Prior work in RE had not ensured this, but we argue that this provides a more realistic setting. Our own experiments indicate that this results in a 1-2% lower performance on fine-grained relations.

Secondly, prior work calculate their performance on relation extraction at the level of *mentions*. That is, each mention pair extracted is scored individually. An issue with this way of scoring on the ACE corpus is that ACE annotators rarely duplicate a relation link for coreferent mentions. For instance, assume that mentions m_i, m_j, and m_k exist in a given sentence, mentions m_i and m_j are coreferent, and the annotator establishes a particular relation type r between m_j and m_k. The annotator will not usually duplicate the same relation r between m_i and m_k and thus the label between these two mentions is then *null*. We are not suggesting that this is an incorrect approach, but clearly there is an issue since an important goal of performing RE is to populate or build an ontology of entities and establish the relations existing among the entities. Thus, we evaluate our performance at the *entity-level*.[7] That is, given a pair of entities, we establish the set of relation types existing between them, based on their mention annotations. Then we calculate recall and precision based on these established relations. Of course, performing such an evaluation requires knowledge about the coreference relations and in this work, we assume we are given this information.

4.3 Knowledge-Enriched System

Evaluating our system BasicRE (trained only on the features described in Section 2) on the nwire corpus, we obtained a F1 score of 50.5, as shown in Table 3. Next, we exploited the relation hierarchy as in Section 3.1 and obtained an improvement of 1.3, as shown in the row +Hier. Next, we added the entity type constraints of Section

[5] Using 10 rounds of training and a regularization parameter of 2.5 improves the result to 72.2. In general, we found that more rounds of training and a higher regularization value benefits coarse-grained relation classification, but not fine-grained relation classification.

[6] The number of relation instances in the nwire and bnews corpora are about the same.

[7] Our experiments indicate that performing the usual evaluation on mentions gives similar performance figures and the trend in Table 3 stays the same.

3.2. Remember that these constraints are imposed on the coarse-grained relations. Thus, they would only affect the fine-grained relation predictions if we also exploit the relation hierarchy. In the table, we show that all the background knowledge helped to improve performance, providing a total improvement of 3.9 to our basic RE system. Though the focus of this work is on fine-grained relations, our approach also improves the performance of coarse-grained relation predictions. BasicRE obtains a F1 score of 65.3 on coarse-grained relations and exploiting background knowledge gives a total improvement of 2.9.

5 Analysis

We explore the situation where we have very little training data. We assume during each cross validation fold, we are given only 10% of the training data we originally had. Previously, when performing 5-fold cross validation on 2,226 valid relation instances, we had about 1780 as training instances in each fold. Now, we assume we are only given about 178 training instances in each fold. Under this condition, BasicRE gives a F1 score of 31.0 on fine-grained relations. Adding all the background knowledge gives an improvement of 7.6 and this represents an error reduction of 39% when measured against the performance difference of 50.5 (31.0) when we have 1780 training instances vs. 178 training instances. On the coarse-grained relations, BasicRE gives a F1 score of 51.1 and exploiting background knowledge gives a total improvement of 5.0.

We also tabulated the list of fine-grained relations that improved by more than 1 F1 score when we incorporated +Wiki, on the experiment using all of nwire data: *phys:near* (physically near), *other-aff:ideology* (ideology affiliation), *art:user-or-owner* (user or owner of artifact), *per-soc:business* (business relationship), *phys:part-whole* (physical part-whole), *emp-org:subsidiary* (organization subsidiary), and *gpe-aff:citizen-or-resident* (citizen or resident). Most of these intuitively seemed to be information one would find being mentioned in an encyclopedia.

6 Related Work

Few prior work has explored using background knowledge to improve relation extraction performance. Zhou et al. (2008) took advantage of the hierarchical ontology of relations by proposing methods customized for the perceptron learning algorithm and support vector machines. In contrast, we propose a generic way of using the relation hierarchy which at the same time, gives globally coherent predictions and allows for easy injection of knowledge as constraints. Recently, Jiang (2009) proposed using features which are common across all relations. Her method is complementary to our approach, as she does not consider information such as the relatedness between different relations. On using semantic resources, Zhou et al. (2005) gathered two gazettes, one containing country names and another containing words indicating personal relationships. In relating the tasks of RE and coreference resolution, Ji et al. (2005) used the output of a RE system to rescore coreference hypotheses. In our work, we reverse the setting and explore using coreference to improve RE.

7 Conclusion

In this paper, we proposed a broad range of methods to inject background knowledge into a relation extraction system. Some of these methods, such as exploiting the relation hierarchy, are general in nature and could be easily applied to other NLP tasks. To combine the various relation predictions and knowledge, we perform global inference within an ILP framework. Besides allowing for easy injection of knowledge as constraints, this ensures globally coherent models and predictions.

Acknowledgements This research was partly sponsored by Air Force Research Laboratory (AFRL) under prime contract no. FA8750-09-C-0181. We thank Ming-Wei Chang and James Clarke for discussions on this research.

References

Banko, Michele, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007.

Open information extraction from the web. In *Proceedings of IJCAI-07*, pages 2670–2676.

Boschee, Elizabeth, Ralph Weischedel, and Alex Zamanian. 2005. Automatic information extraction. In *Proceedings of the International Conference on Intelligence Analysis*.

Brown, Peter F., Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Bunescu, Razvan C. and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of HLT/EMNLP-05*, pages 724–731.

Carlson, Andrew J., Chad M. Cumby, Jeff L. Rosen, and Dan Roth. 1999. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department, May.

Chang, Ming-Wei, Lev Ratinov, Nicholas Rizzolo, and Dan Roth. 2008. Learning and inference with constraints. In *Proceedings of AAAI-08*, pages 1513–1518.

Do, Quang and Dan Roth. 2010. On-the-fly constraint-based taxonomic relation identification. Technical report, University of Illinois. http://L2R.cs.uiuc.edu/∼danr/Papers/DoRo10.pdf.

Freund, Yoav and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Ji, Heng, David Westbrook, and Ralph Grishman. 2005. Using semantic relations to refine coreference decisions. In *Proceedings of HLT/EMNLP-05*, pages 17–24.

Jiang, Jing and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *Proceedings of HLT-NAACL-07*, pages 113–120.

Jiang, Jing. 2009. Multi-task transfer learning for weakly-supervised relation extraction. In *Proceedings of ACL-IJCNLP-09*, pages 1012–1020.

Kambhatla, Nanda. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of ACL-04*, pages 178–181.

Koo, Terry, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08:HLT*, pages 595–603.

Mintz, Mike, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP-09*, pages 1003–1011.

Ratinov, Lev and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL-09*, pages 147–155.

Ratinov, Lev, Doug Downey, and Dan Roth. 2010. Wikification for information retrieval. Technical report, University of Illinois. http://L2R.cs.uiuc.edu/∼danr/Papers/RatinovDoRo10.pdf.

Roth, Dan and Wen Tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of CoNLL-04*, pages 1–8.

Roth, Dan and Wen Tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. In Getoor, Lise and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.

Weld, Daniel S., Raphael Hoffman, and Fei Wu. 2008. Using wikipedia to bootstrap open information extraction. *ACM SIGMOD Special Issue on Managing Information Extraction*, 37(4):62–68.

Zelenko, Dmitry, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.

Zhang, Min, Jie Zhang, Jian Su, and GuoDong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of COLING-ACL-06*, pages 825–832.

Zhou, Guodong, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of ACL-05*.

Zhou, GuoDong, Min Zhang, DongHong Ji, and QiaoMing Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of EMNLP-CoNLL-07*, pages 728–736.

Zhou, Guodong, Min Zhang, Dong-Hong Ji, and Qiaoming Zhu. 2008. Hierarchical learning strategy in semantic relation extraction. *Information Processing & Management*, 44(3):1008–1021.

Jointly Modeling WSD and SRL with Markov Logic

Wanxiang Che and Ting Liu
Research Center for Information Retrieval
MOE-Microsoft Key Laboratory of Natural Language Processing and Speech
School of Computer Science and Technology
{car, tliu}@ir.hit.edu.cn

Abstract

Semantic role labeling (SRL) and word sense disambiguation (WSD) are two fundamental tasks in natural language processing to find a sentence-level semantic representation. To date, they have mostly been modeled in isolation. However, this approach neglects logical constraints between them. We therefore exploit some pipeline systems which verify the automatic all word sense disambiguation could help the semantic role labeling and vice versa. We further propose a Markov logic model that jointly labels semantic roles and disambiguates all word senses. By evaluating our model on the OntoNotes 3.0 data, we show that this joint approach leads to a higher performance for word sense disambiguation and semantic role labeling than those pipeline approaches.

1 Introduction

Semantic role labeling (SRL) and word sense disambiguation (WSD) are two fundamental tasks in natural language processing to find a sentence-level semantic representation. Semantic role labeling aims at identifying the relations between predicates in a sentence and their associated arguments. Word sense disambiguation is the process of identifying the correct meaning, or sense of a word in a given context. For example, for the sentence in Figure 1, we can find out that the predicate token "hitting" at position 3 has sense "cause to move by striking" and the sense label is "hit.01". The argument headed by the token "cat" at position 1 with sense "feline mammal" (cat.01) is referring to the player (A0), and the argument headed by the token "ball" at position 5 with sense "round object that is hit in games" (ball.01) is referring to the game object (A1) being hit.

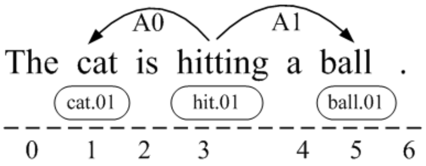


Figure 1: A sample of word sense disambiguation and semantic role labeling.

Normally, semantic role labeling and word sense disambiguation are regarded as two independent tasks, i.e., the word sense information is rarely used in a semantic role labeling system and vice versa. A few researchers have used semantic roles to help the verb sense disambiguation (Dang and Palmer, 2005). More people used predicate senses in semantic role labeling (Hajič et al., 2009; Surdeanu et al., 2008). However, both of the pipeline methods ignore possible dependencies between the word senses and semantic roles, and can result in the error propagation problem. The same problem also appears in other natural language processing tasks.

In order to make different natural language processing tasks be able to help each other, jointly modeling methods become popular recently, such as joint Chinese word segmentation and part-of-speech tagging (Kruengkrai et al., 2009; Zhang and Clark, 2008; Jiang et al., 2008), joint lemmatization and part-of-speech prediction (Toutanova and Cherry, 2009), joint morphological segmentation and syntactic parsing (Goldberg and Tsarfaty, 2008), joint text and aspect ratings for sentiment summarization (Titov and McDonald, 2008), and joint parsing and named entity recognition (Finkel and Manning, 2009). For semantic role labeling, Dahlmeier et al. (2009) proposed a method to maximize the joint probability of the seman-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 161–169,
Beijing, August 2010

tic role of preposition phrases and the preposition sense.

In order to do better joint learning, a novel statistical relational learning framework, Markov logic (Domingos and Lowd, 2009) was introduced to join semantic role labeling and predicate senses (Meza-Ruiz and Riedel, 2009). Markov logic combines the first order logic and Markov networks, to develop a joint probability model over all related rules. Global constraints (introduced by Punyakanok et al. (2008)) among semantic roles can be easily added into Markov logic. And the more important, the jointly modeling can be realized using Markov logic naturally.

Besides predicates and prepositions, other word senses are also important information for recognizing semantic roles. For example, if we know "cat" is an "agent" of the predicate "hit" in a sentence, we can guess that "dog" can also be an "agent" of "hit", though it does not appear in the training data. Similarly, the semantic role information can also help to disambiguate word senses. In addition, the predicate sense and the argument sense can also help each other. In the sentence "The cat is hitting a ball.", if we know "hit" here has a game related sense, we can guess that the "ball" should have the sense "is a round object in games". In the same way, the correct "ball" sense can help to disambiguate the sense of "hit". The joint probability, that they are disambiguated correctly simultaneously will be larger than other abnormalities.

The release of OntoNotes (Hovy et al., 2006) provides us an opportunity to jointly model all word senses disambiguation and semantic role labeling. OntoNotes is a large corpus annotated with constituency trees (based on Penn Treebank), predicate argument structures (based on Penn PropBank), all word senses, etc. It has been used in some natural language processing tasks, such as joint parsing and named entity recognition (Finkel and Manning, 2009), and word sense disambiguation (Zhong et al., 2008).

In this paper, we first propose some pipeline systems which exploit automatic all word sense disambiguation into semantic role labeling task and vice versa. Then we present a Markov logic model which can easily express useful global constraints and jointly disambiguate all word senses and label semantic roles.

Experiments on the OntoNotes 3.0 corpus show that (1) the automatic all word sense disambiguation and semantic role labeling tasks can help each other when using pipeline approaches, and more important, (2) the joint approach using Markov logic leads to higher accuracy for word sense disambiguation and performance (F_1) for semantic role labeling than pipeline approaches.

2 Related Work

Joint models were often used in semantic role labeling community. Toutanova et al. (2008) and Punyakanok et al. (2008) presented a re-ranking model and an integer linear programming model respectively to jointly learn a global optimal semantic roles assignment. Besides jointly learning semantic role assignment of different constituents for one task (semantic role labeling), their methods have been used to jointly learn for two tasks (semantic role labeling and syntactic parsing). However, it is easy for the re-ranking model to loss the optimal result, if it is not included in the top n results. In addition, the integer linear programming model can only use hard constraints. A lot of engineering work is also required in both models.

Recently, Markov logic (Domingos and Lowd, 2009) became a hot framework for joint model. It has been successfully used in temporal relations recognition (Yoshikawa et al., 2009), co-reference resolution (Poon and Domingos, 2008), etc. It is very easy to do joint modeling using Markov logic. The only work is to define relevant formulas. Meza-Ruiz and Riedel (2009) have joined semantic role labeling and predicate senses disambiguation with Markov logic.

The above idea, that the predicate senses and the semantic role labeling can help each other, may be inspired by Hajič et al. (2009), Surdeanu et al. (2008), and Dang and Palmer (2005). They have shown that semantic role features are helpful to disambiguate verb senses and vice versa.

Besides predicate senses, Dahlmeier et al. (2009) proposed a joint model to maximize probability of the preposition senses and the semantic role of prepositional phrases.

Except for predicate and preposition senses, Che et al. (2010) explored all word senses for semantic role labeling. They showed that all word senses can improve the semantic role labeling performance significantly. However, the golden word senses were used in their experiments. The results are still unknown when an automatic word sense disambiguation system is used.

In this paper, we not only use all word senses disambiguated by an automatic system, but also make the semantic role labeling results to help word sense disambiguation synchronously with a joint model.

3 Markov Logic

Markov logic can be understood as a knowledge representation with a weight attached to a first-order logic formula. Let us describe Markov logic in the case of the semantic role labeling task. We can model this task by first introducing a set of logical predicates such as $role(p, a, r)$ and $lemma(i, l)$, which means that the argument at position a has the role r with respect to the predicate at position p and token at position i has lemma l respectively. Then we specify a set of weighted first order formulas that define a distribution over sets of ground atoms of these predicates (or so-called possible worlds).

Ideally, the distribution we define with these weighted formulas assigns high probability to possible worlds where semantic role labeling is correct and a low probability to worlds where this is not the case. For instance, for the sentence in Figure 1, a suitable set of weighted formulas would assign a high probability to the world:

$$\begin{gathered} lemma(1, cat), lemma(3, hit), lemma(5, ball) \\ role(3, 1, A0), role(3, 5, A1) \end{gathered}$$

and low probabilities to other cases.

A Markov logic network (MLN) M is a set of weighted formulas, i.e., a set of pairs (ϕ, ω), where ϕ is a first order formula and ω is the real weight of the formula. M defines a probability distribution over possible worlds:

$$p(y) = \frac{1}{Z} \exp(\sum_{(\phi,\omega)\in M} \omega \sum_{c \in C^{\phi}} f_c^{\phi}(y))$$

where each c is a binding of free variables in ϕ to constants. Each f_c^{ϕ} is a binary feature function that returns 1 if the possible world y includes the ground formula by replacing the free variables in ϕ with the constants in c is true, and 0 otherwise. C^{ϕ} is the set of all bindings for the variables in ϕ. Z is a normalization constant.

4 Model

We divide our system into two stages: word sense disambiguation and semantic role labeling. For comparison, we can process them with pipeline strategy, i.e., the word sense disambiguation results are used in semantic role labeling or the semantic role labeling results are used in word sense disambiguation. Of course, we can jointly process them with Markov logic easily.

We define two hidden predicates for the two stages respectively. For word sense disambiguation, we define the predicate $sense(w, s)$ which indicates that the word at position w has the sense s. For semantic role labeling, the predicate $role(p, a, r)$ is defined as mentioned in above.

Different from Meza-Ruiz and Riedel (2009), which only used sense number as word sense representation, we use a triple (lemma, part-of-speech, sense num) to represent the word sense s. For example, (hit, v, 01) denotes that the verb "hit" has sense number 01. Obviously, our representation can distinguish different word senses which have the identical sense number. In addition, we use one argument classification stage with predicate $role$ to label semantic roles as Che et al. (2009). Similarly, no argument identification stage is used in our model. The approach can improve the recall of the system.

In addition to the hidden predicates, we define observable predicates to represent the information available in the corpus. Table 1 presents these predicates.

4.1 Local Formula

A local formula means that its groundings relate any number of observed ground atoms to exactly one hidden ground atom. For example

$$lemma(p, +l_1) \wedge lemma(a, +l_2) \Rightarrow role(p, a, +r)$$

Predicates	Description
$word(i, w)$	Token i has word w
$pos(i, t)$	Token i has part-of-speech t
$lemma(i, l)$	Token i has lemma l
$chdpos(i, t)$	The part-of-speech string of token i's all children is t
$chddep(i, d)$	The dependency relation string of token i's all children is t
$firstLemma(i, l)$	The leftmost lemma of a subtree rooted by token i is l
$lastLemma(i, l)$	The rightmost lemma of a subtree rooted by token i is l
$posFrame(i, fr)$	fr is a part-of-speech frame at token i
$dep(h, a, de)$	The dependency relation between an argument a and its head h is de
$isPredicate(p)$	Token p is a predicate
$posPath(p, a, pa)$	The part-of-speech path between a predicate p and an argument a is pa
$depPath(p, a, pa)$	The dependency relation path between a predicate p and an argument a is pa
$pathLen(p, a, le)$	The path length between a predicate p and an argument a is le
$position(p, a, po)$	The relative position between a predicate p and an argument a is po
$family(p, a, fa)$	The family relation between a predicate p and an argument a is fa
$wsdCand(i, t)$	Token i is a word sense disambiguation candidate, here t is "v" or "n"
$uniqe(r)$	For a predicate, semantic role r can only appear once

Table 1: Observable Predicates.

means that if the predicate lemma at position p is l_1 and the argument lemma at position a is l_2, then the semantic role between the predicate and the argument is r with some possibility.

The $+$ notation signifies that Markov logic generates a separate formula and a separate weight for each constant of the appropriate type, such as each possible pair of lemmas (l_1, l_2, r). This type of "template-based" formula generation can be performed automatically by a Markov logic engine, such as the *thebeast*[1] system.

The local formulas are based on features employed in the state-of-the-art systems. For word sense disambiguation, we use the basic features mentioned by Zhong et al. (2008). The semantic role labeling features are from Che et al. (2009), the best system of the CoNLL 2009 shared task. The final features are listed in Table 2.

Features	SRL	WSD
Lemma	●	●
POS	●	●
FirstwordLemma	●	
HeadwordLemma	●	
HeadwordPOS	●	
LastwordLemma	●	
POSPath	●	
PathLength	●	
Position	●	
PredicateLemma	●	
PredicatePOS	●	
RelationPath	●	
DepRelation	●	
POSUpPath	●	
POSFrame	●	
FamilyShip	●	
BagOfWords		●
Window3OrderedWords		●
Window3OrderedPOSs		●

Table 2: Local Features.

What follows are some simple examples in order to explain how we implement each feature as a formula (or a set of formulas).

Consider the "Position" feature. We first introduce a predicate $position(p, a, po)$ that denotes the relative position between predicate p and argument a is po. Then we add a formula

$$position(p, a, +po) \Rightarrow role(p, a, +r)$$

for all possible combinations of $position$ and $role$ relations.

The "BagOfWords" feature means that the sense of a word w is determined by all of lemmas in a sentence. Then, we add the following formula set:

$$wsdCand(w, +t_w) \wedge lemma(w, +l_w) \wedge lemma(1, +l_1) \Rightarrow sense(w, +s)$$
$$\ldots$$
$$wsdCand(w, +t_w) \wedge lemma(w, +l_w) \wedge lemma(2, +l_i) \Rightarrow sense(w, +s)$$
$$\ldots$$
$$wsdCand(w, +t_w) \wedge lemma(w, +l_w) \wedge lemma(n, +l_n) \Rightarrow sense(w, +s)$$

where, the w is the position of current word and t_w is its part-of-speech tag, l_w is its lemma. l_i is the lemma of token i. There are n tokens in a sentence totally.

4.2 Global Formula

Global formulas relate more than one hidden ground atoms. We use this type of formula for two purposes:

[1] http://code.google.com/p/thebeast/

1. To capture the global constraints among different semantic roles;

2. To reflect the joint relation between word sense disambiguation and semantic role labeling.

Punyakanok et al. (2008) proposed an integer linear programming (ILP) model to get the global optimization for semantic role labeling, which satisfies some constraints. This approach has been successfully transferred into dependency parse tree based semantic role labeling system by Che et al. (2009). The final results must satisfy two constraints which can be described with Markov logic formulas as follows:

C1: Each word should be labeled with one and only one label.

$$role(p, a, r_1) \wedge r_1 \neq r_2 \Rightarrow \neg role(p, a, r_2)$$

The same unique constraint also happens on the word sense disambiguation, i.e.,

$$sense(w, s_1) \wedge s_1 \neq s_2 \Rightarrow \neg sense(w, s_2)$$

C2: Some roles (A0∼A5) appear only once for a predicate.

$$role(p, a_1, r) \wedge uniqe(r) \wedge a_1 \neq a_2 \Rightarrow \neg role(p, a_2, r)$$

It is also easy to express the joint relation between word sense disambiguation and semantic role labeling with Markov logic. What we need to do is just adding some global formulas. The relation between them can be shown in Figure 2. Inspired by CoNLL 2008 (Surdeanu et al., 2008) and 2009 (Hajič et al., 2009) shared tasks, where most of successful participant systems used predicate senses for semantic role labeling, we also model that the word sense disambiguation implicates the semantic role labeling.

Here, we divide the all word sense disambiguation task into two subtasks: predicate sense disambiguation and argument sense disambiguation. The advantages of the division method approach lie in two aspects. First, it makes us distinguish the contributions of predicate and argument word sense disambiguation respectively. Second, as previous discussed, the predicate and argument sense disambiguation can help each other. Therefore, we can reflect the help with the division and use Markov logic to represent it.

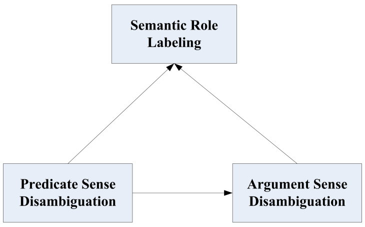


Figure 2: Global model between word sense disambiguation and semantic role labeling.

Finally, we use three global formulas to implement the three lines with direction in Figure 2. They are:

$$\begin{array}{rcl} sense(p, +s) & \Rightarrow & role(p, a, +r) \\ sense(a, +s) & \Rightarrow & role(p, a, +r) \\ sense(p, +s) & \Rightarrow & sense(a, +s) \end{array}$$

5 Experiments

5.1 Experimental Setting

In our experiments, we use the OntoNotes Release 3.0[2] corpus, the latest version of OntoNotes (Hovy et al., 2006). The OntoNotes project leaders describe it as “a large, multilingual richly-annotated corpus constructed at 90% internanotator agreement.” The corpus has been annotated with multiple levels of annotation, including constituency trees, predicate argument structure, word senses, co-reference, and named entities. For this work, we focus on the constituency trees, word senses, and predicate argument structures. The corpus has English, Chinese, and Arabic portions, and we just use the English portion, which has been split into four sections: broadcast conversation (bc), broadcast news (bn), magazine (mz), and newswire (nw). There are several datasets in each section, such as cnn and voa.

We will do our experiments on all of the OntoNotes 3.0 English datasets. For each dataset, we aimed for roughly a 60% train / 20% development / 20% test split. See Table 3 for the detailed statistics. Here, we use the human annotated part-of-speech and parse trees provided by OntoNotes. The lemma of each word is extracted using WordNet tool[3].

[2] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2009T24

[3] http://wordnet.princeton.edu/

		Training		Developing		Testing	
bc	cctv	1,042	(0000-0003)	328	(0004-0004)	333	(0005-0005)
	cnn	2,927	(0000-0004)	963	(0005-0006)	880	(0007-0008)
	msnbc	2,472	(0000-0003)	1,209	(0004-0005)	1,315	(0006-0007)
	phoenix	590	(0000-0001)	240	(0002-0002)	322	(0003-0003)
bn	abc	594	(0001-0040)	146	(0041-0054)	126	(0057-0069)
	cnn	1,610	(0001-0234)	835	(0235-0329)	1,068	(0330-0437)
	mnb	309	(0001-0015)	111	(0016-0020)	114	(0021-0025)
	nbc	281	(0001-0023)	128	(0024-0031)	78	(0032-0039)
	pri	1,104	(0001-0068)	399	(0069-0090)	366	(0091-0112)
	voa	1,159	(0001-0159)	315	(0160-0212)	315	(0213-0265)
mz	sinorama	5,051	(1001-1048)	1,262	(1049-1063)	1,456	(1064-1078)
nw	wsj	8,138	(0020-1446)	2,549	(1447-1705)	3,133	(1730-2454)
	xinhua	2,285	(0001-0195)	724	(0196-0260)	670	(0261-0325)
All		27,562		9,209		10,176	

Table 3: Training, developing and testing set sizes for the datasets in sentences. The file ranges (in parenthesis) refer to the numbers within the names of the original OntoNotes 3.0 files. Here, we remove 4,873 sentences without semantic role labeling annotation.

Because we used semantic role labeling system which is based on dependence syntactic trees, we convert the constituency trees into dependence trees with an Constituent-to-Dependency Conversion Tool[4].

The *thebeast* system is used in our experiment as Markov logic engine. It uses cutting planes inference technique (Riedel, 2008) with integer linear programming. The weights are learned with MIRA (Crammer and Singer, 2003) online learning algorithm.

To our knowledge, this is the first word sense disambiguation and semantic role labeling experiment on OntoNotes 3.0 corpus. In order to compare our joint model with previous work, we build several systems:

Baseline: There are two independent baseline systems: word sense disambiguation and semantic role labeling. In each of baseline systems, we only use the local formulas (Section 4.1) and the global formulas which only express the global constraints (Section 4.2).

Pipeline: In a pipeline system, we use additional features outputted by preceded stages. Such as in semantic role labeling pipeline system, we use word sense as features, i.e., we set $sense(w, s)$ as an observable predicate and add $sense(p, s) \Rightarrow role(p, a, r)$ and $sense(a, s) \Rightarrow role(p, a, r)$ formulas into semantic role labeling task. As for word sense disambiguation task, we add a set of formulas $role(p, a_i, r) \Rightarrow sense(p, s)$, where a_i is the i^{th} argument of the predicate at position p, and a formula $role(p, a, r) \Rightarrow sense(p, s)$ for the argument at position a respectively.

Jointly: We use all global formulas mentioned in Section 4.2. With Markov logic, we can add global constraints and get the word sense disambiguation and the semantic role labeling results simultaneously.

5.2 Results and Discussion

The performance of these systems on test set is shown in Table 4. All of the parameters are fine tuned on the development set.

Here, we only consider the noun and verb word sense disambiguation, which cover most of multisense words. Therefore, the word sense disambiguation performance means the accuracy of all nouns and verbs in the test set. The performance of semantic role labeling is calculated using the semantic evaluation metric of the CoNLL 2009 shared task scorer[5]. It measures the precision, recall, and F_1 score of the recovered semantic dependencies. The F_1 score is used as the final performance metric. A semantic dependency is created for each predicate and its arguments. The label of such dependency is the role of the argument. The same with the CoNLL 2009 shared task, we assume that the predicates have been identified

[4] http://nlp.cs.lth.se/software/treebank_converter/

[5] http://ufal.mff.cuni.cz/conll2009-st/eval09.pl

		WSD	SRL
Most Frequent Sense		85.58	—
Baseline		89.37	83.97
Pipeline	PS	89.53	84.17
	AS	89.41	83.94
	PS + AS	—	84.24
Jointly	PS ⇒ SRL	89.53	84.27
	AS ⇒ SRL	89.49	84.16
	PS ⇒ AS	89.45	—
	PS + AS ⇒ SRL	89.54	84.34
	Fully	89.55	84.36

Table 4: The results of different systems. Here, PS means predicate senses and AS means argument senses.

correctly.

The first row of Table 4 gives the word sense disambiguation result with the most frequent sense, i.e., the #01 sense of each candidate word which normally is the most frequent one in a balance corpus.

The second row shows the baseline performances. Here, we note that the 89.37 word sense disambiguation accuracy and the 83.97 semantic role labeling F_1 we obtained are comparable to the state-of-the-art systems, such as the 89.1 word sense disambiguation accuracy given by Zhong et al. (2008) and 85.48 semantic role labeling performance given by Che et al. (2010) on OntoNotes 2.0 respectively, although the corpus used in our experiments is upgraded version of theirs[6]. Additionally, the performance of word sense disambiguation is higher than that of the most frequent sense significantly (z-test[7] with $\rho < 0.01$). Therefore, the experimental results show that the Markov logic can achieve considerable performances for word sense disambiguation and semantic role labeling on the latest OntoNotes 3.0 corpus.

There are two kinds of pipeline systems: word sense disambiguation (WSD) based on semantic role labeling and semantic role labeling (SRL) based on word sense disambiguation. For the using method of word senses, we first only exploit predicate senses (PS) as mentioned by Surdeanu et al. (2008) and Hajič et al. (2009). Then, in order to examine the contribution of word senses except for predicates, we use argument senses (AS) in isolation. Finally, all word senses (PS + AS) were considered.

We can see that when the predicate senses (PS) are used to label semantic role, the performance of semantic role labeling can be improved from 83.97 to 84.17. The conclusion, that the predicate sense can improve semantic role labeling performance, is similar with CoNLL 2008 (Surdeanu et al., 2008) and 2009 (Hajič et al., 2009) shared tasks. However, the improvement is not significant (χ^2-test[8] with $\rho < 0.1$). Additionally, the semantic role labeling can improve the predicate sense disambiguation significantly from 89.37 to 89.53 (z-test with $\rho < 0.1$). The same conclusion was obtained by Dang and Palmer (2005).

However, when we only use argument senses (AS), both of the word sense disambiguation and semantic role labeling performances are almost unchanged (from 89.37 to 89.41 and from 83.97 to 83.94 respectively). For the semantic role labeling task, the reason is that the original lemma and part-of-speech features have been able to describe the argument related information. This kind of sense features is just reduplicate. On the other hand, the argument senses cannot be determined only by the semantic roles. For example, the semantic role “A1” cannot predict the argument sense of “ball” exactly. The predicates must be considered simultaneously.

Therefore, we use the last strategy (PS + AS), which combines the predicate sense and the argument sense together to predict semantic roles. The results show that the performance can be improved significantly (χ^2-test with $\rho < 0.05$) from 83.97 to 84.24. Accordingly, the experiment proves that automatic all word sense disambiguation can further improve the semantic role labeling performance. Different from Che et al. (2010), where the semantic role labeling can be improved with correct word senses about $F_1 = 1$, our improvement is much lower. The main reason is that the performance of our word sense disambiguation with the most basic features is not high enough. Another limitation of the pipeline strat-

[6] Compared with OntoNotes 2.0, the version 3.0 incorporates more corpus.

[7] http://www.dimensionresearch.com/resources/calculators/ztest.html

[8] http://graphpad.com/quickcalcs/chisquared1.cfm

egy is that it is difficult to predict the combination between predicate and argument senses. This is an obvious shortcoming of the pipeline method.

With Markov logic, we can easily join different tasks with global formulas. As shown in Table 4, we use five joint strategies:

1. PS ⇒ SRL: means that we jointly disambiguate predicate senses and label semantic roles. Compared with the pipeline PS system, word sense disambiguation performance is unchanged. However, the semantic role labeling performance is improved from 84.17 to 84.27. Compared with the baseline's 83.97, the improvement is significant (χ^2-test with $\rho < 0.05$).

2. AS ⇒ SRL: means that we jointly disambiguate argument senses and label semantic roles. Compared with the pipeline AS system, both of word sense disambiguation and semantic role labeling performances are improved (from 89.41 to 89.49 and from 83.94 to 84.16 respectively). Although, the improvement is not significant, it is observed that the joint model has the capacity to improve the performance, especially for semantic role labeling, if we could have a more accurate word sense disambiguation.

3. PS ⇒ AS: means that we jointly disambiguate predicate word senses and argument senses. This kind of joint model does not influence the performance of semantic role labeling. The word sense disambiguation outperforms the baseline system from 89.37 to 89.45. The result verifies our assumption that the predicate and argument senses can help each other.

4. PS + AS ⇒ SRL: means that we jointly disambiguate all word senses and label semantic roles. Compared with the pipeline method which uses the PS + AS strategy, the joint method can further improve the semantic role labeling (from 84.24 to 84.34). Additionally, it can obtain the predicate and argument senses together. The all word sense disambiguation performance (89.54) is higher than the baseline (89.37) significantly (z-test with $\rho < 0.1$).

5. Fully: finally, we use all of the three global formulas together, i.e., we jointly disambiguate predicate senses, argument senses, and label semantic roles. It fully joins all of the tasks. Both of all word sense disambiguation and semantic role labeling performances can be further improved. Although the improvements are not significant compared with the best pipeline system, they significantly (z-test with $\rho < 0.1$ and χ^2-test with $\rho < 0.01$ respectively) outperform the baseline system. Additionally, the performance of the fully joint system does not outperform partly joint systems significantly. The reason seems to be that there is some overlap among the contributions of the three joint systems.

6 Conclusion

In this paper, we presented a Markov logic model that jointly models all word sense disambiguation and semantic role labeling. We got the following conclusions:

1. The baseline systems with Markov logic is competitive to the state-of-the-art word sense disambiguation and semantic role labeling systems on OntoNotes 3.0 corpus.

2. The predicate sense disambiguation is beneficial to semantic role labeling. However, the automatic argument sense disambiguation itself is harmful to the task. It must be combined with the predicate sense disambiguation.

3. The semantic role labeling not only can help predicate sense disambiguation, but also argument sense disambiguation (a little). In contrast, because of the limitation of the pipeline model, it is difficult to make semantic role labeling to help predicate and argument sense disambiguation simultaneously.

4. It is easy to implement the joint model of all word sense disambiguation and semantic role labeling with Markov logic. More important, the joint model can further improve the performance of the all word sense disambiguation and semantic role labeling than pipeline systems.

Acknowledgement

This work was supported by National Natural Science Foundation of China (NSFC) via grant 60803093, 60975055, the “863” National High-Tech Research and Development of China via grant 2008AA01Z144, and Natural Scientific Research Innovation Foundation in Harbin Institute of Technology (HIT.NSRIF.2009069).

References

Che, Wanxiang, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin, and Ting Liu. 2009. Multilingual dependency-based syntactic and semantic parsing. In *Proceedings of CoNLL 2009: Shared Task*, pages 49–54, June.

Che, Wanxiang, Ting Liu, and Yongqiang Li. 2010. Improving semantic role labeling with word sense. In *NAACL 2010*, pages 246–249, June.

Crammer, Koby and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Dahlmeier, Daniel, Hwee Tou Ng, and Tanja Schultz. 2009. Joint learning of preposition senses and semantic roles of prepositional phrases. In *Proceedings of EMNLP 2009*, pages 450–458, August.

Dang, Hoa Trang and Martha Palmer. 2005. The role of semantic roles in disambiguating verb senses. In *Proceedings of ACL 2005*, pages 42–49, Morristown, NJ, USA.

Domingos, Pedro and Daniel Lowd. 2009. *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Finkel, Jenny Rose and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proceedings of NAACL 2009*, pages 326–334, June.

Goldberg, Yoav and Reut Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *Proceedings of ACL 2008*, pages 371–379, June.

Hajič, Jan, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL 2009: Shared Task*, pages 1–18, June.

Hovy, Eduard, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of NAACL 2006*, pages 57–60, June.

Jiang, Wenbin, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL 2008*, pages 897–904, June.

Kruengkrai, Canasai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint chinese word segmentation and pos tagging. In *Proceedings of ACL-IJCNLP 2009*, pages 513–521, August.

Meza-Ruiz, Ivan and Sebastian Riedel. 2009. Jointly identifying predicates, arguments and senses using markov logic. In *Proceedings of NAACL 2009*, pages 155–163, June.

Poon, Hoifung and Pedro Domingos. 2008. Joint unsupervised coreference resolution with Markov Logic. In *Proceedings of EMNLP 2008*, pages 650–659, October.

Punyakanok, Vasin, Dan Roth, and Wen tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

Riedel, Sebastian. 2008. Improving the accuracy and efficiency of map inference for markov logic. In *Proceedings of UAI 2008*, pages 468–475. AUAI Press.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL 2008*, pages 159–177, August.

Titov, Ivan and Ryan McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL 2008*, pages 308–316, June.

Toutanova, Kristina and Colin Cherry. 2009. A global model for joint lemmatization and part-of-speech prediction. In *Proceedings of ACL-IJCNLP 2009*, pages 486–494, August.

Toutanova, Kristina, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2).

Yoshikawa, Katsumasa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with markov logic. In *Proceedings of ACL-IJCNLP 2009*, pages 405–413, August.

Zhang, Yue and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL 2008*, pages 888–896, June.

Zhong, Zhi, Hwee Tou Ng, and Yee Seng Chan. 2008. Word sense disambiguation using OntoNotes: An empirical study. In *Proceedings of EMNLP 2008*, pages 1002–1010, October.

Bipolar Person Name Identification of Topic Documents Using Principal Component Analysis

Chein Chin Chen
Department of Information Management
National Taiwan University
paton@im.ntu.edu.tw

Chen-Yuan Wu
Department of Information Management
National Taiwan University
r97725035@ntu.edu.tw

Abstract

In this paper, we propose an unsupervised approach for identifying bipolar person names in a set of topic documents. We employ principal component analysis (PCA) to discover bipolar word usage patterns of person names in the documents and show that the signs of the entries in the principal eigenvector of PCA partition the person names into bipolar groups spontaneously. Empirical evaluations demonstrate the efficacy of the proposed approach in identifying bipolar person names of topics.

1 Introduction

With the advent of Web2.0, many online collaborative tools, e.g., weblogs and discussion forums are being developed to allow Internet users to express their perspectives on a wide variety of topics via Web documents. One benefit is that the Web has become an invaluable knowledge base for Internet users to learn about a topic comprehensively. Since the essence of Web2.0 is knowledge sharing, collaborative tools are generally designed with few constraints so that users will be motivated to contribute their knowledge. As a result, the number of topic documents on the Internet is growing exponentially. Research subjects, such as topic threading and timeline mining (Nallapati et al., 2004; Feng and Allan, 2007; Chen and Chen, 2008), are thus being studied to help Internet users comprehend numerous topic documents efficiently.

A topic consists of a sequence of related events associated with a specific time, place, and person(s) (Nallapati et al., 2004). Topics that involve bipolar (or competitive) viewpoints are often attention-getting and attract a large number of topic documents. For such topics, identifying the polarity of the named entities, especially person names, in the topic documents would help readers learn the topic efficiently. For instance, for the 2008 American presidential election, Internet users can find numerous Web documents about the Democrat and Republican parties. Identifying important people in the competing parties would help readers form a balanced view of the campaign.

Existing works on topic content mining focus on extracting important themes in topics. In this paper, we propose an unsupervised approach that identifies bipolar person names in a set of topic documents automatically. We employ principal component analysis (PCA) (Smith, 2002) to discover bipolar word usage patterns of important person names in a set of topic documents, and show that the signs of the entries in the principal eigenvector of PCA partition the person names in bipolar groups spontaneously. In addition, we present two techniques, called off-topic block elimination and weighted correlation coefficient, to reduce the effect of data sparseness on person name bipolarization. The results of experiments based on two topic document sets written in English and Chinese respectively demonstrate that the proposed PCA-based approach is effective in identifying bipolar person names. Furthermore, the approach is language independent.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 170–178,
Beijing, August 2010

2 Related Work

Our research is closely related to opinion mining, which involves identifying the polarity (or sentiment) of a word in order to extract positive or negative sentences from review documents (Ganapathibhotla and Liu, 2008). Hatzivassiloglou and McKeown (1997) validated that language conjunctions, such as *and*, *or*, and *but*, are effective indicators for judging the polarity of conjoined adjectives. The authors observed that most conjoined adjectives (77.84%) have the same orientation, while conjunctions that use *but* generally connect adjectives of different orientations. They proposed a log-linear regression model that learns the distributions of conjunction indicators from a training corpus to predict the polarity of conjoined adjectives. Turney and Littman (2003) manually selected seven positive and seven negative words as a polarity lexicon and proposed using pointwise mutual information (PMI) to calculate the polarity of a word. A word has a positive orientation if it tends to co-occur with positive words; otherwise, it has a negative orientation. More recently, Esuli and Sebastiani (2006) developed a lexical resource, called SentiWordNet, which calculates the degrees of objective, positive, and negative sentiments of a synset in WordNet. The authors employed a bootstrap strategy to collect training datasets for the sentiments and trained eight sentiment classifiers to assign sentiment scores to a synset. Kanayama and Nasukawa (2006) posited that polar clauses with the same polarity tend to appear successively in contexts. The authors derived the coherent precision and coherent density of a word in a training corpus to predict the word's polarity. Ganapathibhotla and Liu (2008) investigated comparative sentences in product reviews. To identify the polarity of a comparative word (e.g., longer) with a product feature (e.g., battery life), the authors collected phrases that describe the Pros and Cons of products from Epinions.com and proposed one-side association (OSA), which is a variant of PMI. OSA assigns a positive (negative) orientation to the comparative-feature combination if the synonyms of the comparative word and feature tend to co-occur in the Pros (resp. Cons) phrases.

Our research differs from existing approaches in three respects. First, most works identify the polarity of adjectives and adverbs because the syntactic constructs generally express sentimental semantics. In contrast, our method identifies the polarity of person names. Second, to the best of our knowledge, all existing polarity identification methods require external information sources (e.g., WordNet, manually selected polarity words, or training corpora). However, our method identifies bipolar person names by simply analyzing person name usage patterns in topic documents without using external information. Finally, our method does not require any language constructs, such as conjunctions; hence, it can be applied to different languages.

3 Method

3.1 Data Preprocessing

Given a set of topic documents, we first decompose the documents into a set of non-overlapping *blocks* $B = \{b_1, b_2, \ldots, b_n\}$. A block can be a paragraph or a document, depending on the granularity of PCA sampling. Let $U = \{u_1, u_2, \ldots, u_m\}$ be a set of *textual units* in B. In this study, a unit refers to a person name. Then, the document set can be represented as an *m*x*n* *unit-block association matrix A*. A column in A, denoted as $\underline{b_i}$, represents a decomposed block *i*. It is an *m*-dimensional vector whose *j*'th entry, denoted as $b_{i,j}$, is the frequency of u_j in b_i. In addition, a row in A, denoted as $\underline{u_i}$, represents a textual unit *i*; and it is an *n*-dimensional vector whose *j*'th entry, denoted as $u_{i,j}$, is the frequency of u_i in b_j.

3.2 PCA-based Person Name Bipolarization

Principal component analysis is a well-known statistical method that is used primarily to identify the most important feature pattern in a high-dimensional dataset (Smith, 2002). In our research, it identifies the most important unit pattern in the topic blocks by first constructing an *m*x*m* *unit relation matrix R*, in which the (*i,j*)-entry (denoted as $r_{i,j}$) denotes the correlation coefficient of u_i and u_j. The correlation is computed as follows:

$$r_{i,j} = corr(u_i, u_j) = \frac{\sum_{k=1}^{n}(u_{i,k} - u_i^{\sim}) * (u_{j,k} - u_j^{\sim})}{\sqrt{\sum_{k=1}^{n}(u_{i,k} - u_i^{\sim})^2} * \sqrt{\sum_{k=1}^{n}(u_{j,k} - u_j^{\sim})^2}},$$

where $u_i^{\sim}=1/n\sum^{n}_{k=1}u_{i,k}$ and $u_j^{\sim}=1/n\sum^{n}_{k=1}u_{j,k}$ are the average frequencies of units *i* and *j* respectively.

The range of $r_{i,j}$ is within [-1,1] and the value represents the degree of correlation between u_i and u_j under the decomposed blocks. If $r_{i,j} = 0$, we say that u_i and u_j are uncorrelated; that is, occurrences of unit u_i and unit u_j in the blocks are independent of each other. If $r_{i,j} > 0$, we say that units u_i and u_j are positively correlated. That is, u_i and u_j tend to co-occur in the blocks; otherwise, both tend to be jointly-absent. If $r_{i,j} < 0$, we say that u_i and u_j are negatively correlated; that is, if one unit appears, the other tends not to appear in the same block simultaneously. Note that if $r_{i,j} \neq 0$, $|r_{i,j}|$ scales the strength of a positive or negative correlation. Moreover, since the correlation coefficient is commutative, $r_{i,j}$ will be identical to $r_{j,i}$ such that matrix R will be symmetric.

A unit pattern is represented as a vector $\underline{v}$ of dimension m in which the i'th entry v_i indicates the weight of i'th unit in the pattern. Since matrix R depicts the correlation of the units in the topic blocks, given a constituent of $\underline{v}$, $\underline{v}^T R \underline{v}$ computes the variance of the pattern to characterize the decomposed blocks. A pattern is important if it characterizes the variance of the blocks specifically. PCA can then identify the most important unit pattern by using the following object function:

$$\max \underline{v}^T R \underline{v},$$
$$\text{s.t. } \underline{v}^T \underline{v} = 1.$$

Without specifying any constraint on $\underline{v}$, the objective function becomes arbitrarily large with large entry values of $\underline{v}$. Constraint $\underline{v}^T \underline{v} = 1$ limits the search space within the set of length-normalized vectors. Chen and Chen (2008) show that the desired $\underline{v}$ for the above constrained optimization problem is the eigenvector of R with the largest eigenvalue. Furthermore, as R is a symmetric matrix, such an eigenvector always exists (Spence et al., 2000) and the optimization problem is solvable.

PCA is not the only method that identifies important textual patterns in terms of eigenvectors. For instance, Gong and Liu (2001), Chen and Chen (2008) utilize the eigenvectors of symmetric matrices to extract salient concepts and salient themes from documents respectively[1]. The difference between PCA and other eigenvector-based approaches lies in the way the unit relation matrix is constructed. PCA calculates $r_{i,j}$ by using the correlation coefficient, whereas the other approaches employ the inner product or cosine formula[2] (Manning et al., 2008) to derive the relationship between textual units. Specifically, the correlation coefficient is identical to the cosine formula if we normalize each unit with its mean:

$$corr(u_i, u_j) = \frac{\sum_{k=1}^{n} (u_{i,k} - u_i^{\sim}) * (u_{j,k} - u_j^{\sim})}{\sqrt{\sum_{k=1}^{n} (u_{i,k} - u_i^{\sim})^2} * \sqrt{\sum_{k=1}^{n} (u_{j,k} - u_j^{\sim})^2}}$$
$$= \frac{\sum_{k=1}^{n} u_{i,k}^{*} * u_{j,k}^{*}}{\sqrt{\sum_{k=1}^{n} u_{i,k}^{*\,2}} * \sqrt{\sum_{k=1}^{n} u_{j,k}^{*\,2}}}$$
$$= \text{cosine}(\underline{u}_i^{*}, \underline{u}_j^{*}),$$

where $\underline{u}_i^{*} = \underline{u}_i - u_i^{\sim}[1,1,\ldots,1]^T$; $\underline{u}_j^{*} = \underline{u}_j - u_j^{\sim}[1, 1,\ldots,1]^T$; and are the mean-normalized vectors of $\underline{u}_i$ and $\underline{u}_j$, respectively. Conceptually, the mean normalization process is the only difference between PCA and other eigenvector-based approaches.

Since the eigenvectors of a symmetric matrix form an orthonormal basis of R^m, they may contain negative entries (Spence et al., 2000). Even though Kleinberg (1999) and Chen and Chen (2008) have shown experimentally that negative entries in an eigenvector are as important as positive entries for describing a certain unit pattern, the meaning of negative entries in their approaches is unexplainable. This is because textual units (e.g., terms, sentences, and documents) in information retrieval are usually characterized by frequency-based metrics, e.g., term frequency, document frequency, or TFIDF (Manning et al., 2008), which can never be negative. In PCA, however, the mean normalization process of the correlation coefficient gives bipolar meaning to positive and negative entries and that helps us partition textual units into bipolar groups in accordance with their signs in $\underline{v}$.

[1] The right singular vectors of a matrix A used by Gong and Liu (2001) are equivalent to the eigenvectors of a symmetric matrix A^TA whose entries are the inner products of the corresponding columns of A.

[2] The inner product is equivalent to the cosine formula when the calculated vectors are length normalized (Manning et al., 2008).

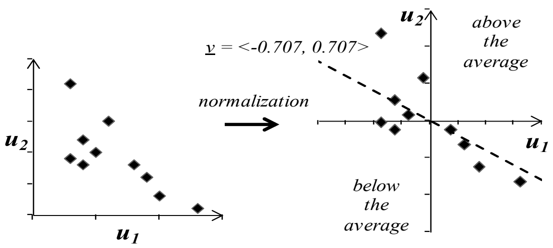


Figure 1. The effect of the mean normalization process.

The synthesized example in Figure 1 illustrates the effect of the normalization process. In this example, we are only interested in textual units u_1 and u_2; the corpus consists of ten blocks. Graphically, each block can be represented as a point in a 2-dimensional vector space. The mean normalization process moves the origin of the 2-dimensional vector space to the centroid of the blocks that makes negative unit values explainable. A negative unit of a block in this normalized vector space indicates that the number of occurrences of the unit in the block is less than the unit's average; by contrast, a positive unit means that the number of occurrences of the unit in a block is above the average. In the figure, the most important unit pattern $\underline{v}$ <-0.707, 0.707> calculated by PCA is represented by the dashed line. The signs of $\underline{v}$'s entries indicate that the occurrence of u_1 will be lower than the average if u_2 occurs frequently in a block. In addition, as the signs of entries in an eigenvector are invertible (Spence et al., 2000), the constituent of $\underline{v}$ also claims that if u_1 occurs frequently in a block, then the probability that we will observe u_2 in the same block will be lower than expected. The instances of bipolar word usage behavior presented in $\underline{v}$ are consistent with the distribution of the ten blocks. As mentioned in Section 2, Kanayama and Nasukawa (2006) validated that polar text units with the same polarity tend to appear together to make contexts coherent. Consequently, we believe that the signs in PCA's principal eigenvector are effective in partitioning textual units into bipolar groups.

3.3 Sparseness of Textual Units

A major problem with employing PCA to process textual data is the sparseness of textual units. To illustrate this problem, we collected 411 news documents about the 2009 NBA Finals from Google News and counted the frequency that each person name occurred in the documents. We also evaluate the documents in the experiment section to determine if the proposed approach is capable of bipolarizing the person names into the teams that played in the finals correctly. We rank the units according to their frequencies and list the frequencies in descending order in Figure 2. The figure shows that the frequency distribution follows Zipf's law (Manning et al., 2008); and for most units, the distribution in a block will be very sparse.

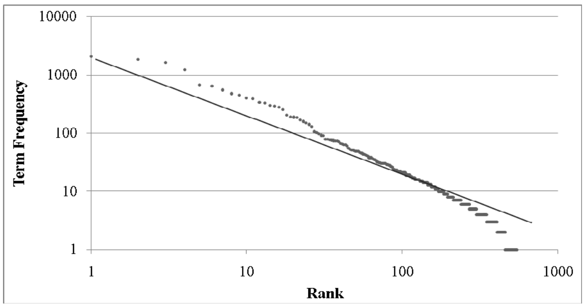


Figure 2. The rank-frequency distribution of person names on logarithmic scales (base 10).

We observe that a unit will not to occur in a block in the following three scenarios. 1) The polarity of the block is the opposite of the polarity of the unit. For instance, if the unit represents a player in one team and the block narrates information about the other team, the block's author would not mention the unit in the block to ensure that the block's content is coherent. 2) Even if the polarity of a block is identical to that of the unit; the length of the block may not be sufficient to contain the unit. 3) The block is off-topic so the unit will not appear in the block. In the last two scenarios, the absence of units will impact the estimation of the correlation coefficient. To alleviate the problem, we propose two techniques, the weighted correlation coefficient and off-block elimination, which we describe in the following sub-sections.

Weighted Correlation Coefficient

The so-called data sparseness problem in scenario 2 affects many statistical information retrieval and language models (Manning et al., 2008). For units with the same polarity, data sparseness could lead to underestimation of their correlations because the probability that the units will occur together is reduced. Conversely, for uncorrelated units or units with opposite polarities, data sparseness may lead to overestimation of their correlations because they are frequently jointly-absent in the decomposed blocks. While smoothing approaches, such as Laplace's law (also known as adding-one smoothing), have been developed to alleviate data sparseness in language models (Manning et al., 2008), they are not appropriate for PCA. This is because the correlation coefficient of PCA measures the divergence between units from their means, so adding one to each block unit will not change the divergence. To summarize, data sparseness could influence the correlation coefficient when units do not co-occur. Thus, for two units u_i and u_j, we separate B into co-occurring and non-co-occurring parts and apply the following weighted correlation coefficient:

$$corr_w(u_i, u_j) =$$

$$\left(\begin{array}{l} (1-\alpha) \sum_{b \in co(i,j)} (u_{i,b} - u_i^{\sim}) * (u_{j,b} - u_j^{\sim}) + \\ \qquad \alpha \sum_{b \in B-co(i,j)} (u_{i,b} - u_i^{\sim}) * (u_{j,b} - u_j^{\sim}) \end{array} \right) /$$

$$\sqrt{(1-\alpha) \sum_{b \in co(i,j)} (u_{i,b} - u_i^{\sim})^2 + \alpha \sum_{b \in B-co(i,j)} (u_{i,b} - u_i^{\sim})^2} *$$

$$\sqrt{(1-\alpha) \sum_{b \in co(i,j)} (u_{j,b} - u_j^{\sim})^2 + \alpha \sum_{b \in B-co(i,j)} (u_{j,b} - u_j^{\sim})^2},$$

where $corr_w(u_i,u_j)$ represents the weighted correlation coefficient between units i and j; and $co(i,j)$ denotes the set of blocks in which units i and j co-occur. The range of parameter α is within [0,1]. It weights the influence of non-co-occurring blocks when calculating the correlation coefficient. When $\alpha = 0.5$, the equation is equivalent to the standard correlation coefficient; and when $\alpha = 0$, the equation only considers the blocks in which units i and j co-occur. Conversely, when $\alpha = 1$, only non-co-occurring blocks are employed to calculate the units' correlation. In the experiment section, we will examine the effect of α on bipolar person name identification.

Off-topic Block Elimination

Including off-topic blocks in PCA will lead to overestimation of the correlation between units. This is because units are usually jointly-absent from off-topic blocks that make uncorrelated or even negatively correlated units positively correlated. To eliminate the effect of off-topic blocks on unit bipolarization, we construct a centroid of all the decomposed blocks by averaging $\underline{b}_i$'s. Then, blocks whose cosine similarity to the centroid is lower than a predefined threshold β are excluded from calculation of the correlation coefficient.

4 Performance Evaluations

In this section, we evaluate two topics with bipolar (or competitive) viewpoints to demonstrate the efficacy of the proposed approach.

4.1 The 2009 NBA Finals

For this experiment, we collected 411 news documents about the 2009 NBA Finals from Google News during the period of the finals (from 2009/06/04 to 2009/06/16). The matchup of the finals was Lakers versus Orlando Magic. In this experiment, a block is a topic document, as paragraph tags are not provided in the evaluated documents. First, we parsed the blocks by using Stanford Named Entity Recognizer[3] to extract all possible named entities. We observed that the parser sometimes extracted false entities (such as Lakers Kobe) because the words in the headlines were capitalized and that confused the parser. To reduce the effect of false extraction by the parser, we examined the extracted named entities manually. After eliminating false entities, the dataset comprised 546 unique named entities; 538 were person names and others represented organizations, such as basketball teams and basketball courts. To examine the effect of the weighted correlation coefficient, parameter α is set between 0 and 1, and increased in increments of 0.1; and the threshold β used by off-topic block elimination is set at 0.3. The frequency distribution of the person names, shown in Figure 2, indicates that many of the person names rarely appeared in the examined blocks, so their distribution was too sparse for PCA. Hence, in the following subsections, we sum the frequencies of the 538 person names in the examined blocks. We select the first k frequent person names, whose accumulated term frequencies reach 60% of the total frequencies, for evaluation. In other words, the evaluated person names account for 60% of the person name occurrences in the examined blocks.

For each parameter setting, we perform principal component analysis on the examined blocks and the selected entities, and partition the entities into two bipolar groups according to

[3] http://nlp.stanford.edu/software/CRF-NER.shtml

their signs in the principal eigenvector. To evaluate the accuracy rate of bipolarization, we need to label the team of each bipolar group. Then, the accuracy rate is the proportion of the entities in the groups that actually belong to the labeled teams. Team labeling is performed by examining the person names in the larger bipolarization group. If the majority of the entities in the group belong to the Lakers (Magic), we label the group as Lakers (Magic) and the other group as Magic (Lakers). If the two bipolar groups are the same size, the group that contains the most Lakers (Magic) entities is labeled as Lakers (Magic), and the other group is labeled as Magic (Lakers). If both groups contain the same number of Lakers (Magic) entities, we randomly assign team labels because all assignments produce the same accuracy score. To the best of our knowledge, there is no similar work on person name bipolarization; therefore, for comparison, we use a baseline method that assigns the same polarity to all the person names.

Magic		Lakers	
Dwight Howard	0.0884	Derek Fisher	-0.0105
Hedo Turkoglu	0.1827	Kobe Bryant	-0.2033
Jameer Nelson	0.3317	Lamar Odom	-0.1372
Jeff Van Gundy^{*+}	0.3749	LeBron James$^{*\wedge}$	-0.0373
Magic Johnson*	0.3815	Mark Jackson$^{*\wedge}$	-0.2336
Rafer Alston	0.3496	Pau Gasol	-0.1858
Rashard Lewis	0.1861	Paul Gasol^{*+}	-0.1645
Stan Van Gundy	0.4035	Phil Jackson	-0.2553

Table 1. The bipolarization results for NBA person names. ($\alpha = 0.8$ and $\beta = 0.3$)

Table 1 shows the bipolarization results for frequent person names in the dataset. The parameter α is set at 0.8 because of its superior performance. The left-hand column of the table lists the person names labeled as Magic and their entry values in the principal eigenvector; and the right-hand column lists the person names labeled as Lakers. It is interesting to note that the evaluated entities contain person names irrelevant to the players in the NBA finals. For instance, the frequency of Magic Johnson, an ex-Lakers player, is high because he constantly spoke in support of the Lakers during the finals. In addition, many documents misspell Pau Gasol as Paul Gasol. Even though the names refer to the same player, the named entity recognizer parses them as distinct entities. We propose two evaluation strategies, called *strict evaluation* and *non-strict evaluation*. The strict evaluation strategy treats the person names that do not refer to the players, coaches in the finals as false positives. Under the non-strict strategy, the person names that are closely related to Lakers or Magic players, such as a player's relatives or misspellings, are deemed true positives if they are bipolarized into the correct teams. In Table 1, a person name annotated with the symbol * indicates that the entity is bipolarized incorrectly. For instance, Magic Johnson is not a member of Magic. The symbol ^ indicates that the person name is neutral (or irrelevant) to the teams in the finals. In addition, the symbol + indicates that the person name represents a relative of a member of the team he/she is bipolarized to; or the name is a misspelling, but it refers to a member of the bipolarized team. This kind of bipolarization is correct under the non-strict evaluation strategy. As shown in Table 1, the proposed method bipolarizes the important persons in the finals correctly without using any external information source. The accuracy rates of strict and non-strict evaluation are 68.8% and 81.3% respectively. The rates are far better than those of the baseline method, which are 37.5% and 43.8% respectively. If we ignore the neutral entities, which are always wrong no matter what bipolarization approach is employed, the strict and non-strict accuracies are 78.6% and 92.9% respectively. In the non-strict evaluation, we only misbipolarized Magic Johnson as Magic. The mistake also reflects a problem with person name resolution when the person names that appear in a document are ambiguous. In our dataset, the word 'Magic' sometimes refers to Magic Johnson and sometimes to Orlando Magic. Here, we do not consider a sophisticated person name resolution scheme; instead, we simply assign the frequency of a person name to all its specific entities (e.g., Magic to Magic Johnson, and Kobe to Kobe Bryant) so that specific person names are frequent enough for PCA. As a result, Magic Johnson tends to co-occur with the members of Magic and is incorrectly bipolarized to the Magic team. Another interesting phenomenon is that LeBron James (a player with Cavaliers) is incorrectly bipolarized to Lakers. This is because Kobe Bryant (a player with Lakers) and LeBron James were rivals for the most valuable player (MVP) award in the 2009 NBA season. The documents that mentioned Kobe Bryant during the finals often compared him with LeBron

James to attract the attention of readers. As the names often co-occur in the documents, LeBron James was wrongly classified as a member of Lakers.

Figures 3 and 4 illustrate the effects of the weighted correlation coefficient and off-topic block elimination on NBA person name bipolarization. As shown in the figures, eliminating off-topic blocks generally improves the system performance. It is noteworthy that, when off-topic blocks are eliminated, large α values produce good bipolarization performances. As mentioned in Section 3.3, a large α implies that non-co-occurring blocks are important for calculating the correlation between a pair of person names. When off-topic blocks are eliminated, the set of non-co-occurring blocks specifically reveals opposing or jointly-absent relationships between entities. Therefore, the bipolarization performance improves as α increases. Conversely, when off-topic blocks are not eliminated, the set of non-co-occurring blocks will contain off-topic blocks. As both entities in a pair tend to be absent in off-topic blocks, a large α value will lead to overestimation of the correlation between bipolar entities. Consequently, the bipolarization accuracy decreases as α increases. It is also interesting to note that the bipolarization performance decreases as α decreases. We observed that some of the topic documents are recaps of the finals, which tend to mention Magic and Lakers players together. As a small α value makes co-occurrence blocks important, recap-style documents will overestimate the correlation between bipolar entities. Consequently, the bipolarization performance is inferior when α is small.

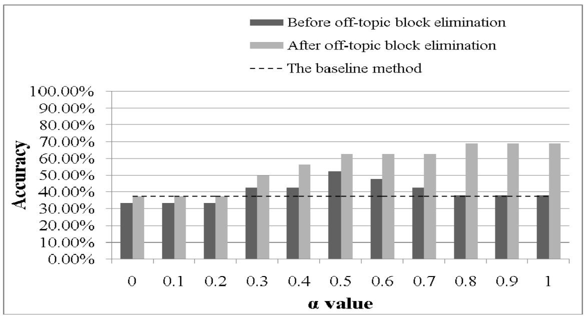


Figure 3. The effects of the weighted correlation coefficient and off-topic block elimination on NBA person name bipolarization. (Strict)

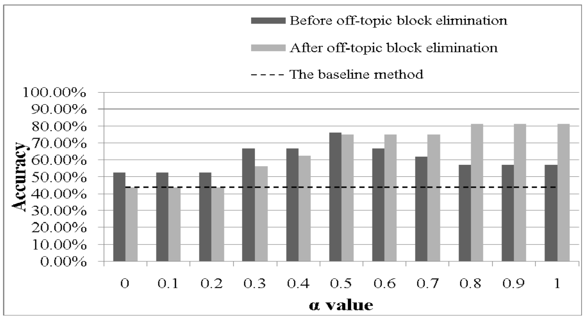


Figure 4. The effects of the weighted correlation coefficient and off-topic block elimination on NBA person name bipolarization. (Non-strict)

4.2 Taiwan's 2009 Legislative By-Elections

For this experiment, we evaluated Chinese news documents about Taiwan's 2009 legislative by-elections, in which two major parties, the Democratic Progressive Party (DPP) and the KouMinTang (KMT), campaigned for three legislative positions. Since the by-elections were regional, not many news documents were published during the campaign. In total, we collected 89 news documents that were published in The Liberty Times [4] during the election period (from 2009/12/27 to 2010/01/11). Then, we used a Chinese word processing system, called Chinese Knowledge and Information Processing (CKIP)[5], to extract possible Chinese person names in the documents. Once again, the names were examined manually to remove false extractions. The dataset comprised 175 unique person names. As many of the names only appeared once, we selected the first k frequent person names whose accumulated frequency was at least 60% of the total term frequency count of the person names for evaluation. We calculated the accuracy of person name bipolarization by the same method as the NBA experiment in order to assess how well the bipolarized groups represented the KMT and the DPP. As none of the selected names were misspelled, we do not show the non-strict accuracy of bipolarization. The threshold β is set at 0.3, and each block is a topic document.

Table 2 shows the bipolarization results for the frequent person names of the candidates of the respective parties, the party chair persons, and important party staff members. The accuracy rates of the bipolarization and the baseline me-

[4] http://www.libertytimes.com.tw/index.htm
[5] http://ckipsvr.iis.sinica.edu.tw/

thods are 70% and 50%, respectively. It is noteworthy that the chairs of the DPP and the KMT, who are Ing-wen Tsai and Ying-jeou Ma respectively, are correctly bipolarized. We observed that, during the campaign, the chairs repeatedly helped their respective party's candidates gain support from the public. As the names of the chairs and the candidates often co-occur in the documents, they can be bipolarized accurately. We also found that our approach bipolarized two candidates incorrectly if the competition between them was fierce. For instance, Kun-cheng Lai and Li-chen Kuang campaigned intensively for a single legislative position. As they often commented on each other during the campaign, they tend to co-occur in the topic documents. PCA therefore misclassifies them as positively correlated and incorrectly groups Kun-cheng Lai with the KMT party.

KMT (國民黨)		DPP (民進黨)	
Kun-cheng Lai (賴坤成)*	0.39	Wen-chin Yu (余文欽)*	-0.56
Li-chen Kuang (鄺麗貞)	0.40	Den-yih Wu (吳敦義)*	-0.03
Li-ling Chen (陳麗玲)	0.01	Chao-tung Chien (簡肇棟)	-0.56
Ying-jeou Ma (馬英九)	0.05	Ing-wen Tsai (蔡英文)	-0.17
		Tseng-chang Su (蘇貞昌)	-0.01
		Jung-chung Kuo (郭榮宗)	-0.01

Table 2. The bipolarization results for the election dataset. ($\alpha = 0.7$)

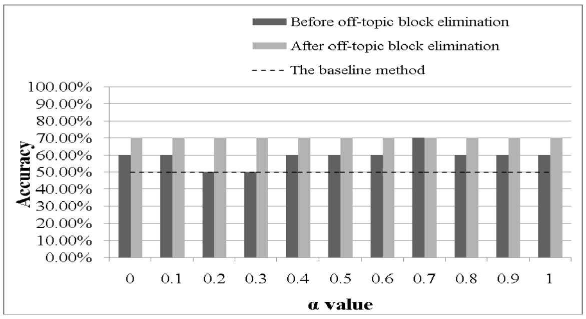


Figure 5. The effects of the weighted correlation coefficient and off-topic block elimination.

Figure 5 shows that off-topic block elimination is effective in person name bipolarization. However, the weighted correlation coefficient only improves the bipolarization performance slightly. We have investigated this problem and believe that the evaluated person names in the documents are frequent enough to prevent the data sparseness problem. While the weighted correlation coefficient does not improve the bipolarization performance significantly, the proposed PCA-based approach can still identify the bipolar parties of important persons accurately. Unlike the results in the last section, the accuracy rate in this experiment does not decrease as α decreases. This is because the topic documents generally report news about a single party. As the documents rarely recap the activities of parties, the co-occurrence blocks accurately reflect the bipolar relationship between the persons. Hence, a small α value can identify bipolar person names effectively.

The evaluations of the NBA and the election datasets demonstrate that the proposed PCA-based approach identifies bipolar person names in topic documents effectively. As the writing styles of topic documents in different domains vary, the weighted correlation coefficient may not always improve bipolarization performance. However, because we eliminate off-topic blocks, a large α value always produces superior bipolarization performances.

5 Conclusion

In this paper, we have proposed an unsupervised approach for identifying bipolar person names in topic documents. We show that the signs of the entries in the principal eigenvector of PCA can partition person names into bipolar groups spontaneously. In addition, we introduce two techniques, namely the weighted correlation coefficient and off-topic block elimination, to address the data sparseness problem. The experiment results demonstrate that the proposed approach identifies bipolar person names of topics successfully without using any external knowledge; moreover, it is language independent. The results also show that off-topic block elimination along with a large α value for the weighted correlation coefficient generally produce accurate person name bipolarization. In the future, we will integrate text summarization techniques with the proposed bipolarization method to provide users with polarity-based topic summaries. We believe that summarizing important information about different polarities would help users gain a comprehensive knowledge of a topic.

Acknowledge

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was supported in part by NSC 97-2221-E-002-225-MY2.

References

Chen, Chien Chin and Meng Chang Chen. 2008. TSCAN: a novel method for topic summarization and content anatomy. In *Proceedings of the 31st annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 579-586.

Esuli, Andrea and Fabrizio Sebastiani. 2006. SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining. In *Proceedings of the 5th Conference on Language Resources and Evaluation*.

Feng, Ao and James Allan. 2007. Finding and Linking Incidents in News. In *Proceedings of the sixteenth ACM Conference on information and knowledge management*, pages 821-830.

Ganapathibhotla, Murthy and Bing Liu. 2008. Mining Opinions in Comparative Sentences. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 241-248.

Gong, Yihong and Xin Liu. 2001. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 19-25.

Hatzivassiloglou, Vasileios and Kathleen R. McKeown. 1997. Predicting the Semantic Orientation of Adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 174-181.

Kanayama, Hiroshi and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 355-363.

Kleinberg, Jon M.. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM 46, 5*, pages 604-632.

Manning, Christopher D., Prabhakar Raghavan and Hinrich Schutze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Nallapati, Ramesh, Ao Feng, Fuchun Peng and James Allan. 2004. Event Threading within News Topics. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 446-453.

Smith, Lindsay I.. 2002. *A Tutorial on Principal Components Analysis*. Cornell University.

Spence, Lawrence E., Arnold J. Insel and Stephen H. Friedberg. 2000. *Elementary Linear Algebra, A Matrix Approach*. Prentice Hall.

Turney, Peter D., and Michael L. Littman. 2003. Measuring Praise and Criticism: Inference of Semantic Orientation from Association. *ACM Transactions on Information Systems (TOIS)*, pages 315-346.

Emotion Cause Detection with Linguistic Constructions

Ying Chen[*†], Sophia Yat Mei Lee[†], Shoushan Li[†], Chu-Ren Huang[†]

[*] Dep. of Computer Engineering
China Agricultural University

[†]Dep. of Chinese and Bilingual Studies
The Hong Kong Polytechnic University

{chenying3176,sophiaym,shoushan.li,churenhuang}@gmail.com

Abstract

This paper proposes a multi-label approach to detect emotion causes. The multi-label model not only detects multi-clause causes, but also captures the long-distance information to facilitate emotion cause detection. In addition, based on the linguistic analysis, we create two sets of linguistic patterns during feature extraction. Both manually generalized patterns and automatically generalized patterns are designed to extract general cause expressions or specific constructions for emotion causes. Experiments show that our system achieves a performance much higher than a baseline model.

1 Introduction

Text-based emotion processing has been a center of attention in the NLP field in the past few years. Most previous researches have focused on detecting the surface information of emotions, especially emotion classes, e.g., "happiness" and "anger" (Mihalcea and Liu 2006, Strapparava and Mihalcea 2008, Abbasi et al, 2008, Tokuhisa et al. 2008). Although most emotion theories recognize the important role of causes in emotion analysis (Descartes, 1649; James, 1884; Plutchik 1980, Wierzbicka 1999), very few studies explore the interactions between emotion and causes. Emotion-cause interaction is the eventive relation which potentially yields the most crucial information in terms of information extraction. For instance, knowing the existence of an emotion is often insufficient to predict future events or decide on the best reaction. However, if the emotion cause is known in addition to the type of emotion, prediction of future events or assessment of potential implications can be done more reliably. In other words, when emotion is treated as an event, causal relation is the pivotal relation to discover. In this paper, we explore one of the crucial deep level types of information of emotion, i.e. cause events.

Our study focuses on explicit emotions in which emotions are often presented by emotion keywords such as "*shocked*" in "*He was shocked after hearing the news*". Emotion causes are the explicitly expressed propositions that evoke the presence of the corresponding emotions. They can be expressed by verbs, nominalizations, and nominals. Lee et al. (2010a) explore the causes of explicit emotions by constructing a Chinese emotion cause corpus. Based on this corpus, we formalize the emotion cause detection problem through extensive data analysis. We find that ~14% emotion causes are complicated events containing multi-clauses, to which previous cause detection systems can hardly be applied directly. Most previous cause detection systems focus on the causal relation between a pair of small-size text units, such as clauses or phrases. They are thus not able to detect emotion causes that are multi-clauses. In this paper, we formalize emotion cause detection as a multi-label classification task (i.e. each instance may contain more than one label), which allows us to capture long-distance information for emotion cause detection.

In term of feature extraction, as emotion cause detection is a case of cause detection, some typical patterns used in existing cause detection systems, e.g., "*because*" and "*thus*", can be adopted. In addition, various linguistic cues are examined which potentially indicate emotion causes, such as causative verbs and epistemic markers (Lee at al. 2010a). Then some linguistic patterns of emotion causes are manually generalized by examining the linguistic context of the empirical data (Lee et al., 2010b). It is expected that these manually generalized patterns often yield a low-coverage problem. Thus, we extracted features which enable us to automatically capture more emotion-specific constructions. Experiments show that such an integrated system with various linguistic features performs promisingly well. We believe that the present study should provide the foundation for future research on emotion analysis, such as the detection of implicit emotion or cause.

The paper is organized as follows. Section 2 discusses the related work on cause-effect detection. Section 3 briefly describes the emotion cause corpus, and then presents our data analysis. Section 4 introduces the multi-label classification system for emotion cause detection. Section 5 describes the two kinds of features for our system, one is based on hand-coded patterns and the other is the generalized features. Section 6 presents the evaluation and performance of our system. Section 7 highlights our main contributions and the possible future work.

2 Related Work

Most previous studies on textual emotion processing focus on emotion recognition or classification given a known emotion context (Mihalcea and Liu 2006, Strapparava and Mihalcea 2008, Abbasi et al, 2008, Tokuhisa et al. 2008). However, the performance is far from satisfactory. One crucial problem in these works is that they limit the emotion analysis to a simple classification and do not explore the underlying information regarding emotions. Most theories conclude that emotions are often invoked by the perception of external events. An effective emotion recognition model should thus take this into account.

To the best of our knowledge, little research has been done with respect to emotion cause detection. Lee et al. (2010a) first investigate the interactions between emotions and the corresponding causes from a linguistic perspective. They annotate a small-scale emotion cause corpus, and identify six groups of linguistic cues facilitating emotion cause detection. Based on these findings, they develop a rule-based system for automatic emotion cause detection (Lee et al., 2010b).

Emotion cause detection can be considered as a kind of causal relation detection, which has been intensively studied for years. Most previous cause detection studies focus on a specific domain, such as aviation (Persing and Ng, 2009) and finance (Low, et al., 2001). Few works (Marcu and Echihabi, 2002; Girju, 2003; Chang and Choi, 2005) examine causal relation for open domains.

In recognizing causal relations, most existing systems involve two steps: 1) cause candidate identification; 2) causal relation detection. To simplify the task, most systems omit the step of identifying cause candidates. Instead, they often predefine or filter out possible causes based on domain knowledge, e.g., 14 kinds of cause types are identified for aviation incidents (Persing and Ng, 2009). For events without specific domain information, open-domain systems choose to limit their cause candidate. For example, the cause-effect pairs are limited to two noun phrases (Chang and Choi, 2005; Girju, 2003), or two clauses connected with fixed conjunction words (Marcu and Echihabi, 2002).

Given pairs of cause-effect candidates, causal relation detection is considered as a binary classification problem, i.e. “causal” vs. “non-causal”. In general, there are two kinds of information extracted to identify the causal relation. One is patterns or constructions expressing a cause-effect relation (Chang and Choi, 2005; Girju, 2003), and the other is semantic information underlying in a text (Marcu and Echihabi, 2002; Persing and Ng, 2009), such as word pair probability. Undoubtedly, the two kinds of information usually interact with each other in a real cause detection system.

In the literature, the three common classification methods, i.e. unsupervised, semi-supervised, and supervised, have all been used for cause detection systems. Marcu and Echihabi (2002) first collected a cause corpus using an unsupervised approach with the help of several conjunction words, such as “*because*” and “*thus*”, and determined the causal relation for a clause pair using the word pair probability. Chang and Choi (2005) used a semi-supervised method to recursively learn lexical patterns for cause recognition based on syntactic trees. Bethard and Martin (2008) put various causal information in a supervised classifier, such as the temporal information and syntactic information.

For our emotion cause detection, several practical issues need to be investigated and resolved. First, for the identification of cause candidates, we need to define a reasonable span of a cause. Based on our data analysis, we find that emotion causes often appear across phrases or even clauses. Second, although in emotion cause detection the effect is fixed, the cause is open-domain. We also notice that besides the common patterns, emotion causes have their own expression patterns. An effective emotion cause detection system should take them into account.

3 Corpus Analysis

In this section, we briefly introduce the Chinese emotion cause corpus (Lee et al., 2010a), and discuss emotion cause distribution.

3.1 Emotion Cause corpus

Lee at al. (2010a) made the first attempt to explore the correlation between emotions and causes, and annotate a Chinese emotion cause corpus. The emotion cause corpus focuses on five primary emotions, namely “happiness”, “sadness”, “fear”, “anger”, and “surprise”. The emotions are explicitly expressed by emotion keywords, e.g., *gao1xing4* “happy”, *shang1xin1* “sad”, etc. The corpus is created as follows.

1. 6,058 entries of Chinese sentences are extracted from the Academia Sinica Balanced Corpus of Mandarin Chinese (Sinica Corpus) with the pattern-match method as well as the list of 91 Chinese primary emotion keywords (Chen et al., 2009). Each entry contains the focus sentence with the emotion keyword “<FocusSentence>” plus the sentence before “<PrefixSentence>” and after “<SuffixSentence>” it. For each entry, the emotion keywords are indexed since more than one emotion may be presented in an entry;
2. Some preprocessing, such as balancing the number of entry among emotions, is done to remove some entries. Finally, 5,629 entries remain;
3. Each emotion keyword is annotated with its corresponding causes if existing. An emotion keyword can sometimes be associated with more than one cause, in such a case, both causes are marked. Moreover, the cause type is also identified, which is either a nominal event or a verbal event (a verb or a nominalization).

Lee at al. (2010a) notice that 72% of the extracted entries express emotions, and 80% of the emotional entries have a cause.

3.2 The Analysis of Emotion Causes

To have a deeper understanding of emotion cause detection, we take a closer look at the emotion cause distribution, including the distribution of emotion cause occurrence and the distribution of emotion cause text.

The occurrence of emotion causes: According to most emotion theories, an emotion is generally invoked by an external event. The corpus shows that, however, 20% of the emotional entries have no cause. Entries without causes explicitly expressed are mainly due to the following reasons:

i) There is not enough contextual information, for instance the previous or the suffix sentence is interjections, e.g., *en heng* “aha”;

ii) When the focus sentence is the beginning or the ending of a paragraph, no prefix sentence or suffix sentence can be extracted as the context. In this case, the cause may be beyond the context;

iii) The cause is obscure, which can be very abstract or even unknown reasons.

The emotion cause text: A cause is considered as a proposition. It is generally assumed that a proposition has a verb which optionally takes a noun occurring before it as the subject and a noun after it as the object. However, a cause can also be expressed as a nominal. In other words, both the predicate and the two arguments are optional provided that at least one of them is present. Thus, the fundamental issue in designing a cause detection system is the definition of the span of a cause text. As mentioned, most previous studies on causal relations choose to ignore the identification of cause candidates. In this paper, we first analyze the distribution of cause text and then determine the cause candidates for an emotion.

Based on the emotion cause corpus, we find that emotion causes are more likely to be expressed by verbal events than nominal events (85% vs. 15%). Although a nominalization (a kind of verbal events) is usually a noun phrase, a proposition containing a verb plays a salient role in the expressions of emotion causes, and thus a cause candidate are more likely to be a clause-based unit.

In addition, the actual cause can sometimes be too long and complicated, which involves several events. In order to explore the span of a cause text, we do the following analysis.

Table 1: The clause distribution of cause texts

Position	Cause (%)	Position	Cause (%)
Left_0	12.90	Right _0	15.54
Left_1	31.37	Right _1	9.55
Left_2	13.31	Right_n (n>1)	9.18
Left_n (n>2)	10.15		
Total	67.73		32.27

Table 2: The multi-clause distribution of cause text

Same clause	%	Cross-clauses	%
Left_0	16.80	Left_2_1_0	0.25
Left_1	31.82	Left_2_1	10.84
Left_2	7.33	Left_1_0	0.62
Right _0	18.97	Right_0_1	2.55
Right _1	10.59		
Total	85.75		14.25

Firstly, for each emotion keyword, an entry is segmented into clauses with four punctuations (i.e. commas, periods, question marks and exclamation marks), and thus an entry becomes a list of cause candidates. For example, when an entry has four clauses, its corresponding list of cause candidates contains five text units, i.e. <left_2, left_1, left_0, right_0, right_1>. If we assume the clause where emotion keyword locates is a focus clause, 'left_2' and 'left_1' are previous two clauses, and 'right_1' is the following one. 'left_0' and 'right_0' are the partial texts of the focus clause, which locate in the left side of and the right side of the emotion keyword, respectively. Moreover, a cause candidate must contain either a noun or a verb because a cause is either a verbal event or a nominal event; otherwise, it will be removed from the list.

Secondly, we calculate whether a cause candidate overlaps with the real cause, as shown in Table 1. We find that emotion causes are more likely to occur in the left of emotion keyword. This observation is consistent with the fact that an emotion is often trigged by an external happened event. Thirdly, for all causes occurring between 'left_2' and 'right_1', we calculate whether a cause occurs across clauses, as in Table 2. We observe that most causes locate within the same clause of the representation of the emotion (85.57%). This suggests that a clause may be the most appropriate unit to detect a cause.

4 Emotion Cause Detection Based on Multi-label Classification

A cause detection system is to identify the causal relation between a pair of two text units. For emotion cause detection, one of the two text units is fixed (i.e. the emotion keyword), and therefore the remaining two unresolved issues are the identification of the other text unit and the causal relation.

From the above data analysis, there are two observations. First, most emotion causes are verbal events, which are often expressed by a proposition (or a clause). Thus, we define another text unit as a clause, namely a cause candidate. Second, as most emotion causes occur between 'left_2' and 'right_1' (~80%), we define the cause candidates for an emotion as <left_2, left_1, left_0, right_0, right_1>.

Differing from the existing cause systems, we formalize emotion cause detection as a multi-label problem. In other words, given an emotion keyword and its context, its label is the locations of its causes, such as "left_1, left_0". This multi-label-based formalization of the cause detection task has two advantages. First, it is an integrated system detecting causes for an emotion from the contextual information. In most previous cause detection systems, a causal relation is identified based on the information between two small text units, i.e. a pair of clauses or noun phrases, and therefore it is often the case that long-distance information is missed. Second, the multi-label-based tagging is able to capture the relationship between two cause candidates. For example, "left_2" and "left_1" are often combined as a complicated event as a cause.

As a multi-label classification task, every multi-label classifier is applicable. In this study, we use a simple strategy: we treat each possible combination of labels appearing in the training data as a unique label. Note that an emotion without causes is labeled as "None". This converts multi-label classification to single-label classification, which is suitable for any multi-class classification technologies. In particular, we choose a Max Entropy tool, Mallet[1], to perform the classification.

5 Linguistic Features

As explained, there are basically two kinds of features for cause detection, namely pattern-based features and semantic-based features. In this study, we develop two sets of patterns based on linguistic analysis: one is a set of manually generalized patterns, and the other contains automatically generalized patterns. All of these patterns explore causal constructions either for general causal relations or for specific emotion cause relations.

5.1 Linguistic Cues

Based on the linguistic analysis, Lee et al. (2010a) identify six groups of linguistic cue words that are highly collocated with emotion causes, as shown in Table 3. Each group of the linguistic cues serves as an indicator marking the causes in different emotional constructions. In this paper, these groups of linguistic cues are reinterpreted from the computational perspective, and are used to develop pattern-based features for the emotion cause detection system.

Table 3: Linguistic cue words for emotion cause detection (Lee et al. 2010a)

Group	Cue Words
I: Prepositions	'for' as in 'I will do this for you': *wei4*, *wei4le* 'for' as in 'He is too old for the job': *dui4*, *dui4yu2* 'as': *yi3*
II: Conjunctions	'because': *yin1*, *yin1wei4*, *you2yu2* 'so': *yu1shi4*, *suo3yi3*, *yin1er2* 'but': *ke3shi4*
III: Light Verbs	"to make": *rang4*, *ling4*, *shi3*
IV: Reported Verbs	'to think about': *xiang3dao4*, *xiang3qi3*, *yi1xiang3*, *xiang3 lai2* 'to talk about': *shuo1dao4*, *shuo1qi3*, *yi1shuo1*, *jiang3dao4*, *jiang3qi3*, *yi1jiang3*, *tan2dao4*, *tan2qi3*, *yi1tan2*, *ti2dao4*, *ti2qi3*, *yi1ti2*
V: Epistemic Markers	'to hear': *ting1*, *ting1dao4*, *ting1shuo1* 'to see': *kan4*, *kan4dao4*, *kan4jian4*, *jian4dao4*, *jian4*, *yan3kan4*, *qiao2jian4* 'to know': *zhi1dao4*, *de2zhi1*, *de2xi1*, *huo4zhi1*, *huo4xi1*, *fa1xian4*, *fa1jue2* 'to exist': *you3*
VI: Others	'is': *deshi4* 'say': *deshuo1* 'at': *yu2* 'can': *neng2*

For emotion cause processing, Group I and II contain cues which are for general cause detection, and while Group III, IV and V include cues specifically for emotion cause detection. Group VI includes other linguistic cues that do not fall into any of the five groups.

Group I covers some prepositions which all roughly mean 'for', and Group II contains the conjunctions that explicitly mark the emotion cause. Group I is expected to capture the prepositions constructions in the focus clause where the emotion keyword locates. Group II tends to capture the rhetorical relation expressed by conjunction words so as to infer causal relation among multi-clauses. These two groups are typical features for general cause detection.

Group III includes three common light verbs which correspond to the English equivalents "to make" or "to cause". Although these light verbs themselves do not convey any concrete meaning, they are often associated with several constructions to express emotions and at the same time indicate the position of emotion causes. For example, "*The birthday party made her happy*".

One apparent difference between emotion causes and general causes is that emotions are often triggered by human activities or the perception of such activities, e.g., "*glad to say*" or "*glad to hear*". Those human activities are often strong indicators for the location of emotion

[1] http://mallet.cs.umass.edu/

causes. Group IV and V are used to capture this kind of information. Group IV is a list of verbs of thinking and talking, and Group V includes four types of epistemic markers which are usually verbs marking the cognitive awareness of emotions in the complement position. The epistemic markers include verbs of seeing, hearing, knowing, and existing.

5.2 Linguistic Patterns

With the six groups of linguistic cues, we generalize 14 rules used in Lee et al. (2010b) to locate the clause positions of an emotion cause, as shown in Table 4. The abbreviations used in the rules are given as follows:

C = Cause
K = Emotion keyword
B = Clauses before the focus clause
F = Focus clause/the clause containing the emotion verb
A = Clauses after the focus clause

Table 4: Linguistic rules for emotion cause detection (Lee et al. 2010b)

No.	Rules
1	i) C(B/F) + III(F) + K(F) ii) C = the nearest N/V before I in F/B
2	i) IV/V/I/II(B/F) + C(B/F) + K(F) ii) C = the nearest N/V before K in F
3	i) I/II/IV/V (B) + C(B) + K(F) ii) C = the nearest N/V after I/II/IV/V in B
4	i) K(F) + V/VI(F) + C(F/A) ii) C = the nearest N/V after V/VI in F/A
5	i) K(F)+II(A)+C(A) ii) C = the nearest N/V after II in A
6	i) III(F) + K(F) + C(F/A) ii) C = the nearest N/V after K in F or A
7	i) *yue4* C *yue4* K "the more C the more K" (F) ii) C = the V in between the two *yue4*'s in F
8	i) K(F) + C(F) ii) C = the nearest N/V after K in F
9	i) V(F) + K(F) ii) C = V+(an aspectual marker) in F
10	i) K(F) + *de* "possession"(F) + C(F) ii) C = the nearest N/V +的+N after *de* in F
12	i) K(B) + IV (B) + C(F) ii) C = the nearest N/V after IV in F
13	i) IV(B) + C(B) + K(F) ii) C = the nearest N/V after IV in B
14	i) C(B) + K(F) ii) C = the nearest N/V before K in B

For illustration, an example of the rule description is given in Rule 1.

Rule 1:
i) C(B/F) + III(F) + K(F)
ii) C = the nearest N/V before III in F/B

Rule 1 indicates that the cause (C) comes before Group III cue words. Theoretically, in identifying C, we look for the nearest verb/noun occurring before Group III cue words in the focus clause (F) or the clauses before the focus clause (B), and consider the clause containing this verb/noun as a cause. Practically, for each cause candidate, i.e. 'left_1', if it contains this verb/noun, we create a feature with "left_1_rule_1=1".

5.3 Generalized Patterns

Rule-based patterns usually achieve a rather high accuracy, but suffer from low coverage. To avoid this shortcoming, we extract a generalized feature automatically according to the rules in Table 4. The features are able to detect two kinds of constructions, namely functional constructions, i.e. rhetorical constructions, and specific constructions for emotion causes.

Local functional constructions: a cause occurring in the focus clause is often expressed with certain functional words, such as "*because of*", "*due to*". In order to capture the various expressions of these functional constructions, we identify all functional words around the given emotion keyword. For an emotion keyword, we search 'left_0' from the right until a noun or a verb is found. Next, all unigrams and bigrams between the noun or the verb and the emotion keyword are extracted. The same applies to 'right_0'.

Long-distance conjunction constructions: Group II enumerates only some typical conjunction words. To capture more general rhetorical relations, according to the given POS tags, the conjunction word is extracted for each cause candidate, if it occurs at the beginning of the candidate.

Generalized action and epistemic verbs: Group IV and V cover only partial action and epistemic verbs. To capture possible related expressions, we take the advantage of Chinese characters. In Chinese, each character itself usually has a meaning and some characters have a strong capability to create words with extended meaning. For example, the character "*ting1*-listen" combines with other characters to create words expressing “listening”, such as *ting1jian4*, *ting1wen5*. With the selected characters regarding reported verbs and epistemic markers, each cause candidate is checked to see whether it contains the predefined characters.

6 Experiments

For the emotion cause corpus, we reserve 80% as the training data, 10% as the development data, and 10% as the test data. During evaluation, we first convert the multi-label tag outputted from our system into a binary tag (‘Y’ means the presence of a causal relation; ‘N’ indicates the absence of a causal relation) between the emotion keyword and each candidate in its corresponding cause candidates. Thus, the evaluation scores for binary classification based on three common measures, i.e. precision, recall and F-score, are chosen.

6.1 Linguistic Feature Analysis

According to the distribution in Table 1, we design a naive baseline to allow feature analysis. The baseline searches for the cause candidates in the order of <left_1, right_0, left_2, left_0, right_1>. If the candidate contains a noun or verb, consider this clause as a cause and stop.

We run the multi-label system with different groups of features and the performances are shown in Table 5. The feature set begins with linguistic patterns (LP), and is then incorporated with local functional constructions (LFC), long-distance conjunction constructions (LCC), and generalized action and epistemic verbs (GAE), one by one. Since the ‘N’ tag is overwhelming, we report only the Mac average scores for both ‘Y’ and ‘N’ tags.

In Table 5, we first notice that the performances achieve significant improvement from the baseline to the final system (~17%). This indicates that our linguistic features are effective for emotion cause detection. In addition, we observe that LP and LFC are the best two effective features, whereas LCC and GAE have slight contributions. This shows that our feature extraction has a strong capability to detect local causal constructions, and is yet unable to detect the long-distance or semantic causal information. Here, ‘local’ refers to the information in the focus clause. We also find that incorporating LFC, which is a pure local feature, generally improves the performances of all cause candidates, i.e. ~5% improvement for ‘left_1’. This indicates that our multi-label integrated system is able to convey information among cause candidates.

Table 5: The overall performance with different feature sets of the multi-label system

	Precision	Recall	F-score
Baseline	56.64	57.70	56.96
LP	74.92	66.70	69.21
+ LFC	72.80	71.94	72.35
+ LCC	73.60	72.50	73.02
+ GAE	73.90	72.70	73.26

Table 6: The separate performances for ‘Y’ and ‘N’ tags of the multi-label system

	‘Y’	‘N’
Baseline	33.06	80.85
LP	48.32	90.11
+ LFC	55.45	89.24
+ LCC	56.48	89.57
+ GPE	56.84	89.68

Table 6 shows the performances (F-scores) for ‘Y’ and ‘N’ tags separately. First, we notice that the performances of the ‘N’ tag are much better than the ones of ‘Y’ tag. Second, it is surprising that incorporating the linguistic features significantly improves only the ‘Y’ tag (from 33% to 56%), but does not affect ‘N’ tag. This suggests that our linguistic features are effective to detect the presence of causal relation, and yet do not hurt the detection of ‘non_causal’ relation. For the ‘Y’ tag, the features LP and LFC achieve ~15% and ~7% improvements respectively. LCC and GPE, on the other hand, show slight improvements only.

Finally, Table 7 shows the detailed performances of our multi-label system with all features. The last row shows the overall performances of ‘Y’ and ‘N’ tags. For the ‘Y’ tag, the closer the cause candidates are to the emotion keyword, the better performances the system achieves. This proves that the features we propose effectively detect local emotion causes, more effort,

Table 7: The detailed performance for the multi-label system including all features

'Y' tag	Precision	Recall	F-score	'N' tag	Precision	Recall	F-score
Left_0	68.92	68.92	68.92	Left_0	93.72	93.72	93.72
Left_1	57.63	63.35	60.36	Left_1	82.90	79.22	81.02
Left_2	29.27	20.69	24.24	Left_2	89.23	92.93	91.04
Right_0	67.78	64.89	66.30	Right_0	82.63	84.41	83.51
Right_1	54.84	30.91	39.54	Right_1	92.00	96.90	94.38
Total	58.84	54.98	56.84	Total	88.96	90.42	89.68

Table 8: The detailed performance for the single-label system including all features

'Y' tag	Precision	Recall	F-score	'N' tag	Precision	Recall	F-score
Left_0	65.39	68.92	67.11	Left_0	93.65	92.62	93.13
Left_1	61.19	50.93	55.59	Left_1	79.64	85.60	82.51
Left_2	28.57	20.69	24.00	Left_2	89.20	92.68	90.91
Right_0	70.13	57.45	63.16	Right_0	80.30	87.63	83.81
Right_1	33.33	40.00	36.36	Right_1	92.50	90.24	91.36
Total	55.67	50.00	52.68	Total	87.85	90.08	88.95

however, should be put on the detection of long-distance causes. In addition, we find that the detection of long-distance causes usually relies on two kinds of information for inference: rhetorical relation and deep semantic information.

6.2 Modeling Analysis

To compare our multi-label model with single-label models, we create a single-label system as follows. The single-label model is a binary classification for a pair comprising the emotion keyword and a candidate in its corresponding cause candidates. For each pair, all linguistic features are extracted only from the focus clause and its corresponding cause candidate. Note that we only use the features in the focus clause for "left_0" and "right_0". The performances are shown in Table 8.

Comparing Tables 7 and 8, all F-scores of the 'Y' tag increase and the performances of the 'N' tag remain almost the same for both the single-label model and our multi-label model. We also find that the multi-label model takes more advantage of local information, and improves the performances, particularly for "left_1".

To take an in-depth analysis of the cause detection capability of the multi-label model, an evaluation is designed that the label is treated as a tag from the multi-label classifier. Due to the tag sparseness problem (as in Table 2), only the "left_2, left_1" tag is detected in the test data, and its performance is 21% precision, 26% recall and 23% F-score. Furthermore, we notice that ~18% of the "left_1" tags are detected through this combination tag. This shows that some causes need to take into account the mutual information between clauses. Although the scores are low, it still shows that our multi-label model provides an effective way of detecting some of the multi-clauses causes.

7 Conclusion

We treat emotion cause detection as a multi-label task, and develop two sets of linguistic features for emotion cause detection based on linguistic cues. The experiments on the small-scale corpus show that both the multi-label model and the linguistic features are able to effectively detect emotion causes. The automatic detection of emotion cause will in turn allow us to extract directly relevant information for public opinion mining and event prediction. It can also be used to improve emotion detection and classification. In the future, we will attempt to improve our system from two aspects. On the one hand, we will explore more powerful multi-label classification models for our system. On the other hand, we will investigate more linguistic patterns or semantic information to further help emotion cause detection.

References

Abbasi, A., H. Chen, S. Thoms, and T. Fu. 2008. Affect Analysis of Web Forums and Blogs using Correlation Ensembles". In *IEEE Tran. Knowledge and Data Engineering*, vol. 20(9), pp. 1168-1180.

Bethard, S. and J. Martin. 2008. Learning Semantic Links from a Corpus of Parallel Temporal and Causal Relations. In *Proceedings of ACL.*

Descartes, R. 1649. The Passions of the Soul. In J. Cottingham et al. (Eds), *The Philosophical Writings of Descartes*. Vol. 1: 325-404.

Chang, D.-S. and K.-S. Choi. 2006. Incremental cue phrase learning and bootstrapping method for causality extraction using cue phrase and word pair probabilities. *Information Processing and Management*. 42(3): 662-678.

Chen, Y., S. Y. M. Lee and C.-R. Huang. 2009. Are Emotions Enumerable or Decomposable? And Its Implications for Emotion Processing. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation*.

Girju, R. 2003. Automatic Detection of Causal Relations for Question Answering. In *the 41st Annual Meeting of the Association for Computational Linguistics, Workshop on Multilingual Summarization and Question Answering - Machine Learning and Beyond*, Sapporo, Japan.

James, W. 1884. What is an Emotion? *Mind*, 9(34):188–205.

Lee, S. Y. M., Y. Chen and C.-R. Huang. 2010a. A Text-driven Rule-based System for Emotion Cause Detection. In *Proceedings of NAACL-HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*.

Lee, S. Y. M., Y. Chen, S. Li and C.-R. Huang. 2010b. Emotion Cause Events: Corpus Construction and Analysis. In *Proceedings of LREC 2010.*

Low, B. T., K. Chan , L. L. Choi , M. Y. Chin , S. L. Lay. 2001. Semantic Expectation-Based Causation Knowledge Extraction: A Study on Hong Kong Stock Movement Analysis, In *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, p.114-123, April 16-18.

Marcu, D., and A. Echihabi. 2002. An Unsupervised Approach to Recognizing Discourse Relations. In *Proceedings of ACL.*

Mihalcea, R. and H. Liu. 2006. A Corpus-based Approach to Finding Happiness. In *Proceedings of the AAAI Spring Symposium on Computational Approaches to Weblogs*.

Persing, I. and V. Ng. 2009. Semi-Supervised Cause Identification from Aviation Safety Reports. In *Proceedings of ACL.*

Plutchik, R. 1980. *Emotions: A Psychoevolutionary Synthesis*. New York: Harper & Row.

Strapparava, C. and R. Mihalcea. 2008. Learning to Identify Emotions in Text. In *Proceedings of the ACM Conference on Applied Computing ACM-SAC*.

Tokuhisa, R., K. Inui, and Y. Matsumoto. 2008. Emotion recognition Using Massive Examples Extracted from the Web. In *Proceedings of COLING.*

Wierzbicka, A. 1999. *Emotions across Languages and Cultures: Diversity and Universals*. Cambridge: Cambridge University Press.

A Twin-Candidate Based Approach for Event Pronoun Resolution using Composite Kernel

Chen Bin[1] **Su Jian**[2] **Tan Chew Lim**[1]

[1]National University of Singapore
{chenbin,tancl}@comp.nus.edu.sg

[2]Institute for Inforcomm Research, A-STAR
sujian@i2r.a-star.edu.sg

Abstract

Event Anaphora Resolution is an important task for cascaded event template extraction and other NLP study. In this paper, we provide a first systematic study of resolving pronouns to their event verb antecedents for general purpose. First, we explore various positional, lexical and syntactic features useful for the event pronoun resolution. We further explore tree kernel to model structural information embedded in syntactic parses. A composite kernel is then used to combine the above diverse information. In addition, we employed a twin-candidate based preferences learning model to capture the pair wise candidates' preference knowledge. Besides we also look into the incorporation of the negative training instances with anaphoric pronouns whose antecedents are not verbs. Although these negative training instances are not used in previous study on anaphora resolution, our study shows that they are very useful for the final resolution through random sampling strategy. Our experiments demonstrate that it's meaningful to keep certain training data as development data to help SVM select a more accurate hyper plane which provides significant improvement over the default setting with all training data.

1 Introduction

Anaphora resolution, the task of resolving a given text expression to its referred expression in prior texts, is important for intelligent text processing systems. Most previous works on anaphora resolution mainly aims at object anaphora in which both the anaphor and its antecedent are mentions of the same real world objects

In contrast, an event anaphora as first defined in (Asher, 1993) is an anaphoric reference to an event, fact, and proposition which is representative of eventuality and abstract entity. Consider the following example:

This was an all-white, all-Christian community that all the sudden was taken over -- not taken over, that's a very bad choice of words, but **[*invaded*]$_1$** *by, perhaps different groups.*
[*It*]$_2$ *began when a Hasidic Jewish family bought one of the town's two meat-packing plants 13 years ago.*

The anaphor **[*It*]$_2$** in the above example refers back to an event, "all-white and all-Christian city of Postville is diluted by different ethnic groups." Here, we take the main verb of the event, **[*invaded*]$_1$** as the representation of this event and the antecedent for pronoun **[*It*]$_2$**.

According to (Asher, 1993), antecedents of event pronoun include both gerunds (e.g. destruction) and inflectional verbs (e.g. destroying). In our study, we focus on the inflectional verb representation, as the gerund representation is studied in the conventional anaphora resolution. For the rest of this paper, "event pronouns" are pronouns whose antecedents are event verbs while "non-event anaphoric pronouns" are those with antecedents other than event verbs.

Entity anaphora resolution provides critical links for cascaded event template extraction. It also provides useful information for further inference needed in other natural language processing tasks such as discourse relation and entailment. Event anaphora (both pronouns and noun phrases) contributes a significant proportion in anaphora corpora, such as OntoNotes. 19.97% of its total number of entity chains contains event verb mentions.

In (Asher, 1993) chapter 6, a method to resolve references to abstract entities using discourse representation theory is discussed. However, no computation system was proposed for entity anaphora resolution. (Byron, 2002) proposed semantic filtering as a complement to salience calculations to resolve event pronoun targeted by us. This knowledge deep approach only

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 188–196,
Beijing, August 2010

works for much focused domain like trains spoken dialogue with handcraft knowledge of relevant events for only limited number of verbs involved. Clearly, this approach is not suitable for general event pronoun resolution say in news articles. Besides, there's also no specific performance report on event pronoun resolution, thus it's not clear how effective their approach is. (Müller, 2007) proposed pronoun resolution system using a set of hand-crafted constraints such as "argumenthood" and "right-frontier condition" together with logistic regression model based on corpus counts. The event pronouns are resolved together with object pronouns. This explorative work produced an 11.94% F-score for event pronoun resolution which demonstrated the difficulty of event anaphora resolution. In (Pradhan, *et.al*, 2007), a general anaphora resolution system is applied to OntoNotes corpus. However, their set of features is designed for object anaphora resolution. There is no specific performance reported on event anaphora. We suspect the event pronouns are not correctly resolved in general as most of these features are irrelevant to event pronoun resolution.

In this paper, we provide the first systematic study on pronominal references to event antecedents. First, we explore various positional, lexical and syntactic features useful for event pronoun resolution, which turns out quite different from conventional pronoun resolution except sentence distance information. These have been used together with syntactic structural information using a composite kernel. Furthermore, we also consider candidates' preferences information using twin-candidate model.

Besides we further look into the incorporation of negative instances from non-event anaphoric pronoun, although these instances are not used in previous study on co-reference or anaphora resolution as they make training instances extremely unbalanced. Our study shows that they can be very useful for the final resolution after random sampling strategy.

We further demonstrate that it's meaningful to keep certain training data as development data to help SVM select a more accurate hyper-plane which provide significant improvement over the default setting with all training data.

The rest of this paper is organized as follows. Section 2 introduces the framework for event pronoun resolution, the considerations on training instance, the various features useful for event pronoun resolution and SVM classifier with adjustment of hyper-plane. Twin-candidate model is further introduced to capture the preferences among candidates. Section 3 presents in details the structural syntactic feature and the kernel functions to incorporate such a feature in the resolution. Section 4 presents the experiment results and some discussion. Section 5 concludes the paper.

2 The Resolution Framework

Our event-anaphora resolution system adopts the common learning-based model for object anaphora resolution, as employed by (Soon *et al.*, 2001) and (Ng and Cardie, 2002a).

2.1 Training and Testing instance

In the learning framework, training or testing instance of the resolution system has a form of $fv(candi_i, ana)$ where $candi_i$ is the i^{th} candidate of the antecedent of anaphor ana. An instance is labeled as positive if $candi_i$ is the antecedent of ana, or negative if $candi_i$ is not the antecedent of ana. An instance is associated with a feature vector which records different properties and relations between ana and $candi_i$. The features used in our system will be discussed later in this paper.

During training, for each event pronoun, we consider the preceding verbs in its current and previous two sentences as its antecedent candidates. A positive instance is formed by pairing an anaphor with its correct antecedent. And a set of negative instances is formed by pairing an anaphor with its candidates other than the correct antecedent. In addition, more negative instances are generated from non-event anaphoric pronouns. Such an instance is created by pairing up a non-event anaphoric pronoun with each of the verbs within the pronoun's sentence and previous two sentences. This set of instances from non-event anaphoric pronouns is employed to provide extra power on ruling out non-event anaphoric pronouns during resolution. This is inspired by the fact that event pronouns are only 14.7% of all the pronouns in the OntoNotes corpus. Based on these generated training instances, we can train a binary classifier using any discriminative learning algorithm.

The natural distribution of textual data is often imbalanced. Classes with fewer examples are under-represented and classifiers often perform far below satisfactory. In our study, this becomes a significant issue as positive class (event anaphoric) is the minority class in pronoun resolution task. Thus we utilize a random down sampling method to reduce majority class samples to an equivalent level with the minority class samples which is described in (Kubat and Matwin, 1997) and (Estabrooks *et al*, 2004). In (Ng and Cardie, 2002b), they proposed a negative sample selection scheme which included only negative instances found in between an anaphor and its antecedent. However, in our event pronoun resolution, we are distinguishing the event-anaphoric from non-event anaphoric which is different from (Ng and Cardie, 2002b).

2.2 Feature Space

In a conventional pronoun resolution, a set of syntactic and semantic knowledge has been reported as in (Strube and Müller, 2003; Yang *et al*, 2004;2005a;2006). These features include number agreement, gender agreement and many others. However, most of these features are not useful for our task, as our antecedents are inflectional verbs instead of noun phrases. Thus we have conducted a study on effectiveness of potential positional, lexical and syntactic features. The lexical knowledge is mainly collected from corpus statistics. The syntactic features are mainly from intuitions. These features are purposely engineered to be highly correlated with positive instances. Therefore such kind of features will contribute to a high precision classifier.

- **Sentence Distance**

This feature measures the sentence distance between an anaphor and its antecedent candidate under the assumptions that a candidate in the closer sentence to the anaphor is preferred to be the antecedent.

- **Word Distance**

This feature measures the word distance between an anaphor and its antecedent candidate. It is mainly to distinguish verbs from the same sentence.

- **Surrounding Words and POS Tags**

The intuition behind this set of features is to find potential surface words that occur most frequently with the positive instances. Since most of verbs occurred in front of pronoun, we have built a frequency table from the preceding 5 words of the verb to succeeding 5 surface words of the pronoun. After the frequency table is built, we select those words with confidence[1] > 70% as features. Similar to Surrounding Words, we have built a frequency table to select indicative surrounding POS tags which occurs most frequently with positive instances.

- **Co-occurrences of Surrounding Words**

The intuition behind this set of features is to capture potential surface patterns such as "*It caused…*" and *"It leads to"*. These patterns are associated with strong indication that pronoun "*it*" is an event pronoun. The range for the co-occurrences is from preceding 5 words to succeeding 5 words. All possible combinations of word positions are used for a co-occurrence words pattern. For example "*it leads to*" will generate a pattern as "***S1_S2_lead_to***" where ***S1*** and ***S2*** mean succeeding position 1 and 2. Similar to previous surrounding words, we will conduct corpus statistics analysis and select co-occurrence patterns with a confidence greater than 70%. Following the same process, we have examined co-occurrence patterns for surrounding POS tags.

- **Subject/Object Features**

This set of features aims to capture the relative position of the pronoun in a sentence. It denotes the preference of pronoun's position at the clause level. There are 4 features in this category as listed below.

Subject of Main Clause

This feature indicates whether a pronoun is at the subject position of a main clause.

Subject of Sub-clause

This feature indicates whether a pronoun is at the subject position of a sub-clause.

Object of Main Clause

This feature indicates whether a pronoun is at the object position of a main clause.

Object of Sub-clause

This feature indicates whether a pronoun is at the object position of a sub-clause.

- **Verb of Main/Sub Clause**

Similar to the Subject/Object features of pronoun, the following two features capture the rela-

[1] $Confidence = \frac{\#\ of\ word_i occurred\ with\ positive\ instance}{\#\ of\ word_i occurrences}$

tive position of a verb in a sentence. It encodes the preference of verb position between main verbs in main/sub clauses.

Main Verb in Main Clause

This feature indicates whether a verb is a main verb in a main clause.

Main Verb in Sub-clause

This feature indicates whether a verb is a main verb in a sub-clause.

2.3 Support Vector Machine

In theory, any discriminative learning algorithm is applicable to learn a classifier for pronoun resolution. In our study, we use Support Vector Machine (Vapnik, 1995) to allow the use of kernels to incorporate the structure feature. One advantage of SVM is that we can use tree kernel approach to capture syntactic parse tree information in a particular high-dimension space.

Suppose a training set S consists of labeled vectors $\{(x_i, y_i)\}$, where x_i is the feature vector of a training instance and y_i is its class label. The classifier learned by SVM is:

$$f(x) = sign\left(\sum_{i=1} y_i a_i x \cdot x_i + b\right)$$

where a_i is the learned parameter for a support vector x_i. An instance x is classified as positive if $f(x) \geq 0$. Otherwise, x is negative.

- **Adjust Hyper-plane with Development Data**

Previous works on pronoun resolution such as (Yang *et al*, 2006) used the default setting for hyper-plane which sets $\mathrm{f(x)} = 0$. And an instance is positive if $f(x) \geq 0$ and negative otherwise. In our study, we look into a method of adjusting the hyper-plane's position using development data to improve the classifier's performance.

Considering a default model setting for SVM as shown in Figure 2(for illustration purpose, we use a 2-D example).

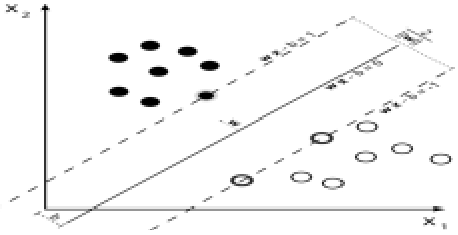


Figure 2: 2-D SVM Illustration

The objective of SVM learning process is to find a set of weight vector w which maximizes the margin (defined as $\frac{2}{\|w\|}$) with constraints defined by support vectors. The separating hyper-plane is given by $w \cdot x + b = 0$ as bold line in the center. The margin is the region between the two dotted lines (bounded by $w \cdot x + b = 1$ and $w \cdot x + b = -1$). The margin is a space without any information from training instances. The actual hyper-plane may fall in any place within the margin. It does not necessarily occur in the. However, the hyper-plane is used to separate positive and negative instances during classification process without consideration of the margin. Thus if an instance falls in the margin, SVM can only decide class label from hyper-plane which may cause misclassification in the margin.

Based on the previous discussion, we propose an adjustment of the hyper-plane using development data. For simplicity, we adjust the hyper-plane function value instead of modeling the function itself. The hyper-plane function value will be further referred as a threshold θ. The following is a modified version of a learned SVM classifier.

$$f(x, \theta) = \begin{cases} 1 & \text{if} \left(\sum_{i=1} y_i a_i x \cdot x_i + b\right) \geq \theta \\ -1 & \text{if} \left(\sum_{i=1} y_i a_i x \cdot x_i + b\right) < \theta \end{cases}$$

where θ is the threshold, a_i is the learned parameter for a feature x_i and y_i is its class label. A set of development data is used to adjust the hyper-plane function threshold θ in order to maximize the accuracy of the learned SVM classifier on development data. The adjustment of hyper-plane is defined as:

$$\theta_{best} = argmax_{\theta \in \Theta}\left(\sum_{x \in X} I(y, f(x, \theta))\right)$$

where $I(y, f)$ is an indicator function which output 1 if $f(x, \theta)$is same sign as y and 0 otherwise. Thereafter, the learned threshold θ is applied to the testing set.

3 Incorporating Structural Syntactic Information

A parse tree that covers a pronoun and its antecedent candidate could provide us much syntactic information related to the pair which is explicitly or implicitly represented in the tree. Therefore, by comparing the common sub-structures between two trees we can find out to what degree two trees contain similar syntactic information, which can be done using a convolution tree kernel. The value returned from tree kernel reflects similarity between two instances in syntax. Such syntactic similarity can be further combined with other knowledge to compute overall similarity between two instances, through a composite kernel. Normally, parsing is done at sentence level. However, in many cases a pronoun and its antecedent candidate do not occur in the same sentence. To present their syntactic properties and relations in a single tree structure, we construct a syntax tree for an entire text, by attaching the parse trees of all its sentences to an upper node. Having obtained the parse tree of a text, we shall consider how to select the appropriate portion of the tree as the structured feature for a given instance. As each instance is related to a pronoun and a candidate, the structured feature at least should be able to cover both of these two expressions.

3.1 Structural Syntactic Feature

Generally, the more substructure of the tree is included, the more syntactic information would be provided, but at the same time the more noisy information that comes from parsing errors would likely be introduced. In our study, we examine three possible structured features that contain different substructures of the parse tree:

- **Minimum Expansion Tree**

This feature records the minimal structure covering both pronoun and its candidate in parse tree. It only includes the nodes occurring in the shortest path connecting the pronoun and its candidate, via the nearest commonly commanding node. When the pronoun and candidate are from different sentences, we will find a path through pseudo "TOP" node which links all the parse trees. Considering the example given in section 1,

This was an all-white, all-Christian community that all the sudden was taken over -- not taken over, that's a very bad choice of words, but **[*invaded*]$_1$** *by, perhaps different groups.*

[*It*]$_2$ *began when a Hasidic Jewish family bought one of the town's two meat-packing plants 13 years ago.*

The minimum expansion structural feature of the instance {***invaded, it***} is annotated with bold lines and shaded nodes in figure 1.

- **Simple Expansion Tree**

Minimum-Expansion could, to some degree, describe the syntactic relationships between the candidate and pronoun. However, it is incapable of capturing the syntactic properties of the candidate or the pronoun, because the tree structure surrounding the expression is not taken into consideration. To incorporate such information, feature Simple-Expansion not only contains all the nodes in Minimum-Expansion, but also includes the first-level children of these nodes[2] except the punctuations. The simple-expansion structural feature of instance **{*invaded, it*}** is annotated in figure 2. In the left sentence's tree, the node "NP" for "*perhaps different groups*" is terminated to provide a clue that we have a noun phrase at the object position of the candidate verb.

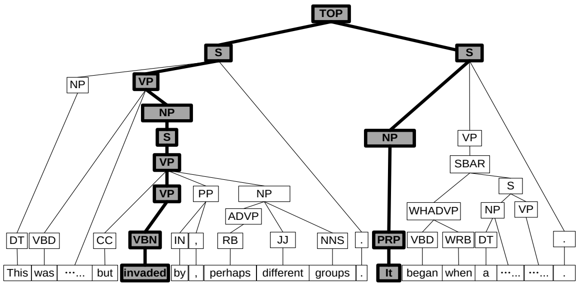


Figure 1: Minimum-Expansion Tree

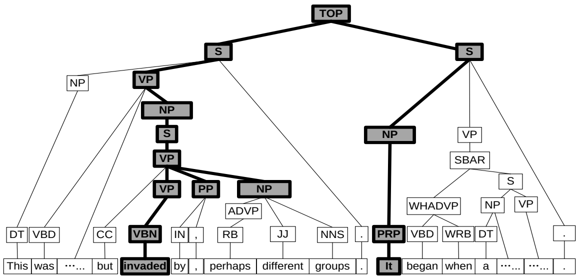


Figure 2: Simple Expansion Tree

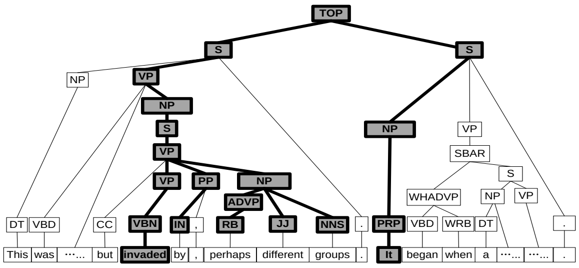


Figure 3: Full-Expansion Tree

- **Full Expansion Tree**

This feature focuses on the whole tree structure between the candidate and pronoun. It not only includes all the nodes in Simple-Expansion, but also the nodes (beneath the nearest commanding parent) that cover the words between the candi-

[2] If the pronoun and the candidate are not in the same sentence, we will not include the nodes denoting the sentences before the candidate or after the pronoun.

date and the pronoun[3]. Such a feature keeps the most information related to the pronoun and candidate pair. Figure 3 shows the structure for feature full-expansion for instance {***invaded, it***}. As illustrated, the "NP" node for "*perhaps different groups*" is further expanded to the POS level. All its child nodes are included in the full-expansion tree except the surface words.

3.2 Convolution Parse Tree Kernel and Composite Kernel

To calculate the similarity between two structured features, we use the convolution tree kernel that is defined by Collins and Duffy (2002) and Moschitti (2004). Given two trees, the kernel will enumerate all their sub-trees and use the number of common sub-trees as the measure of similarity between two trees. The above tree kernel only aims for the structured feature. We also need a composite kernel to combine the structured feature and the flat features from section 2.2. In our study we define the composite kernel as follows:

$$K_{comp}(x_1, x_2) = \frac{K_{tree}(x_1, x_2)}{|K_{tree}(x_1, x_2)|} + \frac{K_{flat}(x_1, x_2)}{|K_{flat}(x_1, x_2)|}$$

where K_{tree} is the convolution tree kernel defined for the structured feature, and K_{flat} is the kernel applied on the flat features. Both kernels are divided by their respective length[4] for normalization. The new composite kernel K_{comp}, defined as the sum of normalized K_{tree} and K_{flat}, will return a value close to 1 only if both the structured features and the flat features have high similarity under their respective kernels.

3.3 Twin-Candidate Framework using Ranking SVM Model

In a ranking SVM kernel as described in (Moschitti *et al, 2006*) for Semantic Role Labeling, two argument annotations (as argument trees) are presented to the ranking SVM model to decide which one is better. In our case, we present two syntactic trees from two candidates to the ranking SVM model. The idea is inspired by (Yang, *et.al,* 2005b;2008). The intuition behind the twin-candidate model is to capture the information of how much one candidate is more preferred than another. The candidate wins most of the pair wise comparisons is selected as antecedent.

The feature vector for each training instance has a form of $fv = (candi_i, candi_j)$. An instance is positive if $cand_i$ is a better antecedent choice than $candi_j$. Otherwise, it is a negative instance. For each feature vector, both tree structural features and flat features are used. Thus each feature vector has a form of $fv = (t_i, t_j, v_i, v_j)$ where t_i and t_j are trees of candidate i and j respectively, v_i and v_j are flat feature vectors of candidate i and j respectively.

In the training instances generation, we only generate those instances with one candidate is the correct antecedent. This follows the same strategy used in (Yang *et al*, 2008) for object anaphora resolution.

In the resolution process, a list of m candidates is extracted from a three sentences window. A total of $\binom{m}{2}$ instances are generated by pairing-up the m candidates pair-wisely. We used a Round-Robin scoring scheme for antecedent selection. Suppose a SVM output for an instance $fv = (candi_i, candi_j)$ is 1, we will give a score 1 for $candi_i$ and -1 for $candi_j$ and vice versa. At last, the candidate with the highest score is selected as antecedent. In order to handle a non-event anaphoric pronoun, we have set a threshold to distinguish event anaphoric from non-event anaphoric. A pronoun is considered as event anaphoric if its score is above the threshold. In our experiments, we kept a set of development data to find out the threshold in an empirical way.

4 Experiments and Discussions

4.1 Experimental Setup

OntoNotes Release 2.0 English corpus as in (Hovy *et al*, 2006) is used in our study, which contains 300k words of English newswire data (from the Wall Street Journal) and 200k words of English broadcast news data (from ABC, CNN, NBC, Public Radio International and Voice of America). Table 1 shows the distribution of various entities. We focused on the resolution of 502 event pronouns encountered in the corpus. The resolution system has to handle both the event pronoun identification and antecedent selection tasks. To illustrate the difficulty of event pronoun resolution, 14.7% of all pronoun mentions are event anaphoric and only 31.5% of

[3] We will not expand the nodes denoting the sentences other than where the pronoun and the candidate occur.

[4] The length of a kernel K is defined as $|K(x_1, x_2)| = \sqrt{K(x_1, x_1) \cdot K(x_2, x_2)}$

event pronoun can be resolved using “most recent verb” heuristics. Therefore a most-recent-verb baseline will yield an f-score 4.63%.

To conduct event pronoun resolution, an input raw text was preprocessed automatically by a pipeline of NLP components. The noun phrase identification and the predicate-argument extraction were done based on Stanford Parser (Klein and Manning, 2003a;b) with F-score of 86.32% on Penn Treebank corpus.

<table>
<tr><td colspan="3"><b>Non-Event Anaphora:</b> <i>4952 80.03%</i></td></tr>
<tr><td rowspan="4"><b>Event Anaphora:</b>
<i>1235</i>
<i>19.97%</i></td><td colspan="2"><b>Event NP:</b> <i>733 59.35%</i></td></tr>
<tr><td rowspan="3"><b>Event Pronoun:</b>
<i>502 40.65%</i></td><td><b>It:</b> <i>29.0%</i></td></tr>
<tr><td><b>This:</b> <i>16.9%</i></td></tr>
<tr><td><b>That:</b> <i>54.1%</i></td></tr>
</table>

Table 1: The distribution of various types of 6187 anaphora in OntoNotes 2.0

For each pronoun encountered during resolution, all the inflectional verbs within the current and previous two sentences are taken as candidates. For the current sentence, we take only those verbs in front of the pronoun. On average, each event pronoun has 6.93 candidates. Non-event anaphoric pronouns will generate 7.3 negative instances on average.

4.2 Experiment Results and Discussion

In this section, we will present our experimental results with discussions. The performance measures we used are precision, recall and F-score. All the experiments are done with a 10-folds cross validation. In each fold of experiments, the whole corpus is divided into 10 equal sized portions. One of them is selected as testing corpus while the remaining 9 are used for training. In experiments with development data, 1 of the 9 training portions is kept for development purpose. In case of statistical significance test for differences is needed, a two-tailed, paired-sample Student’s t-Test is performed at 0.05 level of significance.

In the first set of experiments, we are aiming to investigate the effectiveness of each single knowledge source. Table 2 reports the performance of each individual experiment. The flat feature set yields a baseline system with 40.6% f-score. By using each tree structure along, we can only achieve a performance of 44.4% f-score using the minimum-expansion tree. Therefore, we will further investigate the different ways of combining flat and syntactic structure knowledge to improve resolution performances.

	Precision	**Recall**	**F-score**
Flat	0.406	0.406	0.406
Min-Exp	0.355	0.596	0.444
Simple-Exp	0.347	0.512	0.414
Full-Exp	0.323	0.476	0.385

Table 2: Contribution from Single Knowledge Source

The second set of experiments is conducted to verify the performances of various tree structures combined with flat features. The performances are reported in table 3. Each experiment is reported with two performances. The upper one is done with default hyper-plane setting. The lower one is done using the hyper-plane adjustment as we discussed in section 2.3.

	Precision	**Recall**	**F-score**
Min-Exp + Flat	0.433	0.512	0.469
	(0.727)	*(0.446)*	*(0.553)*
Simple-Exp +Flat	0.423	0.534	0.472
	(0.652)	*(0.492)*	*(0.561)*
Full-Exp + Flat	0.416	0.526	0.465
	(0.638)	*(0.496)*	*(0.558)*

Table 3: Comparison of Different Tree Structure +Flat

As table 3 shows, minimum-expansion gives highest precision in both experiment settings. Minimum-expansion emphasizes syntactic structures linking the anaphor and antecedent. Although using only the syntactic path may lose the contextual information, but it also prune out the potential noise within the contextual structures. In contrast, the full-expansion gives the highest recall. This is probably due to the widest knowledge coverage provides by the full-expansion syntactic tree. As a trade-off, the precision of full-expansion is the lowest in the experiments. One reason for this may be due to OntoNotes corpus is from broadcasting news domain. Its texts are less-formally structured. Another type of noise is that a narrator of news may read an abnormally long sentence. It should appear as several separate sentences in a news article. However, in broadcasting news, these sentences maybe simply joined by conjunction word “and”. Thus a very nasty and noisy structure is created from it. Comparing the three knowledge source, simple-expansion achieves moderate precision and recall which results in the highest f-score. From this, we can draw a conclusion that simple-expansion achieves a balance between the indicative structural information and introduced noises.

In the next set of experiments, we will compare different setting for training instances generation. A typical setting contains no negative instances generated from non-event anaphoric pronoun. This is not an issue for object pronoun resolution as majority of pronouns in an article is anaphoric. However in our case, the event pronoun consists of only 14.7% of the total pronouns in OntoNotes. Thus we incorporate the instances from non-event pronouns to improve the precision of the classifier. However, if we include all the negative instances from non-event anaphoric pronouns, the positive instances will be overwhelmed by the negative instances. A down sampling is applied to the training instances to create a more balanced class distribution. Table 4 reports various training settings using simple-expansion tree structure.

Simple-Exp Tree	Precision	Recall	F-score
Without Non-event Negative	0.423	**0.534**	0.472
Incl. All Negative	**0.733**	0.410	0.526
Balanced Negative	0.599	0.506	0.549
Development Data	0.652	0.492	**0.561**

Table 4: Comparison of Training Setup, Simple-Exp

In table 4, the first line is experiment without any negative instances from non-event pronouns. The second line is the performance with all negative instances from non-event pronouns. Third line is performance using a balanced training set using down sampling. The last line is experiment using hyper-plane adjustment. The first line gives the highest recall measure because it has no discriminative knowledge on non-event anaphoric pronoun. The second line yields the highest precision which complies with our claim that including negative instances from non-event pronouns will improve precision of the classifier because more discriminative power is given by non-event pronoun instances. The balanced training set achieves a better f-score comparing to models with no/all negative instances. This is because balanced training set provides a better weighted positive/negative instances which implies a balanced positive/negative knowledge representation. As a result of that, we achieve a better balanced f-score. In (Ng and Cardie, 2002b), they concluded that only the negative instances in between the anaphor and antecedent are useful in the resolution. It is same as our strategy without negative instances from non-event anaphoric pronouns. However, our study showed an improvement by adding in negative instances from non-event anaphoric pronouns as showed in table 4. This is probably due to our random sampling strategy over the negative instances near to the event anaphoric instances. It empowers the system with more discriminative power. The best performance is given by the hyper-plane adaptation model. Although the number of training instances is further reduced for development data, we can have an adjustment of the hyper-plane which is more fit to dataset.

In the last set of experiments, we will present the performance from the twin-candidates based approach in table 5. The first line is the best performance from single candidate system with hyper-plane adaptation. The second line is performance using the twin-candidates approach.

Simple-Exp Tree	Precision	Recall	F-score
Single Candidate	0.652	0.492	0.561
Twin-Candidates	0.626	**0.540**	**0.579**

Table 5: Single vs. Twin Candidates, Simple-Exp

Comparing to the single candidate model, the recall is significantly improved with a small trade-off in precision. The difference in results is statistically significant using t-test at 5% level of significance. It reinforced our intuition that preferences between two candidates are contributive information sources in co-reference resolution.

5 Conclusion and Future Work

The purpose of this paper is to conduct a systematic study of the event pronoun resolution. We propose a resolution system utilizing a set of flat positional, lexical and syntactic feature and structural syntactic feature. The state-of-arts convolution tree kernel is used to extract indicative structural syntactic knowledge. A twin-candidates preference learning based approach is incorporated to reinforce the resolution system with candidates' preferences knowledge. Last but not least, we also proposed a study of the various incorporations of negative training instances, specially using random sampling to handle the imbalanced data. Development data is also used to select more accurate hyper-plane in SVM for better determination.

To further our research work, we plan to employ more semantic information into the system such as semantic role labels and verb frames.

Acknowledgment

We would like to thank Professor Massimo Poesio from University of Trento for the initial discussion of this work.

References

N. Asher. 1993. *Reference to Abstract Objects in Discourse*. Kluwer Academic Publisher. 1993.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.1995.

M. Kubat and S. Matwin, 1997. Addressing the curse of imbalanced data set: One sided sampling. In *Proceedings of the Fourteenth International Conference on Machine Learning*,1997. pg179–186.

T. Joachims. 1999. Making large-scale svm learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press.1999.

W. Soon, H. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. In *Computational Linguistics*, Vol:27(4), pg521– 544.

D. Byron. 2002. Resolving Pronominal Reference to Abstract Entities, in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*. July 2002. , USA

M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*. July 2002. , USA

V. Ng and C. Cardie. 2002a. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02).* July 2002. , USA. pg104–111.

V. Ng, and C. Cardie. 2002b. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING02). (2002)*

M. Strube and C. Müller. 2003. A Machine Learning Approach to Pronoun Resolution in Spoken Dialogue. . In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03),* 2003

D. Klein and C. Manning. 2003a. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Advances in Neural Information Processing Systems 15 (NIPS 2002),* Cambridge, MA: MIT Press, pp. 3-10.

D. Klein and C.Manning. 2003b. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03),* 2003. pg423-430.

X. Yang, G. Zhou, J. Su, and C.Tan. 2003. Coreference Resolution Using Competition Learning Approach. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03),* 2003. pg176–183.

A. Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL'04)*, pg335–342.

A. Estabrooks, T. Jo, and N. Japkowicz. 2004. A multiple resampling method for learning from imbalanced data sets. In *Computational Intelligence V*ol:20(1). pg18–36.

X. Yang, J. Su, G. Zhou, and C. Tan. 2004. Improving pronoun resolution by incorporating coreferential information of candidates. In *Proceedings of 42th Annual Meeting of the Association for Computational Linguistics*, 2004. pg127–134.

X. Yang, J. Su and C.Tan. 2005a. Improving Pronoun Resolution Using Statistics-Based Semantic Compatibility Information. *In Proceedings of Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05).* June 2005.

X. Yang, J. Su and C.Tan. 2005b. A Twin-Candidates Model for Coreference Resolution with Non-Anaphoric Identification Capability. In *Proceedings of IJCNLP-2005*. Pp. 719-730, 2005

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. OntoNotes: The 90\% Solution. In *Proceedings of the Human Language Technology Conference of the NAACL*, 2006

X. Yang, J. Su and C.Tan. 2006. Kernel-Based Pronoun Resolution with Structured Syntactic Knowledge. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (ACL'06).* July 2006. Australia.

A. Moschitti, Making tree kernels practical for natural language learning. In *Proceedings EACL 2006,* Trento, Italy, 2006.

C. Müller. 2007. Resolving it, this, and that in unrestricted multi-party dialog. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL'07).* 2007. Czech Republic. pg816–823.

X. Yang, J. Su and C.Tan. 2008. A Twin-Candidates Model for Learning-Based Coreference Resolution. In *Computational Linguistics*, Vol:34(3). pg327-356.

S. Pradhan, L. Ramshaw, R. Weischedel, J. MacBride, and L. Micciulla. 2007. Unrestricted Coreference: Identifying Entities and Events in OntoNotes. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC),* Sep. 2007.

Unsupervised Synthesis of Multilingual Wikipedia Articles

Chen Yuncong
The Human Language Technology Center
The Hong Kong University of Science and Technology
ee_cyxab@stu.ust.hk

Pascale Fung
The Human Language Technology Center
The Hong Kong University of Science and Technology
pascale@ee.ust.hk

Abstract

In this paper, we propose an unsupervised approach to automatically synthesize Wikipedia articles in multiple languages. Taking an existing high-quality version of any entry as content guideline, we extract keywords from it and use the translated keywords to query the monolingual web of the target language. Candidate excerpts or sentences are selected based on an iterative ranking function and eventually synthesized into a complete article that resembles the reference version closely. 16 English and Chinese articles across 5 domains are evaluated to show that our algorithm is domain-independent. Both subjective evaluations by native Chinese readers and ROUGE-L scores computed with respect to standard reference articles demonstrate that synthesized articles outperform existing Chinese versions or MT texts in both content richness and readability. In practice our method can generate prototype texts for Wikipedia that facilitate later human authoring.

1 Introduction

Wikipedia has over 260 versions in different languages, but the great disparity in their scope and quality is hindering the effective spread of knowledge. The English version is currently the dominant one with over 3 million articles while the Chinese version, for example, has only one tenth the amount. Most Chinese articles suffer from content incoherence and lack of details compared to their English counterparts. Some of these articles are human-authored translation of the English version with varying degrees of accuracy and completeness, and others are ill-arranged combinations of excerpts directly adapted from external sources. The former takes considerable human effort and the latter tends to produce fragmented and incomplete texts. The intuitive solution of machine translation is also not feasible because it hardly provides satisfactory readability.

These problems call for a *synthesis* approach. In order to present the information conveyed by an English article in Chinese, instead of literally translate it, we build a topic-template expressed by the keywords extracted from the English article. Machine-translation of these keywords helps to yield the topic-template in Chinese. Using the topic-template in Chinese, we form a pool of candidate excerpts by retrieving Chinese documents from the Internet. These online documents are usually human-authored and have optimal readability and coherence. Candidate excerpts are further split into segments as synthesis unit. For segment selection, we propose an iterative ranking function that aims to maximize textual similarity, keywords coverage, and content coherence, while penalizes information redundancy.

A feature of our approach is the use of bilingual resources throughout the synthesis process. We calculate similarity scores of two texts based on both English and Chinese versions of them, which forms a more precise measure than using either version alone.

For the sake of clarity, we will use English and Chinese as examples of source and target language respectively when describing the methodology. Nonetheless, our approach is not constrained to any specific language pair and supports both direction of synthesis.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 197–205,
Beijing, August 2010

2 Related Work

Much work has been done to explore the multilingualism of Wikipedia. (Adafre et al. 2006) investigated two approaches to identify similarity between articles in different languages for automatic generation of parallel corpus, including a machine-translation based approach and one using a bilingual lexicon derived from the hyperlink structure underlying Wikipedia articles. Both methods rely on pair-wise comparisons made at the sentential level, which hardly account for similarity or coherence in the paragraph scope. Besides it is not a generative algorithm and thus inapplicable to our problem where comparable sentences in Chinese are simply not available.

A generative approach was proposed by (Sauper and Barzilay, 2009) to create highly-structured Wikipedia articles (e.g. descriptions of diseases) composed of information drawn from the Internet. It uses an automatically-induced domain-specific template, and the perceptron algorithm augmented with a global integer linear programming (ILP) formulation to optimize both local fit of information into each section and global coherence across the entire article. This method works only for specific domains where articles have obviously separable sections (e.g. Causes and Symptoms) and it requires a training corpus for each domain to induce the template. Moreover, the synthesis units they use are complete excerpts rather than individual sentences as in our approach. Their choice is based on the assumption that texts on the Internet appear in complete paragraphs, with structure strictly adhere to the fixed training templates, which may be true for specific domains they test on, but fails to hold for domain-independent application. Instead, our algorithm aims to synthesize the article in the sentential level. We select sentences to fit the source content at run time, regardless to whether a pre-determined structural template exists or not. Therefore the requirement on the structures of source articles becomes very flexible, enabling our system to work for arbitrary domain. In a sense, rather than being a structure-aware approach, our algorithm performs in a content-aware manner. This also makes maintaining coherence throughout article a lot more challenging.

Works on monolingual extractive text summarization also lend insights into our problem. (Goldstein et al., 2000) used sequential sentence selection based on Maximal Marginal Relevance Multi-Document (MMR-MD) score to form summarizations for multiple documents, with the constraint of sentence count. Since our problem does not have this constraint, we employ a variant of MMR-MD and introduced new terms specific to this task. (Takamura and Okumura, 2009) formulated a text summarization task as a maximum coverage problem with knapsack constraint and proposed a variety of combinatorial mathematics-based algorithms for solving the optimization problem.

For multi-lingual summarization, (Evans, 2005) applied the concept of multi-lingual text similarity to summarization and improved readability of English summaries of Arabic text by replacing machine translated Arabic sentences with highly similar English sentences whenever possible.

3 Methodology

Figure 1 describes the high-level algorithm of our approach. The system takes as input the English Wikipedia page and outputs an article in Chinese.

First, the structured English article is extracted from the Wikipedia page. Due to the relative independence of contents in different sections in typical Wikipedia articles (e.g. childhood, early writings), a separate synthesis task is performed on each section and all synthesized sections are eventually combined in the original order to form the Chinese article.

For each section, keywords are extracted from the English text using both tf-idf and the graph-based TextRank algorithm. Named entities, time indicators, and terms with Wikipedia hyperlinks are also included. These keywords express the topics of the current section and are regarded as the content guideline. We then use Google Translate and Google Dictionary to

```
Input:
        English version of an entry
Output:
        Synthesized Chinese version
Algorithm:
        1: Parse the English Wikipedia page to extract the structured texts.
        2: For each section:
                2.1: Extract keywords.
                2.2: Use Chinese translation of keywords to search online Chinese texts.
                2.3: Filter retrieved Chinese texts and split them into segments.
                2.4: Synthesize the current section using candidate segments.
        3: Generate the Chinese Wikipedia page by combining synthesized sections according
        to the original structure of English version.
```

Figure 1. High-level algorithm of the synthesis approach

obtain the Chinese translations of these keywords and thereby convert the content guideline into Chinese. The Chinese keywords are then combined with the translated subject term and section title to form queries that are used to retrieve online Chinese documents by Google search. The returned Chinese documents are clustered and filtered based on both their format and content. The remaining candidate excerpts are further split using the TextTiling algorithm (Hearst, 1997) into segments that constitutes the text units for synthesis. This unit size ensures both semantic completeness within each unit and flexibility of combining multiple units into coherent paragraphs. Segments are chosen according to scores computed iteratively by a variant of the MMR-MD scoring function that considers not only the relevance of an individual segment to the source section but also its impact on the provisional synthesized section as a whole.

3.1 Wikipedia Page Preprocessing

The source Wikipedia page is parsed to remove non-textual page elements (e.g. images, info-boxes and side-bars). Only texts and headings are extracted and their structures are maintained as templates for final integration of synthesized sections.

3.2 Keyword Extraction

The keyword set K for a section is the union of 6 categories of content-bearing terms.

$$K = \bigcup K_c$$

K_1: set of terms with high tf-idf score (top 5%)
K_2: set of terms with high TextRank score (top 5%)
K_3: set of named entities
K_4: set of temporal indicators (e.g. June, 1860)
K_5: set of terms with Wikipedia links
K_6: section title

For K_1, tf-idf scores are computed by:

$$tfidf_i = \sqrt{tf_i} \times \log\left(\frac{N}{df_i} + 1\right)$$

where tf_i is the term frequency of term *i* in the section and df_i is the document frequency of term *i* in a corpus consists of 2725 high-quality English Wikipedia articles[1], which well represent the language style of Wikipedia.

For K_2, we compute TextRank scores according to (Mihalcea and Tarau, 2004). It is a graph-based model where words as vertices recursively vote for the weights of their linked neighbors (e.g. words appear in the same sentence as them) using the formula:

$$WS(V_i) = \\ (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

[1] http://evanjones.ca/software/wikipedia2text.html

Where $In(V_i)$ is the set of vertices with forward links to i, $Out(V_i)$ is the set of vertices receiving links from i, w_{ji} is the weight of edge between V_i and V_j. In the case of a word graph, we simplify this formula by assuming the graph to be undirected and unweighted. Each pair of words occurring in the same sentence share an edge between them and all word vertices have initial weights of 1.

Unlike tf-idf which considers only word-specific values and tends to give higher weights for rare words, TextRank uses global information about how a word is used in its context to induce its importance and has the advantage of highlighting keywords that are relatively common but highly relevant. In this sense, these two measures complement each other. Named entities are recognized using the named entity chunker provided by the NLTK (Natural Language ToolKit) package[2].

3.3 Keyword Translation

Keywords are then translated using Google Dictionary to form Chinese queries. Usually one English keyword has several translations and they will be used jointly when forming the search query.

Google Dictionary often fails to generate correct transliteration for rare names, so we augment it with a function of parenthesized phrase translation. We basically seeks named-entity strings from online documents that are in the format of '*CHINESE* (*ENGLISH*)' and extracts the Chinese transliteration from the pattern using regular expression combined with a Pinyin (Chinese Romanization)[3] /English pronunciation lookup table. Since Chinese words are not spaced in documents, the Pinyin/English lookup is helpful to determine the boundary of the Chinese transliteration based on the fact that most Chinese transliterations start with characters pronounced similar to the initial syllables in corresponding English names. This function is relatively simple but works surprisingly well as many rare named entities are available in this pattern on the Web.

3.4 Web Search

Keywords in Chinese alternatively form query pairs with the Wikipedia subject term. Each pair is used to retrieve a set of (16 in our experiments) Chinese documents containing both words with Google Search. If a keyword has multiple translations, they are joined by the string 'OR' in the query which is the way to specify alternatives in Google logic. If a keyword is a named entity, its English version is also used as an alternative in order to acquire documents in which the subject is referred to by its English name instead of transliterations. For the subject "Chekhov/契诃夫", a keyword with two transliterations "Taganrog/塔甘罗格/塔干罗格" and another keyword with two transliterations "father/父亲/爸爸" will result in two query pairs: "Chekhov OR 契诃夫 Taganrog OR 塔甘罗格 OR 塔干罗格" and "Chekhov OR 契诃夫 父亲 OR 爸爸".

3.5 Candidate Filtering

The retrieved excerpts are filtered first by criteria on format include text length and the percentage of white-space and non-Chinese characters. Pair-wise similarity is then computed among all the remaining excerpts and those above a certain threshold are clustered. Within a cluster only the centroid excerpt with maximum similarity with the source section will be selected. This stage typically eliminates ¾ of the documents that are either not sufficiently relevant or redundant. The similarity measure we use is a combination of both English and Chinese versions of cosine similarity and Jaccard index.

$$\begin{aligned}\mathrm{SIM}(\mathrm{a},\mathrm{b}) = 0.3 \times \mathrm{COS}_{\mathrm{EN}}(\mathrm{a},\mathrm{b}) + 0.3 \times \mathrm{COS}_{\mathrm{CH}}(\mathrm{a},\mathrm{b}) +\\ 0.2 \times \mathrm{JAC}_{\mathrm{EN}}(\mathrm{a},\mathrm{b}) + 0.2 \times \mathrm{JAC}_{\mathrm{CH}}(\mathrm{a},\mathrm{b})\end{aligned}$$

For Chinese excerpts, English similarity is computed by first translating them into English by Google Translate and taking tf-idf as token weights. Similar procedure works for computing Chinese similarity for English excerpts, except that Chinese texts need to be

[2] The package is available at http://www.nltk.org

[3] Pinyin information is obtained from Unicode Han Database at http://www.unicode.org/reports/tr38/

segmented[4] first and weights are based on tf only. These machine translations do not require grammatical correctness since they are essentially used as bags of words in both cosine similarity and Jaccard index. During this stage, every excerpt acquires bi-lingual versions, which is important for the extended similarity measure in the iterative ranking function.

Filtered excerpts are further split into segments using the TextTiling algorithm. After clustering the remaining segments form the candidate units for synthesis of the current section.

3.6 Iterative Scoring Function

Based on the idea that the 'goodness' of a segment should be evaluated both on its individual relevance to the source and the overall impact on the synthesized section, we summarize four factors for scoring a segment: (1) Intuitively a segment scores higher if it has higher similarity to the source section; (2) A segment makes positive contribution to synthesized section if it introduces some keywords mentioned in the source; (3) A segment tends to improve the coherence of synthesized section if it comes from the same excerpts as the other segments in synthesized section; (4) A sentence should be penalized if its content is redundant with the synthesized section.

Integrating the four factors above, we propose that for source text *r*, the score of the *i*th candidate segment s_i in the *n*th iteration is formulated as:

$$Q_{r,n}(s_i) = w_s \times S_r(s_i) + w_k \times K_{r,n}(s_i) + \\ w_c \times C_n(s_i) - w_R \times R_n(s_i)$$

This formula is composed of 4 terms corresponding to the 'goodness' factors: $S_r(s_i)$ for similarity, $K_{r,n}(s_i)$ for keyword coverage, $C_n(s_i)$ for coherence, and $R_n(s_i)$ for redundancy. The corresponding weights are tuned in a large number of experiments as to achieve optimal performance. This function is a variant of the original MMR-MD score tailored for our application.

$S_r(s_i)$ is a comprehensive similarity measure of segment s_i to the reference text r.

$$S_r(s_i) = w_1 \times SIM(s_i, \mathrm{r}) + w_2 \times SIM(s_i, \mathrm{p}) + \\ w_3 \times SIM(e_i, \mathrm{r}) + w_4 \times SIM(e_i, \mathrm{p})$$

where p is the parent section of r and e_i is the parent excerpt of s_i. Similarities between parent excerpts are also examined because sometimes two segments, especially short segments, despite their textual similarity actually come from very different contexts and exhibit different focuses. In this case, the latter three terms will suppress the score between these two segments which would otherwise be erroneously high and therefore produce a more precise measure of similarity.

$\mathrm{K_{r,n}(s_i)}$ measures the contribution of $\mathrm{s_i}$ in terms of uncovered keywords.

$$\mathrm{K_{r,n}(s_i)} = \sum_{\substack{\mathrm{k \in U_{r,n}} \\ \mathrm{k \neq subject}}} \mathrm{idf(k)}$$

$$\mathrm{U_{r,n}} = \mathrm{K_r} - \bigcup_{\mathrm{s_j \in D_n}} \mathrm{K_j}$$

where $\mathrm{D_n}$ is the winner set in the nth iteration. $\mathrm{K_r}$ is the set of keywords in the reference text and $\mathrm{K_j}$ is the set of keywords in the selected segment $\mathrm{s_j}$. $\mathrm{U_{r,n}}$ represents the set of keywords in the reference that are not yet been covered by the provisional synthesized text in the nth iteration. $\mathrm{K_{r,n}(s_i)}$ quantifies the keyword contribution as the sum of idf values of uncovered keywords. The subject term is excluded because it as a keyword does not reflect any topic bias and is therefore not a good indicator for coverage.

$C_n(s_i)$ is a term that reflects the coherence and readability in the synthesized text.

$$C_n(s_i) = \left|\left\{\mathrm{s_j} \middle| \mathrm{e_j} = \mathrm{e_i}, \mathrm{s_j} \in D_n\right\}\right|$$

[4] The segmentation tool using forward maximum matching is obtained at http://technology.chtsai.org/mmseg

where e_i is the parent excerpt of s_i and e_j is the parent excerpt of s_j. Segments from the same excerpts tend to be less redundant and more coherent. Therefore candidates that share the same parent excerpts as segments in winner set are more favorable and rewarded by this term. This is a major difference from the original MMR-MD function in which sentences from different documents are favored. This is because their formula is targeted for automatic summarization where more emphasis is put on diversity rather than coherence.

$R_n(s_i)$ measures the redundancy of the synthesized text if s_i is included. It is quantified as the maximum similarity of s_i with all selected segments.

$$R_n(s_i) = \max_{s_j \in D_n} S(s_i, s_j)$$

3.7 Segment Selection Algorithm

Figure 2 describes the segment selection algorithm. Starting with a candidate set and an empty winner set, we iteratively rank the candidates by Q and in each iteration the top-ranked segment is examined. There are two circumstances a segment would be selected for the winner set:

(1) if the segment scores sufficiently high
(2) the segment does not score high enough for an unconditional selection, but as long as it introduces uncovered keywords, its contribution to the overall content quality may still overweigh the compromised similarity

In the second circumstance however, since we are only interested in the uncovered keywords, it may not be necessary for the entire segment to be included in the synthesized text. Instead, we only include the sentences in this segment that contain those keywords. Therefore we propose two conditions:

- $\boldsymbol{C_{sel\text{-}segment}}$: condition for selecting a segment
$$Q_{r,n}(s_{top}) > 0.8 * \mathrm{Q_{max}}$$
- $\boldsymbol{C_{sel\text{-}sentence}}$: condition for selecting sentences $Q_{r,n}(\mathrm{s_{stop}}) > 0.6 * \mathrm{Q_{max}}$ **and** $K_{r,n}(\mathrm{s_{stop}}) > 0$ **and** $S_r(s_{top}) > 0.3 * \mathrm{S_{max}}$

Thresholds in both conditions are not static but dependent on the highest score of all candidates in order to accommodate diversity in score range for different texts. Finally if no more candidates are able to meet the lowered score threshold, even if they might carry new keywords, we assume they are not suitable for synthesis and return the current winner set. This break condition is formulated as $\boldsymbol{C_{break}}$:

- $\boldsymbol{C_{break}}$: condition to finish selection
$$Q_{r,n}(s_{top}) < 0.6 * \mathrm{Q_{max}}$$

```
Input:
        $S_n$:      candidate set in iteration $n$
        $r$:        the reference text
Define:
        $n$:        iteration index
        $D_n$:      winner set in iteration $n$
        $C_{sel\text{-}segment}$:    $Q_{r,n}(s_{top}) > 0.8 * Q_{max}$
        $C_{sel\text{-}sentence}$:   $Q_{r,n}(s_{top}) > 0.6 * Q_{max}$
                          and $K_{r,n}(s_{top}) > 0$
                          and $S_r(s_{top}) > 0.3 * S_{max}$
        $C_{break}$:           $Q_{r,n}(s_{top}) < 0.6 * Q_{max}$
Algorithm:
        $D_n \leftarrow \emptyset$, $n \leftarrow 0$
        while $S_n \neq \emptyset$:
          $s_{top} \leftarrow \arg\max_{s_i \in S_n} Q_{r,n}(s_i)$
          if $C_{break}$:
            return $D_n$
          else if $C_{sel\text{-}segment}$:
            $D_n \leftarrow D_n + s_{top}$
          else if $C_{sel\text{-}sentence}$:
            $D_n \leftarrow D_n +$ sentences in $s_{top}$ with
                      the uncovered keywords
            $S_n \leftarrow S_n - s_{top}$
            $n \leftarrow n + 1$
Output:
        Synthesized text for the reference $r$
```

Figure 2. Segment selection algorithm

4 Evaluation

4.1 Experiment Setup

We evaluate our system on 16 Wikipedia subjects across 5 different domains as listed in Table 1.

Category	**Subjects**
Person	Anton Chekhov Abu Nuwas Joseph Haydn Li Bai
Organization	HKUST IMF WTO
Events	Woodstock Festival Invasion of Normandy Decembrist Revolt
Science	El Nino Gamma Ray Stingray
Culture	Ceramic Art Spiderman Terrorism

Table 1. Subjects used for evaluation

The subjects are selected from “the List of Articles Every Wikipedia Should Have” [5] published by Wikimedia. These subjects are especially appropriate for our evaluation because we can (1) use a subset of such articles that have high quality in both English and Chinese as standard reference for evaluation; (2) safely assume Chinese information about these subjects is widely available on the Internet; (3) take subjects currently without satisfactory versions in Chinese as our challenge.

Human Evaluation

We presented the synthesized articles of these subjects to 5 native Chinese readers who compare synthesized articles with MT results and existing Chinese versions on Wikipedia which range from translated stubs to human-authored segments. We asked the reviewers to score them on a 5-point scale in terms of four quality indicators: structural similarity to the English version, keyword coverage, fluency, and conciseness.

Automatic Evaluation

In addition to human evaluation, we also compare synthesized articles to several high-quality Chinese Wikipedia articles using ROUGE-L (C.Y. Lin, 2004). We assume these Chinese versions are the goals for our synthesis system and greater resemblance with these standard references indicates better synthesis. ROUGE-L measures the longest common subsequence (LCS) similarity between two documents, rather than simply word overlap so it to some degree reflects fluency.

4.2 Result Analysis

Human Evaluation

Human evaluator feedbacks for articles in different categories are shown in Table 2. Machine-translated versions are judged to have the highest score for structural similarity, but erroneous grammar and word choices make their readability so poor even within sentences and therefore of no practical use.

Generally, articles synthesized by our system outperform most existing Chinese versions in terms of both structural and content similarity. Many existing Chinese versions completely ignore important sections that appear in English versions, while our system tries to offer information with as much fidelity to the English version as possible and is usually able to produce information for every section. Synthesized articles however, tend to be less fluent and more redundant than human-authored versions.

Performance varies in different domains. Synthesis works better for subjects in *Person* category, because the biographical structure provides a specific and fairly unrelated content in each section, making the synthesis less redundancy-prone. On the other hand, there is arbitrariness when organizing articles in *Event* and *Culture* category. This makes it difficult to find online text organized in the same way as the English Wikipedia version, therefore introducing a greater challenge in sentence selection for each section. Articles in the *Science* category usually include rare terminologies, and formatted texts like diagrams and formula, which impede correct translation and successful extraction of keywords.

[5] http://meta.wikimedia.org/wiki/List_of_articles_every_Wikipedia_should_have/Version_1.2

Cat.	**Structural Similarity**			**Coverage**			**Fluency**			**Conciseness**		
	Synt.	Orig.	MT	**Synt.**	Orig.	MT	**Synt.**	Orig.	MT	**Synt.**	Orig.	MT
Psn.	**2.85**	1.49	5	**2.94**	1.84	4.51	**2.71**	4.58	0.83	**1.74**	4.47	n/a
Org.	**1.96**	1.22	5	**2.51**	2.10	4.46	**2.10**	4.42	1.06	**0.99**	4.53	n/a
Evt.	**1.37**	1.13	5	**2.56**	1.94	4.40	**2.45**	4.46	0.81	**0.80**	4.40	n/a
Sci.	**2.43**	1.30	5	**2.68**	2.14	4.42	**2.53**	4.51	1.02	**1.05**	4.50	n/a
Cul.	**1.39**	1.35	5	**2.2**	2.21	4.54	**2.32**	4.54	0.94	**1.34**	4.59	n/a
Avg.	**2.02**	1.30	5	**2.58**	2.05	4.47	**2.42**	4.50	0.93	**1.22**	4.50	n/a

Table 2. Result of human evaluation against English source articles (out of 5 points; Synt: synthesized articles; Orig: the existing human-authored Chinese Wikipedia versions; MT: Chinese versions generated by Google Translate)

Automatic Evaluation

Using ROUGE-L to measure the quality of both synthesized and MT articles against human-authored standard references, we find synthesized articles generally score higher than MT versions. The results are shown in Table 3.

Category	**Recall**		**Precision**		**F-score**	
	Synt.	MT	Synt.	MT	Synt.	MT
Psn.	**0.48**	0.30	**0.20**	0.16	**0.28**	0.22
Org.	**0.40**	0.29	**0.16**	0.13	**0.23**	0.18
Evt.	**0.36**	0.26	0.13	**0.15**	**0.19**	0.19
Sci.	**0.31**	0.22	**0.14**	0.11	**0.19**	0.15
Cul.	**0.37**	0.27	**0.13**	0.12	**0.24**	0.17
Avg.	**0.38**	0.27	**0.15**	0.13	**0.23**	0.18

Table 3. Results of automatic evaluation against gold Chinese reference articles (Synt: synthesized articles; MT: Chinese versions generated by Google Translate)

The synthesized articles, extracted from high quality human-authored monolingual texts, are generally better in precision than the MT articles because there is less erroneous word choice or grammatical mistakes. Most synthesized articles also have higher recall than MT versions because usually a substantial portion of the high-quality Chinese excerpts, after being retrieved by search engine, will be judged by our system as good candidate texts and included into the synthesized article. This naturally increases the resemblance of synthesized articles to standard references, and thus the F-scores. Note that since our method is unsupervised, the inclusion of the standard Chinese articles underscores the precision and recall of our method.

5 Conclusion

In this paper, we proposed an unsupervised approach of synthesizing Wikipedia articles in multiple languages based on an existing high-quality version of any entry. By extracting keywords from the source article and retrieving relevant texts from the monolingual Web in a target language, we generate new articles using an iterative scoring function.

Synthesis results for several subjects across various domains confirmed that our method is able to produce satisfactory articles with high resemblance to the source English article. For many of the testing subjects that are in ‘stub’ status, our synthesized articles can act as either replacement or supplement to existing Chinese versions. For other relatively well-written ones, our system can help provide content prototypes for missing sections and missing topics, bootstrapping later human editing.

A weakness of our system is the insufficient control over coherence and fluency in paragraph synthesis *within* each section, new methods are being developed to determine the proper order of chosen segments and optimize the readability.

We are working to extend our work to a system that supports conversion between major languages such as German, French and Spanish. The employment of mostly statistical methods in our approach facilitates the extension. We have also released a downloadable desktop application and a web application based on this system to assist Wikipedia users.

Reference

Adafre, Sisay F. and Maarten de Rijke, “Finding Similar Sentences across Multiple Languages in Wikipedia”, *Proceedings of the EACL Workshop on New Text*, Trento, Italy, 2006

Bird, Steven, E. Klein, and E. Loper, *Natural Language Processing with Python --- Analyzing Text with the Natural Language Toolkit*, O'Reilly Media, 2009

Evans, David K., “Identifying Similarity in Text: Multi-Lingual Analysis for Summarization”, PhD thesis, Columbia University, 2005.

Goldstein, Jade, Vibhu Mittal, Jaime Carbonell and Mark Kantrowitz, “Multi-document summarization by sentence extraction”, *NAACL-ANLP 2000 Workshop on Automatic summarization*, pages 40-48, 2000

Hearst, Marti A., “TextTiling: Segmenting Text into Multi-paragraph Subtopic Passages”, *Computational Linguistics*, Volume 23, Issue 1, pp. 33-64, 1997

Lin, Chin-Yew, “ROUGE: A Package for Automatic Evaluation of Summaries”, *Proceedings of Workshop on Text Summarization Branches Out, Post-Conference Workshop of ACL 2004*, Barcelona, Spain.

Mihalcea, Rada and Paul Tarau, “TextRank: Bringing order into texts”, *Proceedings of EMNLP*, pages 404–411 Barcelona, Spain, 2004

Sauper, Christina and Regina Barzilay, “Automatically Generating Wikipedia Articles: a Structure-Aware Approach”, *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 208–216, Suntec, Singapore, 2-7 August 2009.

Takamura, Hiroya and Manabu Okumura, “Text Summarization Model based on Maximum Coverage Problem and its Variant”, *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 781-789, 2009

Simplicity is Better: Revisiting Single Kernel PPI Extraction

Sung-Pil Choi
Information Technology Laboratory
Korea Institute of Science and Technology Information
spchoi@kisti.re.kr

Sung-Hyon Myaeng
Department of Computer Science
Korea Advanced Institute of Science and Technology
myaeng@kaist.ac.kr

Abstract

It has been known that a combination of multiple kernels and addition of various resources are the best options for improving effectiveness of kernel-based PPI extraction methods. These supplements, however, involve extensive kernel adaptation and feature selection processes, which attenuate the original benefits of the kernel methods. This paper shows that we are able to achieve the best performance among the state-of-the-art methods by using only a single kernel, convolution parse tree kernel. In-depth analyses of the kernel reveal that the keys to the improvement are the tree pruning method and consideration of tree kernel decay factors. It is noteworthy that we obtained the performance without having to use any additional features, kernels or corpora.

1 Introduction

Protein-Protein Interaction (PPI) Extraction refers to an automatic extraction of the interactions between multiple protein names from natural language sentences using linguistic features such as lexical clues and syntactic structures. A sentence may contain multiple protein names and relations, i.e., multiple PPIs. For example, the sentence in Fig.1 contains a total of six protein names of varying word lengths and three explicit interactions (relations). The interaction type between *phosphoprotein* and the acronym *P* in the parentheses is "*EQUAL*." A longer protein name *phosphoprotein of vesicular stomatitis virus* is related to *nucleocapsid protein* via "*INTERACT*" relation. Like the first PPI, *nucleocapsid protein* is equivalent to the abbreviated term *N*.

It is not straightforward to extract PPIs from a sentence or textual segment. There may be multiple protein names and their relationships, which are intertwined in a sentence. An interaction type may be expressed in a number of different ways.

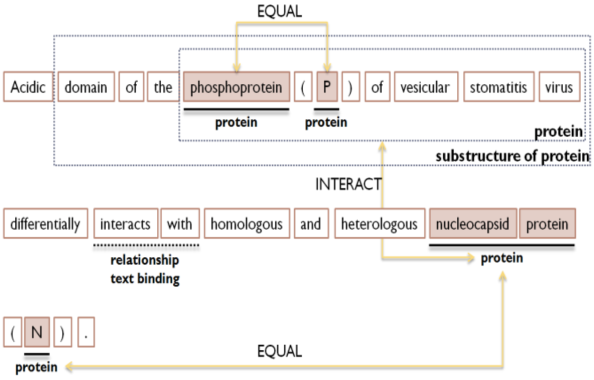


Figure 1. An example sentence containing multiple PPIs involving different names of varying scopes and relations[1]

A significant amount of efforts have been devoted to kernel-based approaches to PPI extractions (PPIE) as well as relation extractions[2] (Zhang et al., 2006; Pyysalo et al., 2008; Guo-Dong et al., 2007; Zhang et al., 2008; Airola et al., 2008; Miwa et al., 2009). They include word feature kernels, parse tree kernels, and graph kernels. One of the benefits of using a kernel method is that it can keep the original

[1] BioInfer, Sentence ID:BioInfer.d10.s0

[2] Relation extraction has been studied massively with the help of the ACE (www.nist.gov/tac) competition workshop and its corpora. The ACE corpora contain valuable information showing the traits of target entities (e.g., entity types, roles) for relation extraction in single sentences. Since all target entities are of the same type, protein name, in PPIE, however, we cannot use relational information that exists among entity types. This makes PPIE more challenging.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 206–214,
Beijing, August 2010

formation of target objects such as parse trees, not requiring extensive feature engineering for learning algorithms (Zelenko et al., 2003).

In an effort to improve the performance of PPIE, researchers have developed not only new kernels but also methods for combining them (GuoDong et al., 2007; Zhang et al., 2008; Airola et al., 2008; Miwa et al., 2009a; Miwa et al., 2009b). While the intricate ways of combing various kernels and using extra resources have played the role of establishing strong baseline performance for PPIE, however, they are viewed as another form of engineering efforts. After all, one of the reasons the kernel methods have become popular is to avoid such engineering efforts.

Instead, we focus on a state-of-the-art kernel and investigate how it can be best utilized for enhanced performance. We show that even with a single kernel, convolution parse tree kernel in this case, we can achieve superior performance in PPIE by devising an appropriate preprocessing and factor adjustment method. The keys to the improvement are tree pruning and consideration of a tree kernel decay factor, which are independent of the machine learning model used in this paper. The main contribution of our work is the extension and application of the particular convolution tree kernel method for PPIE, which gives a lesson that a deep analysis and a subsequent extension of a kernel for maximal performance can override the gains obtained from engineering additional features or combining other kernels.

The remaining part of the paper is organized as follows. In section 2, we survey the existing approaches. Section 3 introduces the parse tree kernel model and its algorithm. Section 4 explains the performance improving factors applied to the parse tree kernel. The architecture of our system is introduced in section 5. Section 6 shows the improvements in effectiveness in multiple PPI corpora and finally we conclude our work in section 7.

2 Related Work

In recent years, numerous studies have attempted to extract PPI automatically from text. Zhou and He (2008) classified various PPIE approaches into three categories: linguistic, rule-based and machine learning and statistical methods.

Linguistic approaches involve constructing special grammars capable of syntactically expressing the interactions in sentences and then applying them to the language analyzers such as part-of-speech taggers, chunkers and parsers to extract PPIs. Based on the level of linguistic analyses, we can divide the linguistic approaches into two categories: shallow parsing (Sekimizu et al., 1998; Gondy et al., 2003) and full parsing methods (Temkin & Gilder, 2003; Nikolai et al., 2004).

Rule-based approaches use manually defined sets of lexical patterns and find text segments that match the patterns. Blaschke et al. (1996) built a set of lexical rules based on clue words denoting interactions. Ono et al. (2001) defined a group of lexical and syntactic interaction patterns, embracing negative expressions, and applied them to extract PPIs from documents about "*Saccharomyces cerevisiae*" and "*Escherichia coli*". Recently, Fundel et al. (2007) proposed a PPI extraction model based on more systematic rules using a dependency parser.

Machine learning and statistical approaches have been around for a while but have recently become a dominant approach for PPI extraction. These methods involve building supervised or semi-supervised models based on training sets and various feature extraction methods (Andrade & Valencia, 1998; Marcotte et al., 2001; Craven & Kumlien, 1999). Among them, kernel-based methods have been studied extensively in recent years. Airola et al. (2008) attempted to extract PPIs using a graph kernel by converting dependency parse trees into the corresponding dependency graphs.

Miwa et al. (2009a) utilized multiple kernels such as word feature kernels, parse tree kernels, and even graph kernels in order to improve the performance of PPI extraction. Their experiments based on five PPI corpora, however, showed that combining multiple kernels gave only minor improvements compared to other methods. To further improve the performance of the multiple kernel system, the same group combined multiple corpora to exploit additional features for a modified SVM model (Miwa et al., 2009b). While they achieved the best performance in PPI extraction, it was possible only with additional kernels and corpora from which additional features were extracted.

Unlike the aforementioned approaches trying to use all possible resources for performance enhancement, this paper aims at maximizing the performance of PPIE using only a single kernel without any additional resources. Without lowering the performance, we attempt to stick to the initial benefits of the kernel methods: *simplicity* and *modularity* (Shawe-Taylor & Cristianini, 2004).

3 Convolution Parse Tree Kernel Model for PPIE

The main idea of a convolution parse tree kernel is to sever a parse tree into its sub-trees and transfer it as a point in a vector space in which each axis denotes a particular sub-tree in the entire set of parse trees. If this set contains M unique sub-trees, the vector space becomes M-dimensional. The similarity between two parse trees can be obtained by computing the inner product of the two corresponding vectors, which is the output of the parse tree kernel.

There are two types of parse tree kernels of different forms of sub-trees: one is *SubTree Kernel* (*STK*) proposed by Vishwanathan and Smola (2003), and the other is *SubSet Tree Kernel* (*SSTK*) developed by Collins and Duffy (2001). In *STK*, each sub-tree should be a complete tree rooted by a specific node in the entire tree and ended with leaf nodes. All the sub-trees must obey the production rules of the syntactic grammar. Meanwhile, *SSTK* can have any forms of sub-trees in the entire parse tree given that they should obey the production rules. It was shown that *SSTK* is much superior to *STK* in many tasks (Moschitti, 2006). He also introduced a fast algorithm for computing a parse tree kernel and showed its beneficial effects on the semantic role labeling problem.

A parse tree kernel can be computed by the following equation:

$$K(T_1, T_2, \lambda, \sigma) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2, \lambda, \sigma) \quad (1)$$

where T_i is i^{th} parse tree and n_1 and n_2 are nodes in N_T, the set of the entire nodes of T. λ represents a tree kernel decay factor, which will be explained later, and σ decides the way the tree is severed. Finally $\Delta(n_1, n_2, \lambda, \sigma)$ counts the number of the common sub-trees of the two parse trees rooted by n_1 and n_2. Figure 2 shows the algorithm.

In this algorithm, the *get_children_number* function returns the number of the direct child nodes of the current node in a tree. The function named *get_node_value* gives the value of a node such as part-of-speeches, phrase tags and words. The *get_production_rule* function finds the grammatical rule of the current node and its children by inspecting their relationship.

```
FUNCTION delta(TreeNode $n_1$, TreeNode $n_2$, $\lambda$, $\sigma$)
$n_1$ = one node of $T_1$;  $n_2$ = one node of $T_2$;
$\lambda$ = tree kernel decay factor;  $\sigma$ = tree division method;
BEGIN
   $nc_1$ = get_children_number($n_1$);
   $nc_2$ = get_children_number($n_2$);
   IF $nc_1$ EQUAL 0 AND $nc_2$ EQUAL 0 THEN
      $nv_1$ = get_node_value($n_1$);
      $nv_2$ = get_node_value($n_2$);
      IF $nv_1$ EQUAL $nv_2$ THEN RETURN 1;
   ENDIF
   $np_1$ = get_production_rule($n_1$);
   $np_2$ = get_production_rule($n_2$);
   IF $np_1$ NOT EQUAL $np_2$ THEN RETURN 0;

   IF $np_1$ EQUAL $np_2$ AND $nc_1$ EQUAL 1
           AND $nc_2$ EQUAL 1 THEN
      RETURN $\lambda$;
   END IF

   mult_delta = 1;
   FOR I = 1 TO $nc_1$
      $nch_1$ = I^th child of $n_1$;   $nch_2$ = I^th child of $n_2$;
      mult_delta = mult_delta ×
                     ($\sigma$ + delta($nch_1$, $nch_2$, $\lambda$, $\sigma$));
   END FOR
   RETURN $\lambda$ × mult_delta;
END
```

Figure 2. Δ (n_1, n_2, λ, σ) algorithm

4 Performance Improving Factors

4.1 Tree Pruning Methods

Tree pruning for relation extraction was firstly introduced by Zhang et al. (2006) and also referred to as "*tree shrinking task*" for removing less related contexts. They suggested five types of the pruning methods and later invented two more in Zhang et al. (2008). Among them, the path-enclosed tree (*PT*) method was shown to give the best result in the relation extraction task based on ACE corpus. We opted for this pruning method in our work.

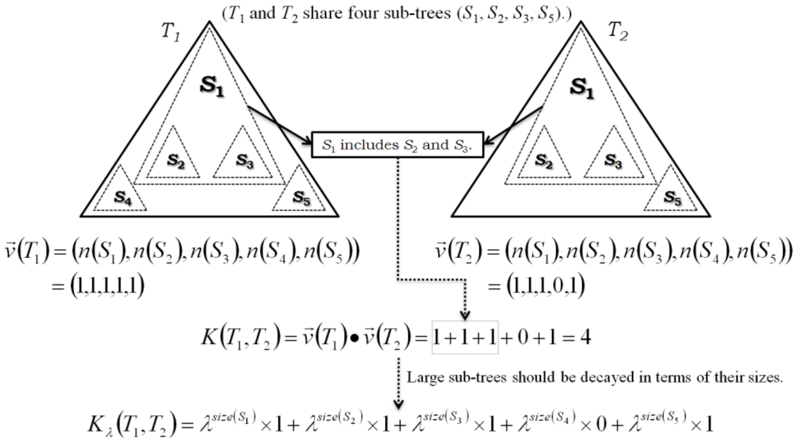

(T_1 and T_2 share four sub-trees (S_1, S_2, S_3, S_5).)
T_1
T_2
S_1
S_2
S_3
S_4
S_5
S_1 includes S_2 and S_3.
$\vec{v}(T_1) = (n(S_1), n(S_2), n(S_3), n(S_4), n(S_5)) = (1,1,1,1,1)$
$\vec{v}(T_2) = (n(S_1), n(S_2), n(S_3), n(S_4), n(S_5)) = (1,1,1,0,1)$
$K(T_1, T_2) = \vec{v}(T_1) \bullet \vec{v}(T_2) = \boxed{1+1+1} + 0 + 1 = 4$
Large sub-trees should be decayed in terms of their sizes.
$K_\lambda(T_1, T_2) = \lambda^{size(S_1)} \times 1 + \lambda^{size(S_2)} \times 1 + \lambda^{size(S_3)} \times 1 + \lambda^{size(S_4)} \times 0 + \lambda^{size(S_5)} \times 1$

Figure 4. The effect of decaying in comparing two trees. $n(\cdot)$ denotes #unique subtrees in a tree.

Figure 3 shows how the *PT* method prunes a tree. To focus on the pivotal context, it preserves only the syntactic structure encompassing the two proteins at hand and the words in between them (the part enclosed by the dotted lines). Without pruning, all the words like *addition, increased* and *activity* would intricately participate in deciding the interaction type of this sentence.

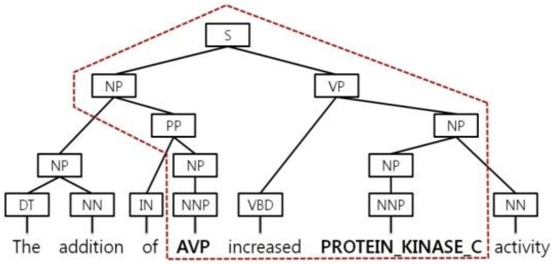


Figure 3. Path-enclosed Tree (PT) Method

Another important effect of the tree pruning is its ability to separate features when two or more interactions exist in a sentence. As in Figure 1, each interaction involves its unique context even though a sentence has multiple interactions. With tree pruning, it is likely to extract context-sensitive features by ignoring external features.

4.2 Tree Kernel Decay Factor

Collins and Duffy (2001) addressed two problems of the parse tree kernel. The first one is that its kernel value tends to be largely dominated by the size of two input trees. If they are large in size, it is highly probable for the kernel to accumulate a large number of overlapping counts in computing their similarity. Secondly, the kernel value of two identical parse trees can become overly large while the value of two different parse trees is much tiny in general. These two aspects can cause a trouble during a training phase because pairs of large parse trees that are similar to each other are disproportionately dominant. Consequently, the resulting models could act like nearest neighbor models (Collins and Duffy, 2001).

To alleviate the problems, Collins and Duffy (2001) introduced a scalability parameter called decay factor, $0 < \lambda \leq 1$ which scales the relative importance of tree fragments with their sizes as in line 33 of Fig. 2. Based on the algorithm, a decay factor decreases the degree of contribution of a large sub-tree exponentially in kernel computation. Figure 4 illustrates both the way a tree kernel is computed and the effect of a decay factor. In the figure, T_1 and T_2 share four common sub-trees (S_1, S_2, S_3, S_5). Let us assume that there are only two trees in a training set and only five unique sub-trees exist. Then each tree can be expressed by a vector whose elements are the number of particular sub-trees. Kernel value is obtained by computing the inner product of the two vectors. As shown in the figure, S_1 is a large sub-sub-trees, S_1, S_2 S_3, and S_4, two of which (S_2, and S_3) are duplicated in the inner product computation. It is highly probable for large sub-trees to contain many smaller sub-trees, which lead to an over-estimated similarity value between two parse trees. As mentioned above, therefore, it is necessary to rein those large sub-trees with respect to their sizes in computing kernel values by using decay factors. In this paper, we treat the decay factor as one of the important optimization parameters for a PPI extraction task.

5 Experimental Results

In order to show the superiority of the simple kernel based method using the two factors used in this paper, compared to the resent results for PPIE using additional resources, we ran a series of experiments using the same PPI corpora cited in the literature. In addition, we show that the method is robust especially for cross-corpus experiments where a classifier is trained and tested with entirely different corpora.

5.1 Evaluation Corpora

To evaluate our approach for PPIE, we used “*Five PPI Corpora*[3]” organized by Pyysalo et al. (2008). It contains five different PPI corpora: AImed, BioInfer, HPRD50, IEPA and LLL. They have been combined in a unified XML format and “*binarized*” in case of involving multiple interaction types.

	AIMed	BioInfer	HPRD50	IEPA	LLL
#Sentence	1,955	1,100	145	486	77
#Positive	1,000	2,534	163	335	164
#Negative	4,834	7,132	270	482	166

Table 1. Five PPI Corpora

Table 1 shows the size of each corpus in “*Five PPI Corpora*.” As mentioned before, a sentence can have multiple interactions, which results in the gaps between the number of sentences and the sum of the number of instances. Negative instances have been automatically generated by enumerating sentences with multiple proteins but not having interactions between them (Pyysalo et al., 2008).

5.2 Evaluation Settings

In order to parse each sentence, we used Charniak Parser[4]. For kernel-based learning, we expanded the original *libsvm* 2.89[5] (Chang & Lin, 2001) so that it has two additional kernels including parse tree kernel and composite kernel[6] along with four built-in kernels[7]

Our experiment uses both macro-averaged and micro-averaged *F*-scores. Macro-averaging computes *F*-scores for all the classes individually and takes average of the scores. On the other hand, micro-averaging enumerates both positive results and negative results on the whole without considering the score of each class and computes total *F*-score.

In 10-fold cross validation, we apply the same split used in Airola et al., (2008), Miwa et al., (2009a) and Miwa et al., (2009b) for comparisons. Also, we empirically estimate the regularization parameters of SVM (*C*-values) by conducting 10-fold cross validation on each training data. We do not adjust the SVM thresholds to the optimal value as in Airola et al., (2008) and Miwa et al., (2009a).

5.3 PPI Extraction Performance

Table 2 shows the best scores of our system. The optimal decay factor varies with each corpus. In LLL, the optimal decay factor is 0.2[8] indicating that the shortage of data has forced our system to normalize parse trees more intensively with a strong decay factor in kernel computation in order to cover various syntactic structures.

	DF	*AC*	*ma*-P	*ma*-R	*ma*-F	$\sigma_{ma\text{-F}}$
A	0.6	83.6	72.8 (55.0)	62.1 (68.8)	**67.0** (60.8)	4.5 (6.6)
B	0.5	79.8	74.5 (65.7)	70.9 (71.1)	**72.6** (68.1)	2.7 (3.2)
H	0.7	74.5	75.3 (68.5)	71.0 (76.1)	**73.1** (70.9)	10.2 (10.3)
I	0.6	74.2	74.1 (67.5)	72.2 (78.6)	**73.1** (71.7)	6.0 (7.8)
L	0.2	82.2	83.2 (77.6)	81.2 (86.0)	**82.1** (80.1)	10.4 (14.1)

Table 2. The highest results of the proposed system w.r.t. decay factors. DF: Decay Factor, *AC*: accuracy, *ma*-F: macro-averaged F1, $\sigma_{ma\text{-F}}$: standard deviation of F-scores in CV. A:AIMed, B:BioInfer, H:HPRD50, I:IEPA, L:LLL. The numbers in parentheses refer to the scores of Miwa et al., (2009a).

Our system outperforms the previous results as in Table 2. Even using rich feature vectors including Bag-Of-Words and shortest path trees

[3] http://mars.cs.utu.fi/PPICorpora/eval-standard.html

[4] http://www.cs.brown.edu/people/ec/#software

[5] http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[6] A kernel combining built-in kernels and parse tree kernel

[7] Linear, polynomial, radial basis function, sigmoid kernels

[8] It was determined by increasing it by 0.1 progressively through 10-fold cross validation.

generated from multiple corpora, Miwa et al., (2009b) reported 64.0% and 66.7% in AIMed and BioInfer, respectively. Our system, however, produced 67.0% in AIMed and 72.6% in BioInfer with a single parse tree kernel. We did not have to perform any intensive feature generation tasks using various linguistic analyzers and more importantly, did not use any additional corpora for training as done in Miwa et al., (2009b). While the performance differences are not very big, we argue that obtaining higher performance values is significant because the proposed system did not use any of the additional efforts and resources.

To investigate the effect of the scaling parameter of the parse tree kernel in PPI extraction, we measure how the performance changes as the decay factor varies (Figure 5). It is obvious that the decay factor influences the overall performance of PPI extraction. Especially, the *F*-scores of the small-scale corpora such as HPRD50 and LLL are influenced by the decay factor. The gaps between the best and worst scores in LLL and HPRD50 are 19.1% and 5.2%, respectively. The fluctuation in *F*-scores of the large-scale corpora (AIMed, BioInfer, IEPA) is not so extreme, which seems to stem from the abundance in syntactic and lexical forms that reduce the normalizing effect of the decay factor. The increase in the decay factor leads to the increase in the precision values of all the corpora except for LLL. The phenomenon is fairly plausible because the decreased normalization power causes the system to compute the tree similarities more intensively and therefore it classifies each instance in a strict and detailed manner. On the contrary, the recall values slightly decrease with respect to the decay factor, which indicates that the tree pruning (*PT*) has already conducted the normalization process to reduce the sparseness problem in each corpus.

Most importantly, along with tree pruning, decay factor could boost the performance of our system by controlling the rigidness of the parse tree kernel in PPI extraction.

Table 3 shows the results of the cross-corpus evaluation to measure the generalization power of our system as conducted in Airola et al., (2008) and Miwa et al., (2009a). Miwa et al., (2009b) executed a set of combinatorial experiments by mixing multiple corpora and presented their results. Therefore, it is not reasonable to compare our results with them due to the size discrepancy between training corpora. Nevertheless, we will compare our results with their approaches in later based on AIMed corpus.

As seen in Table 3, our system outperforms the existing approaches in almost all pairs of corpora. In particular, in the multiple corpora-based evaluations aimed at AIMed which has been frequently used as a standard set in PPI extraction, our approach shows prominent results compared with others. While other approaches showed the performance ranging from 33.3% to 60.8%, our approach achieved much higher scores between 55.9% and 67.0%. More specific observations are:

(1) Our PPIE method trained on any corpus except for IEPA outperforms the other approaches regardless of the test corpus only with a few exceptions with IEPA and LLL.

(2) Even when using LLL or HPRD50, two smallest corpora, as training sets, our system performs well with every other corpus for testing. It indicates that our approach is much less vulnerable to the sizes of training corpora than other methods.

(3) The degree of score fluctuation of our system across different testing corpora is much smaller than other regardless of the training data set. When trained on LLL, for example, the range for our system (55.9% ~ 82.1%) is smaller than the others (38.6% ~ 83.2% and 33.3% ~ 76.8%).

(4) The cross-corpus evaluation reveals that our method outperforms the others significantly. This is more visibly shown especially when the large-scale corpora (AIMed and BioInfer) are used.

(5) PPI extraction model trained on AIMed shows lower scores in IEPA and LLL as compared with other methods, which could trigger further investigation.

In order to convince ourselves further the superiority of the proposed method, we compare it with other previously reported approaches. Table 4 lists the macro-averaged precision, recall and *F*-scores of the nine approaches tested on AIMed. While the experimental settings are different as reported in the literature, they are quite close in terms of the numbers of positive and negative documents.

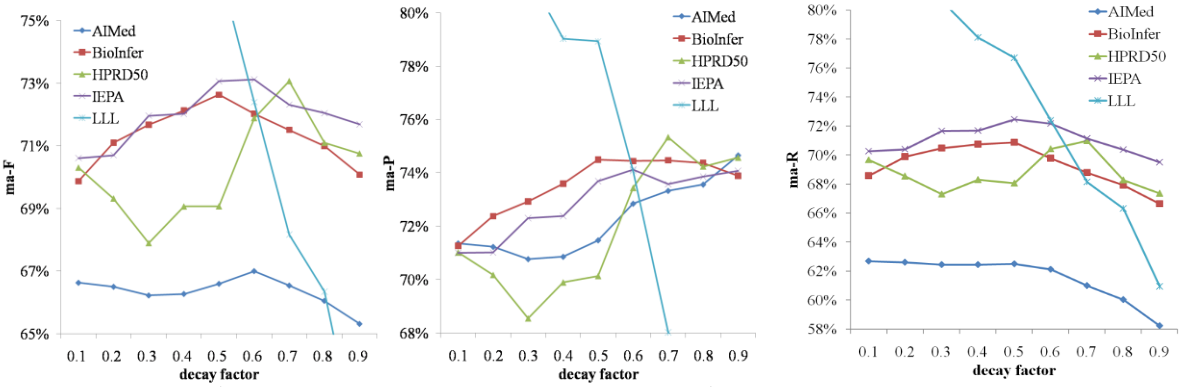


Figure 5. Performance variation with respect to decay factor in Five PPI Corpora. Macro-averaged F1 (left), Precision (middle), Recall (right) evaluated by 10-fold CV

Training corpora	Systems	*F*-Scores in the test corpora				
		AIMed	BioInfer	HPRD50	IEPA	LLL
AIMed	Our System	**67.0**	**64.2**	**72.9**	59.0	62.7
	(Miwa et al., 2009a)	60.8	53.1	68.3	**68.1**	73.5
	(Airola et al., 2008)	56.4	47.1	69.0	67.4	**74.5**
BioInfer	Our System	**65.2**	**72.6**	**71.9**	**72.9**	**78.4**
	(Miwa et al., 2009a)	49.6	68.1	68.3	71.4	76.9
	(Airola et al., 2008)	47.2	61.3	63.9	68.0	78.0
HPRD50	Our System	**63.1**	**65.5**	**73.1**	**69.3**	**73.7**
	(Miwa et al., 2009a)	43.9	48.6	70.9	67.8	72.2
	(Airola et al., 2008)	42.2	42.5	63.4	65.1	67.9
IEPA	Our System	**57.8**	**66.1**	66.3	73.1	78.4
	(Miwa et al., 2009a)	40.4	55.8	66.5	71.7	**83.2**
	(Airola et al., 2008)	39.1	51.7	**67.5**	**75.1**	77.6
LLL	Our System	**55.9**	**64.4**	**69.4**	**71.4**	82.1
	(Miwa et al., 2009a)	38.6	48.9	64.0	65.6	**83.2**
	(Airola et al., 2008)	33.3	42.5	59.8	64.9	76.8

Table 3. Macro-averaged *F*1 scores in cross-corpora evaluation. Rows and columns correspond to the training and test corpora, respectively. We parallel our results with other recently reported results. All the split methods in 10-fold CV are the same for fair comparisons.

As seen in the table, the proposed method is superior to all the others in *F*-scores. The improvement in precision (12.8%) is most significant, especially in comparison with the work of Miwa et al., (2009b), which used multiple corpora (AIMed + IEPA) for training and combined various kernels such as bag-of-words, parse trees and graphs. It is natural that the recall value is lower since a less number of patterns (features) must have been learned. What’s important is that the proposed method has a higher or at least comparable overall performance without additional resources.

Our approach is significantly better than that of Airola et al., (2008), which employed two different forms of graph kernels to improve the initial model. Since they did not use multiple corpora for training, the comparison shows the direct benefit of using the extension of the kernel.

6 Conclusion and Future Works

To improve the performance of PPIE, recent research activities have had a tendency of increasing the complexity of the systems by combining various methods and resources. In this paper, however, we argue that by paying more

	POS	NEG	*ma*-P	*ma*-R	*ma*-F	σ_F
Our System	1,000	4,834	**72.8**	62.1	**67.0**	**4.5**
(Miwa et al., 2009b)	1,000	4,834	60.0	**71.9**	65.2	
(Miwa et al., 2009a)	1,000	4,834	58.7	66.1	61.9	7.4
(Miwa et al., 2008)	1,005	4,643	60.4	69.3	61.5	
(Miyao et al., 2008)	1,059	4,589	54.9	65.5	59.5	
(Giuliano et al., 2006)	-	-	60.9	57.2	59.0	
(Airola et al., 2008)	1,000	4,834	52.9	61.8	56.4	5.0
(Sætre et al., 2007)	1,068	4,563	64.3	44.1	52.0	
(Erkan et al., 2007)	951	4,020	59.6	60.7	60.0	
(Bunescu & Mooney, 2005)	-	-	65.0	46.4	54.2	

Table 4. Comparative results in AIMed. The number of positive instances (POS) and negative instances (NEG) and macro-averaged precision (*ma*-P), recall (*ma*-R) and *F*1-score (*ma*-F) are shown.

attention to a single model and adjusting parameters more carefully, we can obtain at least comparable performance if not better.

This paper indicates that a well-tuned parse tree kernel based on decay factor can achieve the superior performance in PPIE when it is preprocessed by the path-enclosed tree pruning method. It was shown in a series of experiments that our system produced the best scores in single corpus evaluation as well as cross-corpora validation in comparison with other state-of-the-art methods. Contribution points of this paper are as follows:

(1) We have shown that the benefits of using additional resources including richer features can be obtained by tuning a single tree kernel method with tree pruning and decaying factors.

(2) We have newly found that the decay factor influences precision enhancement of PPIE and hence its overall performance as well.

(3) We have also revealed that the parse tree kernel method equipped with decay factors shows superior generalization power even with small corpora while presenting significant performance increase on cross-corpora experiments.

As a future study, we leave experiments with training the classifier with multiple corpora and deeper analysis of what aspects of the corpora gave different magnitudes of the improvements.

Acknowledgment

We want to thank the anonymous reviewers for their valuable comments. This work has been supported in part by KRCF Grant, the Korean government.

Reference

Airola, A., Pyysalo, S., Bjorne, J., Pahikkala, T., Ginter, F. & Salakoski, T. (2008). All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning. BMC Bioinformatics, 9(S2), doi:10.1186/1471-2105-9-S11-S2.

Andrade, M. A. & Valencia, A. (1998). Automatic extraction of keywords from scientific text: application to the knowledge domain of protein families. Bioinformatics, 14(7), 600-607.

Blaschke, C., Andrade, M., Ouzounis, C. & Valencia, A. (1999). Automatic extraction of biological information from scientific text: protein-protein interactions. Proc. Int. Conf. Intell. Syst. Mol. Biol., (pp. 60-67).

Bunescu, R., Ge, R., Kate, R., Marcotte, E., Mooney, R., Ramani, A. & Wong, Y. (2005). Comparative Experiments on Learning Information Extractors for Proteins and their Interactions. Artif. Intell. Med., Summarization and Information Extraction from Medical Documents, 33, 139-155

Collins, M. & Duffy, N. (2001). Convolution Kernels for Natural Language. NIPS-2001, (pp. 625-632).

Craven, M. & Kumlien, J. (1999). Constructing biological knowledge bases by extracting information from text sources. Proceedings of the 7th International conference on intelligent systems for molecular biology, (pp.77-86), Heidelberg, Germany.

Ding, J., Berleant, D., Nettleton, D. & Wurtele, E. (2002). Mining MEDLINE: abstracts, sentences, or phrases?. Proceedings of PSB'02, (pp. 326-337)

Erkan, G., Ozgur, A., & Radev, D. R., (2007). Semi-supervised classification for extracting protein interaction sentences using dependency parsing. In EMNLP 2007.

Fundel, K., Küffner, R. & Zimmer, R. (2007). RelEx – Relation extraction using dependency parse trees. Bioinformatics, 23, 365-371.

Giuliano, C., Lavelli, A., Romano, L., (2006). Exploiting Shallow Linguistic Information for Relation Extraction From Biomedical Literature. Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics.

Gondy, L., Hsinchun C. & Martinez Jesse D. (2003). A shallow parser based on closed-class words to capture relations in biomedical text. J. Biomed. Informatics. 36(3), 145-158.

GuoDong, Z., Min, Z., Dong, H. J. & QiaoMing, Z. (2007). Tree Kernel-based Relation Extraction with Context-Sensitive Structured Parse Tree Information. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Prague, (pp. 728–736)

Marcotte, E. M., Xenarios, I. & Eisenberg D. (2001). Mining literature for protein-protein interactions. Bioinformatics, 17(4), 359-363.

Miwa, M., Sætre, R., Miyao, Y. & Tsujii J. (2009a). Protein-protein interaction extraction by leveraging multiple kernels and parsers. International Journal of Medical Informatics, 78(12), e39-e46.

Miwa, M., Sætre, R., Miyao, Y. & Tsujii J. (2009b). A Rich Feature Vector for Protein-Protein Interaction Extraction from Multiple Corpora. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, (pp. 121-130)

Miwa, M., Sætre, R., Miyao, Y., Ohta, T., & Tsujii, J. (2008). Combining multiple layers of syntactic information for protein-protein interaction extraction. In Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM 2008), (pp. 101–108)

Miyao, Y., Sætre, R., Sagae, K., Matsuzaki, T., & Tsujii, J. (2008). Task-oriented evaluation of syntactic parsers and their representations. Proceedings of the 45th Meeting of the Association for Computational Linguistics (ACL'08:HLT).

Moschitti, A. (2006). Making tree kernels practical for natural language learning. Proceedings of EACL'06, Trento, Italy.

Nikolai, D., Anton, Y., Sergei, E., Svetalana, N., Alexander, N. & llya, M. (2004). Extracting human protein interactions from MEDLINE using a full-sentence parser. Bioinformatics, 20(5), 604-611.

Ono, T., Hishigaki, H., Tanigam, A. & Takagi, T. (2001). Automated extraction of information on protein-protein interactions from the biological literature. Bioinformatics, 17(2), 155-161.

Pyysalo, S., Airola, A., Heimonen, J., Björne, J., Ginter, F. & Salakoski, T. (2008). Comparative analysis of five protein-protein interaction corpora. BMC Bioinformatics, 9(S6), doi:10.1186/1471-2105-9-S3-S6.

Sætre, R., Sagae, K., & Tsujii, J. (2007). Syntactic features for protein-protein interaction extraction. In LBM 2007 short papers.

Sekimizu, T., Park H. S. & Tsujii J. (1998). Identifying the interaction between genes and gene products based on frequently seen verbs in MEDLINE abstracts. Workshop on genome informatics, vol. 9, (pp. 62-71).

Shawe-Taylor, J., Cristianini, N., (2004). Kernel Methods for Pattern Analysis, Cambridge University Press.

Temkin, J. M. & Gilder, M. R. (2003). Extraction of protein interaction information from unstructured text using a context-free grammar. Bioinformatics, 19(16), 2046-2053.

Vishwanathan, S. V. N., Smola, A. J. (2003). Fast Kernels for String and Tree Matching. Advances in Neural Information Processing Systems, 15, 569-576, MIT Press.

Zhang, M., GuoDong, Z. & Aiti, A. (2008). Exploring syntactic structured features over parse trees for relation extraction using kernel methods. Information Processing and Management, 44, 687-701

Zhang, M., Zhang, J., Su, J. & Zhou, G. (2006). A Composite Kernel to Extract Relations between Entities with both Flat and Structured Features. 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, (pp.825-832).

Zhou, D. & He, Y. (2008). Extracting interactions between proteins from the literature. Journal of Biomedical Informatics, 41, 393-407.

An ontology-driven system for detecting global health events

Nigel Collier
National Inst. Informatics
collier@nii.ac.jp

Reiko Matsuda Goodwin
Fordham University
reikogoodwin@gmail.com

John McCrae
Bielefeld University
johnmccrae@gmail.com

Son Doan
Vanderbilt University
son.doan@vanderbilt.edu

Ai Kawazoe
Tsuda College
zoeai@tsuda.ac.jp

Mike Conway
University of Pittsburgh
conwaym@pitt.edu

Asanee Kawtrakul
Kasetart University
ak@ku.ac.th

Koichi Takeuchi
Okayama University
koichi@cs.okayama-u.ac.jp

Dinh Dien
VietNam National University
ddien66@yahoo.com

Abstract

Text mining for global health surveillance is an emerging technology that is gaining increased attention from public health organisations and governments. The lack of multilingual resources such as WordNets specifically targeted at this task have so far been a major bottleneck. This paper reports on a major upgrade to the BioCaster Web monitoring system and its freely available multilingual ontology; improving its original design and extending its coverage of diseases from 70 to 336 in 12 languages.

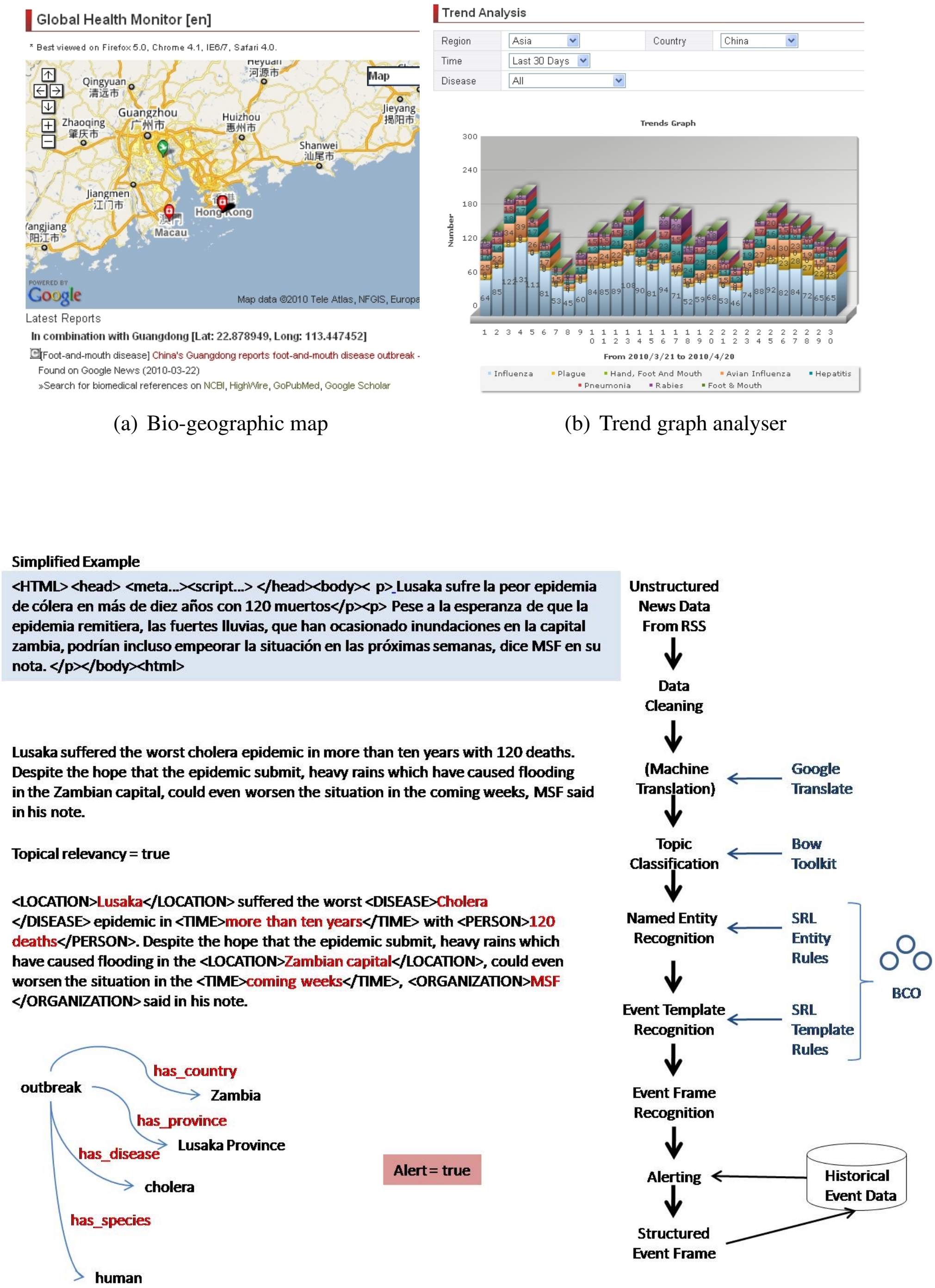
1 Introduction

The number of countries who can sustain teams of experts for global monitoring of human/animal health is limited by scarce national budgets. Whilst some countries have advanced sensor networks, the world remains at risk from the health impacts of infectious diseases and environmental accidents. As seen by the recent A(H5N1), A(H1N1) and SARS outbreaks, a problem in one part of the world can be rapidly exported, leading to global hardship.

The World Health Organization (WHO) estimates that in the future, between 2 to 7.4 million people could be at risk worldwide from a highly contageous avian flu virus that spreads rapidly through the international air travel network (WHO, 2005). Pandemics of novel pathogens have the capacity to overwhelm healthcare systems, leading to widespread morbidity, mortality and socio-economic disruption (Cox et al., 2003). Furthermore, outbreaks of livestock diseases, such as foot-and-mouth disease or equine influenza can have a devastating impact on industry, commerce and human health (Blake et al., 2003). The challenge is to enhance vigilance and control the emergence of outbreaks. Whilst human analysis remains essential to spot complex relationships, automated analysis has a key role to play in filtering the vast volume of data in real time and highlighting unusual trends using reliable predictor indicators.

BioCaster (http://born.nii.ac.jp) (Collier et al., 2008) is a Web 2.0 monitoring station for the early detection of infectious disease events. The system exploits a high-throughput semantic processing pipeline, converting unstructured news texts to structured records, alerting events based on time-series analysis and then sharing this information with users via geolocating maps (Fig. 1(a)), graphs (Fig. 1(b)) and alerts. Underlying the system is a publicly available multilingual application ontology. Launched in 2006 (Collier et al., 2006) the BioCaster Ontology (BCO) has been downloaded by over 70 academic and industrial groups worldwide. This paper reports on a major upgrade to the system and the ontology - expanding the number of languages from 6 to 12, redefining key relations and extending coverage in the number of diseases from 70 to 336, including many veterinary diseases.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 215–222,
Beijing, August 2010



Global Health Monitor [en]

* Best viewed on Firefox 5.0, Chrome 4.1, IE6/7, Safari 4.0.

Latest Reports

In combination with Guangdong [Lat: 22.878949, Long: 113.447452]

[Foot-and-mouth disease] China's Guangdong reports foot-and-mouth disease outbreak -
Found on Google News (2010-03-22)
»Search for biomedical references on NCBI, HighWire, GoPubMed, Google Scholar

(a) Bio-geographic map

Trend Analysis

Region	Asia	Country	China
Time	Last 30 Days		
Disease	All		

(b) Trend graph analyser

Simplified Example

<HTML> <head> <meta...><script...> </head><body>< p>_Lusaka sufre la peor epidemia de cólera en más de diez años con 120 muertos</p><p> Pese a la esperanza de que la epidemia remitiera, las fuertes lluvias, que han ocasionado inundaciones en la capital zambia, podrían incluso empeorar la situación en las próximas semanas, dice MSF en su nota. </p></body><html>

Lusaka suffered the worst cholera epidemic in more than ten years with 120 deaths. Despite the hope that the epidemic submit, heavy rains which have caused flooding in the Zambian capital, could even worsen the situation in the coming weeks, MSF said in his note.

Topical relevancy = true

<LOCATION>Lusaka</LOCATION> suffered the worst <DISEASE>Cholera </DISEASE> epidemic in <TIME>more than ten years</TIME> with <PERSON>120 deaths</PERSON>. Despite the hope that the epidemic submit, heavy rains which have caused flooding in the <LOCATION>Zambian capital</LOCATION>, could even worsen the situation in the <TIME>coming weeks</TIME>, <ORGANIZATION>MSF </ORGANIZATION> said in his note.

(c) BioCaster processes

Figure 1: (a)BioCaster's bio-geographic map for a suspected foot-and-mouth outbreak on 22nd March, 2010 with links to the multilingual ontology, NCBI, HighWire, GoPubMed and Google Scholar; (b) The trends analyser showing aggregated document counts for health events in China between 13nd March and 12th April, 2010; (c) The system's pipeline of processes with example semantic markup.

2 Background

As the world becomes more interconnected and urbanized and animal production becomes increasingly intensive, the speed with which epidemics spread becomes faster, adding to pressure on biomedical experts and governments to make quick decisions. Traditional validation methods such as field investigations or laboratory analysis are the mainstay of public health but can require days or weeks to issue reports. The World Wide Web with its economical and real time delivery of information represents a new modality in health surveillance (Wagner and Johnson, 2006) and has been shown to be an effective source by the World Health Organization (WHO) when Public Health Canada's GPHIN system detected the SARS outbreak in southern China from news reports during November 2002. The recent A(H1N1) 'swine flu' pandemic highlighted the trend towards agencies using unvalidated sources. The technological basis for such systems can be found in statistical classification approaches and light weight ontological reasoning. For example, Google Flu Trends (Ginsberg et al., 2009) is a system that depends almost entirely on automatic statistical classification of user queries; MedISys-PULS (Yangarber et al., 2008), HealthMap (Freifeld et al., 2008) and BioCaster use a mixture of statistical and ontological classification; and GPHIN (Mawudeku and Blench, 2006) and Argus (Wilson, 2007) rely on a mixture of ontological classification and manual analysis.

Compared to other similar systems BioCaster is characterized by its richly featured and publicly downloadable ontology and emphasizes critical evaluation of its text mining modules. Empirical results have included: topic classification, named entity recognition, formal concept analysis and event recognition. In the absence of a community gold standard, task performance was assessed on the best available 'silver' standard - the ProMED-mail network (Madoff and Woodall, 2005), achieving F-score of 0.63 on 14 disease-country pairs over a 365-day period (Collier, 2010).

Despite initial skepticism within the public health community, health surveillance systems based on NLP-supported human analysis of media reports are becoming firmly established in Europe, North America and Japan as sources of health information available to governments and the public (Hartley et al., 2010). Whilst there is no substitute for trained human analysts, automated filtering has helped experts save time by allowing them to sift quickly through massive volumes of media data. It has also enabled them to supplement traditional sources with a broader base of information.

In comparison with other areas of biomedical NLP such as the clinical and genetics' domains, a relative lack of building block resources may have hindered the wider participation of NLP groups in public health applications. It is hoped that the provision of common resources like the BCO can help encourage further development and benchmarking.

3 Method

BioCaster performs analysis of over 9000 news articles per day using the NPACI Rocks cluster middleware (http://www.rockcsclusters.org) on a platform of 48 3.0GHz Xeon cores. Data is ingested 24/7 into a semantic processing pipeline in a short 1 hour cycle from over 1700 public domain RSS feeds such as Google news, the European Media Monitor and ProMED-mail. Since 2009, news has also being gathered under contract from a commercial news aggregation company, providing access to over 80,000 sources across the world's languages.

The new 2010 version of BioCaster uses machine translation into English (eleven languages) to source news stories related to currently occurring infectious and environmental disease outbreaks in humans, animals and plants.

Access to the site is freely available but login registration applies to some functions such as email alerts. Processing is totally automatic, but we have the potential within the login system to enable human moderated alerts which broadcast to Twitter and RSS.

Below we describe in detail two key aspects of the system that have been significantly upgraded: the BCO and the event detection system.

3.1 Ontology

3.1.1 Aim

The BioCaster Ontology aims:

- To describe the terms and relations necessary to detect and risk assess public health events in the grey literature;
- To bridge the gap between (multilingual) grey literature and existing standards in biomedicine;
- To mediate integration of content across languages;
- To be freely available.

The central knowledge source for BioCaster is the multilingual ontology containing domain terms such as diseases, agents, symptoms, syndromes and species as well as domain sensitive relations such as a disease causing symptoms or an agent affecting particular host species. This allows the text mining system to have a basic understanding of the key concepts and relationships within the domain to fill in gaps not mentioned explicitly in the news reports. To the best of our knowledge the BCO is unique as an application ontology, providing freely available multilingual support to system developers interested in outbreak surveillance in the language of the open media.

The BCO however has little to say outside of its application domain, e.g. in disease-gene interaction or for supporting automatic diagnosis. As discussed in Grey Cowell and Smith (2010), there are many other resources available that have the potential to support applications for infectious disease analysis including controlled vocabularies and ontologies such as the the Unified Medical Language System (UMLS) (Lindberg et al., 1993), International Classification of Diseases (ICD-10) (WHO, 2004), SNOMED CT (Stearns et al., 2001), Medical Subject Headings (MeSH) (Lipscomb, 2000) and the Infectious Disease Ontology (IDO) (Grey Cowell and Smith, 2010). In (Collier et al., 2006) we discussed how BCO compared to such ontologies so we will focus from now on the implication of the extensions.

3.1.2 Scope

The new version of the BCO now covers 12 languages including all the United Nation's official languages: Arabic (968 terms), English (4113), French (1281), Indonesian (1081), Japanese (2077), Korean (1176), Malaysian (1001), Russian (1187), Spanish (1171), Thai (1485), Vietnamese (1297) and Chinese (1142). The multilingual ontology can be used as a direct knowledge source in language-specific text mining modules, as an indexing resource for searching across concepts in various languages and as a dictionary for future translation modules. Currently news in all 12 languages is available via the Web portal but news in additional languages such as German, Italian and Dutch are being added using machine translation.

3.1.3 Design

Like EuroWordNet (Vossen, 1998), on which it is loosely based, the BCO adopts a thesaurus-like structure with synonym sets linking together terms across languages with similar meaning. Synonym sets are referred to using *root terms*. Root terms themselves are fully defined instances that provide bridges to external classification schemes and nomenclatures such as ICD10, MeSH, SNOMED CT and Wikipedia. The central backbone taxonomy is deliberately shallow and taken from the ISO's Suggested Upper Merged Ontology (Niles and Pease, 2001). To maintain consistency and computability we kept a single inheritance structure throughout. 18 core domain concepts corresponding to named entities in the text mining system such as DISEASE and SYMPTOM were the results of analysis using a formal theory (Guarino and Welty, 2000).

We have endeavoured to construct definitions for root terms along Aristotelean principles by specifying the difference to the parent. For example in the case of *Eastern encephalitis virus*:

> *Eastern equine encephalitis virus is a species of virus that belongs to the genus Alphavirus of the family Togaviridae (order unassigned) of the group IV ((+)ssRNA) that possesses a positive single stranded RNA genome. It is the*

etiological agent of the eastern equine encephalitis.

We are conscious though that terms used in the definitions still require more rigorous control to be considered useful for machine reasoning. To aid both human and machine analysis root terms are linked by a rich relational structure reflecting domain sensitive relations such as *causes(virus,disease)*, *has_symptom(disease, symptom)*, *has_associated_syndrome(disease, syndrome)*, *has_reservoir(virus, organism).*

In such a large undertaking, the order of work was critical. We proceeded by collecting a list of notifiable diseases from national health agencies and then grouped the diseases according to perceived relevance to the International Health Regulations 2005 (Lawrence and Gostin, 2004). In this way we covered approximately 200 diseases, and then explored freely available resources and the biomedical literature to find academic and layman's terminology to describe their agents, affected hosts, vector species, symptoms, etc. We then expanded the coverage to less well known human diseases, zoonotic diseases, animal diseases and diseases caused by toxic substances such as sarin, hydrogen sulfide, sulfur dioxide and ethylene. At regular stages we checked and validated terms against those appearing in the news media.

As we expanded the number of conditions to include veterinary diseases we found a major structural reorganization was needed to support animal symptoms. For example, a high temperature in humans would not be the same as one in bovids. This prompted us in the new version to group diseases and symptoms around major animal familes and related groups, e.g. *high temperature (human)* and *high temperature (bovine).*

A second issue that we encountered was the need to restructure the hierarchy under *OrganicObject* which was divided between *MicroOrganism* and *Animal*. The structure of the previous version meant that the former were doing double duty as infecting agents and the later were affected hosts. The *MicroOrganism* class contained bacterium, helminth, protozoan, fungus and virus, which then became the domain in a relation 'x causes y'. Expansion forced us to accomodate the fact that some animals such as worms and mites (e.g. scabies) also infect humans as well as animals. The result was a restructuring of the organic classes using the Linnean taxonomy as a guideline, although this is probably not free from errors (e.g. virus is typically not considered to be an organism).

3.2 Event alerting system

Figure 1(c) shows a schematic of the modular design used by the BioCaster text mining system. Following on from machine translation and topic classification is named entity recognition and template recognition which we describe in more detail below. The final structured event frames include slot values normalized to ontology root terms for disease, pathogen (virus or bacterium), country and province. Additionally we also identify 15 aspects of public health events critical to risk assessment such as: spread across international borders, hospital worker infection, accidental or deliberate release, food contamination and vaccine contamination.

Latitude and longitude of events down to the province level are found in two ways: using the Google API up to a limit of 15000 lookups per day, and then using lookup on the BCO taxonomy of 5000 country and province names derived from open sources such as Wikipedia.

Each hour events are automatically alerted to a Web portal page by comparing daily aggregated event counts against historical norms (Collier, 2010). Login users can also sign up to receive emails on specific topics. A topic would normally specify a disease or syndrome, a country or region and a specific risk condition.

In order to extract knowledge from documents, BioCaster maintains a collection of rule patterns in a regular expression language that converts surface expressions into structured information. For example the surface phrase "man exposes airline passengers to measles" would be converted into the three templates "**species(human); disease(measles); international_travel(true)**". Writing patterns to produce such templates can be very time consuming and so the BioCaster project has developed its own

```
D3: :- name(disease){ list(@undiagnosed) words(,1) list(@disease) }
S2: :- name(symptom) { list(@severity) list(@symptom)}
CF1: contaminated_food("true") :- "caused" "by" list(@contaminate_verbs_past)
list(@injested_material)
SP4: species("animal") :- name(animal,A) words(,3) list(@cull_verbs_past)
```

Table 1: Examples of SRL rules for named entity and template recognition. Template rules contain a label, a head and a body, where the head specifies the template pattern to be output if the body expression matches. The body can contain word lists, literals, and wild cards. Various conditions can be placed on each of these such as orthographic matching.

light weight rule language - called the Simple Rule Language (SRL) and a pattern building interface for maintaining the rule base (McCrae et al., 2009). Both are freely available to the research community under an open source license. Currently BioCaster uses approximately 130 rules for entity recognition, 1000 word lists and 3200 template rules (of which half are for location recognition) to identify events of interest in English. Using SRL allows us to quickly adapt the system to newly emerging terminology such as the 11+ designations given to A(H1N1) during the first stages of the 2009 pandemic.

The SRL rulebook for BioCaster can recognize a range of entities related to the task of disease surveillance such as bacteria, chemicals, diseases, countries, provinces, cities and major airports. Many of these classes are recognized using terms imported from the BCO. The rule book also contains specialised thesauri to recognize subclasses of entities such as locations of habitation, eateries and medical service centres. Verb lists are maintained for lexical classes such as detection, mutation, investigation, causation, contamination, culling, blaming, and spreading.

Some examples of SRL rules for named entity recognition are shown in Table 1 and described below:

Rule D3 in the rulebook tags phrases like 'mystery illness' or 'unknown killer bug' by matching on strings contained within two wordlists, @undiagnosed and @disease, separated by up to one word.

Rule S2 allows severity indicators such as 'severe' or 'acute' to modify a list of known symptoms in order to identify symptom entities.

Rule CF1 is an example of a template rule. If the body of the rule matches by picking out expressions such as 'was caused by tainted juice', this triggers the head to output an alert for contaminated food.

Rule SP4 identifies the victim species as 'animal' in contexts like '250 geese were destroyed'.

The rulebook also supports more complex inferences such as the home country of national public health organizations.

Since BioCaster does not employ systematic manual checking of its reports, it uses a number of heuristic filters to increase specificity (the proportion of correctly identified negatives) for reports that appear on the public Web portal pages. For example, reports with no identified disease and country are rejected. Since these heuristics may reduce sensitivity they are not applied to news that appears on the user login portal pages.

4 Results and Discussion

Version 3 of the ontology represents a significant expansion in the coverage of diseases, symptoms and pathogens on version 2. Table 2 summarizes the number of root terms for diseases classified by animal familes.

The thesaurus like structure of the BCO is compatible in many respects to the Simple Knowledge Organization System (SKOS) (Miles et al., 2005). In order to extend exchange and re-use we have produced a SKOS version of the BCO which is available from the BCO site. We have also converted the BCO terms into 12 SRL rule books (1 for each language) for entity tagging. These too are freely available from the BCO site.

As the ontology expands we will consider adopting a more detailed typing of diseases such as *hasInfectingPart* to indicate the organ affected

Species	N	Example
Avian	22	Fowl pox
Bee	6	Chalk brood disease
Bovine	24	Bluetongue
Canine	4	Blastomycosis (Canine)
Caprine	14	Contagious agalactia
Cervine	2	Chronic wasting disease
Equine	17	Strangles
Feline	4	Feline AIDS
Fish	2	Viral hemorr hagic septicemia
Human	216	Scarlet fever
Lagomorph	2	Myxomatosis
Non-human primate	16	Sylvan yellow fever
Other	2	Crayfish plague
Rodent	8	Colorado tick fever (Rodent)
Swine	12	Swine erysipelas

Table 2: Major disease groups organized by affected animal family. N represents the number of root terms.

or *hasProtectionMethod* to indicate broad classes of methods used to prevent or treat a condition. The typology of diseases could also be extended in a more fine grained manner to logically group conditions, e.g. *West Nile virus encephalitis*, *Powassan encephalitis* and the *Japanese B encephalitis* could be connected through a *hasType* relation on *encephalitis*.

5 Conclusion

Multilingual resources specifically targeted at the task of global health surveillance have so far been very rare. We hope that the release of version 3 can be used to support a range of applications such as text classification, cross language search, machine translation, query expansion and so on.

The BCO has been constructed to provide core vocabulary and knowledge support to the BioCaster project but it has also been influential in the construction of other public health oriented application ontologies such as the Syndromic Surveillance Ontology (Okhamatovskaia et al., 2009). The BCO is freely available from http://code.google.com/p/biocaster-ontology/ under a Creative Commons license.

Acknowledgements

The authors greatly acknowledge the many coworkers who have provided comments and feedback on BioCaster. Funding support was provided in part by the Japan Science and Technology Agency under the PRESTO programme.

References

Blake, A., M. T. Sinclair, and G. Sugiyarto. 2003. Quantifying the impact of foot and mouth disease on tourism and the UK economy. *Tourism Economics*, 9(4):449–465.

Collier, N., A. Kawazoe, L. Jin, M. Shigematsu, D. Dien, R. Barrero, K. Takeuchi, and A. Kawtrakul. 2006. A multilingual ontology for infectious disease surveillance: rationale, design and challenges. *Language Resources and Evaluation*, 40(3–4). DOI: 10.1007/s10579-007-9019-7.

Collier, N., S. Doan, A. Kawazoe, R. Matsuda Goodwin, M. Conway, Y. Tateno, Q. Ngo, D. Dien, A. Kawtrakul, K. Takeuchi, M. Shigematsu, and K. Taniguchi. 2008. BioCaster:detecting public health rumors with a web-based text mining system. *Bioinformatics*, 24(24):2940–1, December. doi:10.1093/bioinformatics/btn534.

Collier, N. 2010. What‘s unusual in online disease outbreak news? *Biomedical Semantics*, 1(1), March. doi:10.1186/2041-1480-1-2.

Cox, N., S. Temblyn, and T. Tam. 2003. Influenza pandemic planning. *Vaccine*, 21(16):1801–1803.

Freifeld, C., K. Mandl, B. Reis, and J. Brownstein. 2008. Healthmap: global infectious disease monitoring through automated classification and visualization of internet media reports. *J. American Medical Informatics Association*, 15:150–157.

Ginsberg, J., M. Mohebbi, R. Patel, L. Brammer, M. Smolinski, and L. Brilliant. 2009. Detecting influenza epidemics using search engine query data. *Nature*, 457:1012–1014.

Grey Cowell, L. and B. Smith. 2010. Infectious disease informatics. In Sintchenko, V., editor, *Infectious Disease Informatics*, pages 373–395. Springer New York.

Guarino, N. and C. Welty. 2000. A formal ontology of properties. In Dieng, R. and O. Corby, editors, *EKAW-2000: Proc. 12th Int. Conf. on Knowledge Engineering and Knowledge Management*, pages 97–112.

Hartley, D., N. Nelson, R. Walters, R. Arthur, R. Yangarber, L. Madoff, J. Linge, A. Mawudeku, N. Collier, J. Brownstein, G. Thinus, and N. Lightfoot. 2010. The landscape of international biosurveillance. *Emerging Health Threats J.*, 3(e3), January. doi:10.1093/bioinformatics/btn534.

Lawrence, O. and J. Gostin. 2004. International infectious disease law - revision of the World Health Organization's international health regulations. *J. American Medical Informatics Association*, 291(21):2623–2627.

Lindberg, Donald A.B., L. Humphreys, Betsy, and T. McCray, Alexa. 1993. The unified medical language system. *Methods of Information in Medicine*, 32:281–291.

Lipscomb, C. 2000. Medical subject headings (MeSH). *Bulletin of the Medical Library Assocation*, 88:256–266.

Madoff, Lawrence C. and John P. Woodall. 2005. The internet and the global monitoring of emerging diseases: Lessons from the first 10 years of promedmail. *Archives of Medical Research*, 36(6):724 – 730. Infectious Diseases: Revisiting Past Problems and Addressing Future Challenges.

Mawudeku, A. and M. Blench. 2006. Global public health intelligence network (gphin). In *Proc. 7th Int. Conf. of the Association for Machine Translation in the Americas, Cambridge, MA, USA*, August 8–12.

McCrae, J., M. Conway, and N. Collier. 2009. Simple rule language editor. Google code project, September. Available from: http://code.google.com/p/srl-editor/.

Miles, A., B. Matthews, and M. Wilson. 2005. SKOS Core: Simple knowledge organization for the web. In *Proc. Int. Conf. on Dublin Core and Metadata Applications, Madrid, Spain*, 12–15 September.

Niles, I. and A. Pease. 2001. Towards a standard upper ontology. In Welty, C. and B. Smith, editors, *2nd Int. Conf. on Formal Ontology in Information Systems FOIS-2001, Maine, USA*, October 17–19.

Okhamatovskaia, A., W. Chapman, N. Collier, J. Espino, and D. Buckeridge. 2009. SSO: The syndromic surveillance ontology. In *Proc. Int. Soc. for Disease Surveillance, Miami, USA*, December 3–4.

Stearns, M. Q., C. Price, K. A. Spackman, and A. Y. Wang. 2001. SNOMED clinical terms: overview of the development process and project status. In *Proc. American Medical Informatics Association (AMIA) Symposium*, pages 662–666.

Vossen, P. 1998. Introduction to EuroWordNet. *Computers and the Humanities*, 32:73–89.

Wagner, M. and H. Johnson. 2006. The internet as sentinel. In Wagner, M. et al., editor, *The Handbook of Biosurveillance*, pages 375–385. Academic Press.

WHO. 2004. *ICD-10, International Statistical Classification of Diseases and Related Health Problems, Tenth Revision*. World Health Organization, December.

WHO. 2005. Avian influenza: assessing the pandemic threat. Technical Report WHO/CDS/2005.29, World Health Organization, Geneva, January.

Wilson, J. 2007. Argus: a global detection and tracking system for biological events. *Advances in Disease Surveillance*, 4.

Yangarber, R., P. von Etter, and R. Steinberger. 2008. Content collection and analysis in the domain of epidemiology. In *Proc. Int. Workshop on Describing Medical Web Resources (DRMED 2008), Gotenburg, Sweden*, May 27th.

Constraining robust constructions for broad-coverage parsing with precision grammars

Bart Cramer[†] **and Yi Zhang**[†‡]
Department of Computational Linguistics & Phonetics, Saarland University[†]
LT-Lab, German Research Center for Artificial Intelligence (DFKI)[‡]
{bcramer,yzhang}@coli.uni-saarland.de

Abstract

This paper addresses two problems that commonly arise in parsing with precision-oriented, rule-based models of grammar: lack of speed and lack of robustness. First, we show how we can reduce parsing times by restricting the number of tasks the parser will carry out, based on a generative model of rule applications. Second, we show that a combination of search space restriction and radically overgenerating robustness rules lead to a more robust parser, with only a small penalty in precision. Applying both the robustness rules and a fragment fallback strategy showed better recall than just giving fragment analyses, with equal precision. Results are reported on a medium-sized HPSG grammar for German. [1]

1 Introduction

In the field of natural language processing, it is common wisdom that handwritten, rule-based models generally perform poorly on complex problems, mainly due to the knowledge acquisition bottleneck: it is hard for the human modeller to conceive of all possible scenarios the model has to cope with. In parsing, many approaches have relied on hand-written grammars, and their fragility is one of their largest weaknesses. Such models can fail due to insufficiency of lexical entries or grammatical constructions, but also due to creative or ungrammatical input. In any case, the parser should always return a reasonable output. A very simple technique is partial or fragment parsing (Kiefer et al., 1999; Riezler et al., 2001; Zhang et al., 2007a): if there is no item in the chart that both spans the complete sentence and fulfills the root condition, several chunks that do conform to a root condition are combined by minimising a certain cost function (for instance to favour larger chunks, or more probable chunks).

A second problem with deep parsers is their relatively low efficiency. For online applications, it is impermissible to wait for longer than a minute before the system responds. Apart from studies that were aimed at increasing the efficiency of deep parsers by using smarter algorithms (e.g. using left-corner relations (Van Noord, 1997)), several studies in recent years have suggested that search space restriction can offer a beneficial balance between speed and accuracy as well. Techniques that have been proposed are, among others, supertagging (Clark and Curran, 2007), CFG filtering (Matsuzaki et al., 2007) and beam thresholding (Ninomiya et al., 2005).

A potential disadvantage of the latter technique is that the unifications have taken place by the time the value of the resulting chart item is investigated. One strategy that tries to prevent execution of unlikely tasks altogether is presented by van Noord (2009). In this method, the parser learns from an unannotated corpus which parse steps contributed to the solution as preferred by the disambiguation model (as opposed to a certain gold standard). Hence, this approach is self-learning.

Another study that is close to our approach

[1] The research reported on in this paper has been carried out with financial support from the Deutsche Forschungsgemeinschaft and the German Excellence Cluster of Multimodal Computing & Interaction.

to search space restriction is c-structure pruning (Cahill et al., 2008). The authors show that a large, hand-written, unification-based parser (the XLE LFG parser for English) can perform reasonably faster (18%) without losing accuracy, by not allowing the parser to unify if the resulting item will have a span that does not conform to a CFG tree that was generated from the sentence beforehand by a PCFG parser. Much better results (67% speed-up) are obtained by pruning chart items locally, based on their relative probabilities (Cahill et al., 2008). This is the approach that is closest to the one we present in this paper.

In this paper, we introduce a method that addresses robustness and efficiency concurrently. The search space is restricted by setting a maximum on the number of tasks per chart cell. Because tasks are carried out according to a priority model based on the generative probabilities of the rule applications, it is unlikely that good readings are dropped. More robustness is achieved by adding radically overgenerating rules to the grammar, which could cover all sentences, given an disproportionate amount of time and memory. By strongly restricting the search space, however, the computation requirements remains within bounds. Because the robustness rules are strongly dispreferred by both the priority model and the disambiguation model, all sentences that would be covered by the 'restricted' grammar remain high-precision, but sentences that are not covered will get an additional push from the robustness rules.

1.1 An HPSG grammar for German

The grammar we use (Cramer and Zhang, 2009) is the combination of a hand-written, constraint-based grammar in the framework of HPSG and an open word class lexicon extracted from the Tiger treebank (Brants et al., 2002) in a deep lexical acquisition step. One of the aims of this grammar is to be precision-oriented: it tries to give detailed analyses of the German language, and reject ungrammatical sentences as much as possible. However, this precision comes at the cost of lower coverage, as we will see later in this paper.

Along with the grammar, a treebank has been developed by re-parsing the Tiger treebank, and including those sentences for which the grammar was able to reproduce the original Tiger dependencies. The treebank's size is just over 25k sentences (only selected from the first 45k sentences, so they don't overlap with either the development or test set), and contains the correct HPSG derivation trees. These (projective) derivation trees will function as the training set for the statistical models we develop in this study.

2 Restriction of the search space

2.1 The PET parser

The parser we employ, the PET parser (Callmeier, 2000), is an agenda-driven, bottom-up, unification-based parser. In order to reduce computational demands, state-of-the-art techniques such as subsumption-based packing (Oepen and Carroll, 2000) and the quasi-destructive unification operator (Tomabechi, 1991) have been implemented.

A central component in the parser is the agenda, implemented as a priority queue of parsing tasks (unifications). Tasks are popped from the agenda, until no task is left, after which all passive items spanning the complete sentence are compared with the root conditions as specified by the grammar writer. The best parse is extracted from the parse forest by a Maximum Entropy parse disambiguation model (Toutanova et al., 2002), using selective unpacking (Zhang et al., 2007b).

Two different types of items are identified: passive items and active items. Passive items are 'normal' chart items, in the sense that they can freely combine with other items. Active items still need to combine with a passive item to be complete. Hence, the parser knows two types of tasks as well (see figure 1): *rule+passive* and *active+passive*.

Each time a task succeeds, the following happens:

- For each inserted passive item, add (rule+passive) tasks that combine the passive item with each of the rules, and add (active+passive) tasks that combine with each of the neighbouring active items.
- For each inserted active item, add (active+passive) tasks that combine the remain-

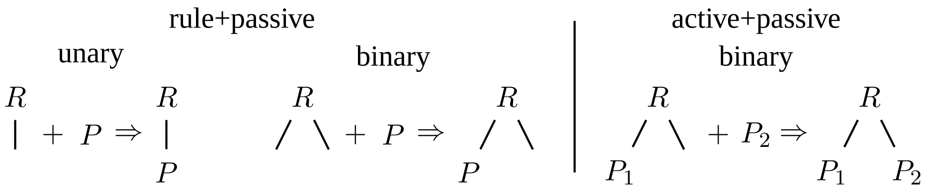


Figure 1: Depicted are the different types of tasks in the PET parser. Not shown are the features structures imposed by the rules and the chart items.

ing gaps in the active item with existing neighbouring passive items in the chart.

2.2 Defining priorities

The priorities of the parsing tasks are calculated based on a generative PCFG model extracted from the treebank by maximum likelihood estimation, smoothed by Lidstone smoothing. Each passive chart item receives a score based on its generative probability, calculated as the product of all applied rule probabilities. For active parsing items, we set the score to be the upper bound of this generative probability, if the item succeeds later in combining with other passive edge(s) to build a complete subtree. This is done by simply assuming the undetermined subtree in the active item receiving a generative score of 1.

The priorities that are assigned to both types of tasks are not yet conditioned on the probability of the topmost rule application. Hence, they are computed using the following simple formula:

$$Pr = p(R) \cdot p(P)$$

where Pr is the task's priority, $p(R)$ the prior probability of the rule category R; and $p(P)$ is the highest possible generative probability of the *resulting* passive item P.

2.3 Restriction strategies

It is a natural thought to allocate more computational resources to longer sentences, and this is exactly what happens in the restriction strategies we develop in this study. We define a cap on the number of tasks for a certain cell/span (i, j), which means that the number of cells is quadratically related to the number of words in a sentence: $n_{cells} = n(n + 1)/2$.

We define three task restriction strategies: *all*, *success*, and *passive*. In *all*, the cap is defined for all tasks, whether the unification is successful or not. *Success* only counts tasks that are successful (i.e. lead to either an active or a passive item), and *passive* only counts tasks that lead to a passive item. In all strategies, morphological and lexical tasks are not counted, and hence not restricted. Unary phrasal rules (such as *empty-det*) are counted, though.

The implementation uses only one priority queue. Each time a task is popped from the agenda, it is checked whether the limit for this span has been reached or not. If so, the task is discarded; otherwise, it is executed.

2.4 Methodology

All our experiments are based on the Tiger treebank (Brants et al., 2002). The grammar's lexicon is based on the first 45k sentences in the treebank, and so are the MaxEnt disambiguation model (Toutanova et al., 2002) and the generative model we developed for this study. The development set (s45001-s47500) was used to finetune the methods, but all final results presented in this paper are with respect to the test set (s47501-s50000). The maximum time for building up the packed parse forest is 60 seconds, after which unpacking is started. Unpacking the first reading usually has negligible computation costs, and is not reported on. Along with the best reading's derivation, the dependencies are output, and com-

Strategy Cap size	exhaustive	all 3000	success 200	passive 100
Time (s)	7.20	1.04	0.92	1.06
Coverage	59.4%	60.5%	60.0%	59.0%
Exact	17.6%	17.6%	17.4%	17.4%
Recall	37.6%	39.5%	38.9%	38.0%
Precision	80.7%	80.3%	80.1%	80.4%
F-score	51.3%	52.9%	52.4%	51.6%

Table 1: A more detailed look into some data points from figure 2. 'Coverage' and 'Exact' are sentential percentages, showing how many sentences receive at least one or the exactly correct reading. Recall, precision and f-score are on a per-dependency basis.

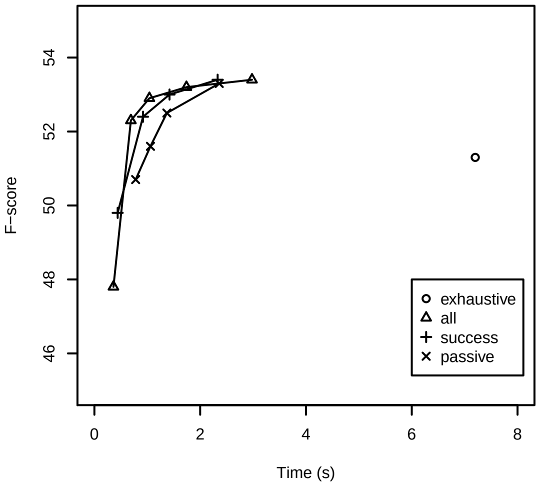


Figure 2: This figure shows the tradeoff between speed and f-score for the standard grammar, using the restriction strategies with different cap sizes.

pared to the gold standard dependencies from the Tiger treebank.

2.5 Results

The results of the experiments, with different cap sizes, are summarized in table 1 and figure 2. As expected, for all strategies it holds that longer computation times lead to higher coverage numbers. The interesting thing is that the restriction of the search space doesn't affect the parses' precision, indicating that the priorities work well: the tasks leading to good solutions are indeed given high priority scores.

A striking observation is that the coverage numbers go up by about 1%, with reductions in parse times of more than 80%. This is due to the use of the timeout, and the generic tendency of our definition of the priorities: because less rule applications lead to higher log probabilities, the agenda will favour tasks with smaller span size. If the agenda doesn't apply too strong a restriction on those tasks, the parser might not create any items spanning the whole sentence after the full 60 seconds, and hence produce no parse. This is mitigated by stronger restriction, leading to a quicker path upwards in the chart.

No large differences of success are found between the different strategies. The intuition behind the *success* and *passive* strategies was that only more effort should be invested into a particular span if not enough chart items for that span have been created. However, the time/quality trade-offs are very similar for all strategies, as shown in figure 2[2].

The strategies we have reported on have one thing in common: their counters are with respect to one particular span, and therefore, they have a very local scope. We have tried other strategies that would give the algorithm more flexibility by defining the caps on more global scale, for instance per span length or for the entire chart. However, this degraded the performance severely, because the parser was not able to divide its attention properly.

[2] One might be tempted to consider the *all* strategy as the best one. However, the time/f-score tradeoff curves look slightly different on the development set.

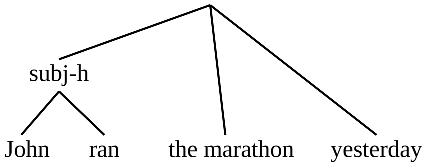
3 Increasing robustness

For hand-written deep parsers, efficiency and coverage are often competing factors: allowing more items to be created might be beneficial for recall, but the parser will also be too slow. However, because the search space can be restricted so rigidly, we can make the grammar more permissive to accept more sentences, hopefully without a heavy efficiency penalty. One way to do this is to remove constraints from the grammar rules. However, that would infringe on the precision-oriented nature of the grammar. Instead, we will keep the normal grammar rules as they are, and create a small number of additional, super-accepting *robustness rules*. The intuition is that when the restricted part of the grammar can find a solution, that solution will indeed be found, and preferred by the statistical models. On the other hand, when the sentence is extragrammatical, the robustness rules may be able to overcome the barriers.

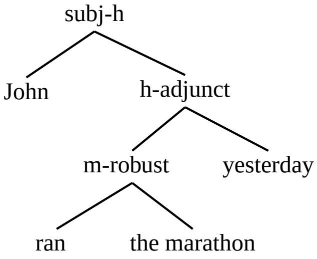
Let's consider the following example, assuming that the grammar only lists 'to run' as an intransitive verb:

'John ran the marathon yesterday'

A fragment approach would come up with the following solution:



'John' will correctly be identified as the subject of 'ran', but that is all. No dependencies are established between 'the marathon' and 'ran', or 'yesterday' and 'ran'. The former is hard to establish, because of the missing lexical item. However, the latter should be doable: the lexicon knows that 'yesterday' is an adverb that modifies verbs. If we could create a robustness rule that would absorb the object ('the marathon') without assigning a dependency, it would at least be able to identify the modifier dependency between 'ran' and 'yesterday'.



In other words, a fragment analysis solely combines items at the top level, whereas a robust parser would ideally be able to overcome barriers in both the lower and the higher regions of the chart, meaning that the damage can be localised and thus minimised. The robustness rules we propose are intended to achieve that.

How does this idea interact with the restriction mechanism explained in the previous section? Robustness rules get an inhibitively large, constant penalty in both the priority model and the disambiguation model. That means that at first the parser will try to build the parse forest with the restricted set of rules, because tasks involving subtrees with only rules from the standard grammar will always have a higher priority than tasks using an item with a robustness rule application in its subtree. When this is finished, the robustness rules try to fill the gaps. Especially in the *success* and *passive* strategies, tasks with robustness rules are discarded if already enough chart items are found for a particular span, meaning that the parser automatically focusses on those parts of the chart that haven't been filled before.

3.1 Defining robustness rules

Defining robustness rules is a sort of grammar engineering, and it took a bit of experimentation to find rules that worked well. One of the factors was the interaction between the subsumption-based packing and the robustness rules. When the chart is built up, items that are subsumed by an existing item are marked as 'frozen', and the latter (more general) item functions as the representative node in the remainder of the parsing process. When unpacking the best solution, the best derivation tree is extracted from the packed forest, which might include a frozen node. Because this frozen node has more constraints than its representative, this derivation tree is not guaranteed to be free of unification failures, and hence, before outputting, this is checked by replaying all the unifications in the derivation tree. This procedure is repeated until a sound derivation has been found.

So what happens when the representative nodes are very general? Many nodes will be packed, and hence the chart will remain compact. However, the unpacking process will become problematic, because many of the proposed derivation trees during unpacking will be incorrect, leading to excessive computation times (in the order of minutes).

Therefore, we chose to define robustness rules such, that the resulting chart items will be equally constrained as their daughters. They are all binary, and have one common ancestor in the type hierarchy:

$$\left[\begin{array}{ll} \multicolumn{2}{l}{\textit{structure-robust}} \\ \text{SYNSEM} & \boxed{1} \\ \text{ROBUST} & + \\ \text{MN-DTR} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{sign}} \\ \text{SYNSEM} & \boxed{1}\left[\text{LOCAL.CAT.HEAD}\ \ \textit{verb}\right] \\ \text{ROBUST} & - \end{array}\right] \\ \text{RB-DTR} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{sign}} \\ \text{SYNSEM} & \left[\text{NONLOCAL}\ \ \textit{no-nonlocal}\right] \\ \text{ROBUST} & - \end{array}\right] \end{array}\right]$$

All rules have a *main* daughter and a *robust* daughter. The co-indexation of the SYNSEM of the main daughter and the SYNSEM of the rule itself has the effect that the resulting chart item will have the exact same syntactic properties as its main daughter, whereas the robust daughter does not contribute to the syntactic properties of the mother node. The ROBUST feature is used to prevent the application of two robust rules consecutively. Additional constraints (not shown) make sure that morphological processing is finished, and that both parts are not involved in a coordination. Robustness rules do not yield a dependency triple (although they mght be guessed accurately by a few heuristics).

We define two pairs of robustness rules, each pair consisting of a rule with MN-DTR first and RB-DTR second, and one rule in the other order:

+V The robust daughter is a verb, which is still allowed to have valence, but cannot have any features in NONLOCAL.

+NV The robust daughter is anything but a verb, cannot have any non-empty valence list, and cannot have any features in NONLOCAL.

3.2 Fragment parsing

As a baseline for comparison, we investigate the existing partial parsing algorithms that pick fragmented analyses from the parse forest as a fallback strategy when there is no full parse available. Kiefer et al. (1999) took a shortest-path approach to find a sequence of fragment analysis that minimizes a heuristics-based cost function. Another variation of the algorithm (Riezler et al., 2001) is to pick fewest chunks that connect the entire sentence. While these early approaches are based on simple heuristics, more sophisticated parse selection methods also use the statistical models to rank the partial analyses. For example, Zhang et al. (2007a) proposed several ways of integrating discriminative parse ranking scores with the partial parse selection algorithm.

In this experiment, we first use the shortest path algorithm to find candidate chunks of partial analysis. All phrasal constituents were given equal weights, and preferred over input and lexical edges. For each chunk (edges spanning the same sub-string of the input sentence), the edge with the highest generative probability is picked. Consequently, the best partial reading (covering that edge) is decoded by the selective unpacking algorithm using the MaxEnt parse ranking model. With each fragment, the partial semantic representations were extracted. Similar to the robustness rules, no cross-fragment dependencies are recovered in this approach. Due to the limited number of chart items and the use of selective unpacking, the computation times for the shortest-path algorithm are marginal.

3.3 Results

The results of this experiment are listed in table 2. For the robust versions of the grammar, no exhaustive parsing results are reported, because they take too long to compute, as can be expected. Coverage number are on a per-sentence

		standard		+V	+NV	+V+NV
		exhaustive	restricted	restricted		
	time (s)	7.20	0.92	4.10	1.42	4.09
no fragment	coverage	59.3%	60.0%	72.6%	69.9%	78.6%
	recall	37.6%	38.9%	48.4%	47.0%	53.8%
	precision	80.7%	80.1%	78.6%	78.2%	77.7%
	f-score	51.3%	52.4%	59.9%	58.7%	63.6%
fragment	coverage	94.3%	98.3%	98.5%	98.7%	98.5%
	recall	50.4%	53.6%	59.5%	56.9%	61.3%
	precision	75.4%	75.0%	75.0%	74.5%	74.7%
	f-score	60.4%	62.5%	66.3%	64.5%	67.3%

Table 2: Results for experiments with different robustness rules, and with or without fragment fallback strategy.

basis, whereas the other percentages are on a per-dependency basis. Time denotes the average number of seconds it takes to build the parse forest. All results under ‘restricted’ are carried out with the *success* strategy, with a cap of 200 tasks (*success-200*). ‘(No) fragment’ indicates whether a fragment parse is returned when no results are obtained after selective unpacking.

The robustness rules significantly increase the sentential coverage, in the case of +V+NV almost 20 percent points. The gains of +V and +NV are fairly additive: they seem to cover different sets of extragrammatical sentences. In the most permissive setting (+V+NV), dependency recall goes up by 16 percent point, with only a 3 percent point decrease of precision, showing that the newly-covered sentences still receive fairly accurate parses. Also, it can be seen that the +V pair of rules is more effective than +NV to increase coverage. The robust grammars are certainly slower than the standard grammar, but still twice as fast as the standard grammar in an exhaustive setting.

Coverage numbers are approximating 100% when the fragment parsing fallback strategy is applied, in all settings. However, it is interesting to see that the recall numbers are higher when the robustness rules are more permissive, but that no significant effect on the precision is observed. This suggests that the lumps that are connected by the fragment parsing mechanism are larger, due to previous applications of the robustness rules. From this, we conclude that the connections made by the robustness rules are of relatively high quality.

We have also tried the *all*-3000 and *passive*-100 settings (the same as listed in table 1). That yielded very similar results, except on the grammar with both +V and +NV enabled. With *passive*-100, there was a small decrease in coverage (76.0%), but this drop was much more pronounced for *all*-3000: 72.0%. This suggests that, if the pressure on the generative model is larger due to heavier overgeneration, counting successful tasks or passive items performs better than just counting the number of executed tasks.

After manual inspection, we found out that the kind of constructions the robustness rules created were very diverse. Most of the rule applications were not in the top of the tree, as was intended. There also seemed to be a correlation between the length of the robust daughter and the quality of the parse. When the robust daughter of the rule was large, the application of the robustness rule looked like an emergency strategy, with a corresponding quality of the parse. However, when the robustness rule connects a verb to a relatively small constituent (a particle or an NP, for example), the resulting derivation tree was of reasonable quality, keeping most of the other dependencies intact.

4 Discussion

Achieving broad coverage in deep parsing while maintaining high precision is difficult. Until now, most existing hand-written grammar-based parsing systems rely on fragment analyses (or various ways of putting fragments together to compose partial readings), but we argued (with the example in section 3) that such an approach delivers inferior results when the tree falls apart at the very bottom. The use of robust constructions offers a way to keep the damage local, but can create an intractable search space. The proposed pruning strategies carefully control the bound of overgeneration, resulting in improvements on both parsing efficiency and coverage, with a significantly smaller degradation in f-score than a pure fragment approach. The combination of grammar engineering, statistical modelling and algorithmic design in the parser brings the parser performance to a new level.

Although the experiments were carried out on a specific grammar framework, we consider the techniques put forward in this paper to be applicable to other linguistic frameworks. The robustness rules are easy to construct (with the precautions from section 3.1 in mind), and all modern deep parsers have a treebank to their disposal, from which the generative model can be learned.

There are still points that can be improved on. Currently, there is no way to determine which of the *robust* rule applications are more promising than others, and the decision to try one before the other is solely based on the the probabilities of the passive items, and not on the generative model. This can be inefficient: for instance, all robustness rules presented in this paper (both +V and +NV) requires the main daughter to be a verb. It would be straightforward to learn from a small treebank that trying to unify the main daughter of a robustness rules (which should have a verbal head) with a specifier-head rule application does not have a high chance on succeeding.

Another possible improvement is to differentiate between different robustness rules. We presented a two-tier system here, but the framework lends itself naturally to more layers with differing degrees of specificity, creating a smoother scale from specific/prioritised to robust/non-prioritised.

References

Brants, S., S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER Treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, pages 24–41.

Cahill, A., J.T. Maxwell III, P. Meurer, C. Rohrer, and V. Rosén. 2008. Speeding up LFG parsing using c-structure pruning. In *Proceedings of the Workshop on Grammar Engineering Across Frameworks*, pages 33–40. Association for Computational Linguistics.

Callmeier, U. 2000. PET–a platform for experimentation with efficient HPSG processing techniques. *Natural Language Engineering*, 6(01):99–107.

Clark, S. and J.R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Cramer, B. and Y. Zhang. 2009. Construction of a German HPSG grammar from a detailed treebank. In *Proceedings of the GEAF workshop ACL-IJCNLP 2009*, pages 37–45.

Kiefer, B., H.U. Krieger, J. Carroll, and R. Malouf. 1999. A bag of useful techniques for efficient and robust parsing. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 473–480. Association for Computational Linguistics.

Matsuzaki, T., Y. Miyao, and J. Tsujii. 2007. Efficient HPSG parsing with supertagging and CFG-filtering. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1671–1676, Hyderabad, India.

Ninomiya, T., Y. Tsuruoka, Y. Miyao, and J. Tsujii. 2005. Efficacy of beam thresholding, unification filtering and hybrid parsing in probabilistic HPSG parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 103–114. Association for Computational Linguistics.

Oepen, S. and J. Carroll. 2000. Ambiguity packing in constraint-based parsing: practical results. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 162–169. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Riezler, S., T.H. King, R.M. Kaplan, R. Crouch, J.T. Maxwell III, and M. Johnson. 2001. Parsing the Wall Street Journal using a Lexical-Functional Grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 271–278.

Tomabechi, H. 1991. Quasi-destructive graph unification. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, pages 315–322. Association for Computational Linguistics.

Toutanova, K., C.D. Manning, S. Shieber, D. Flickinger, and S. Oepen. 2002. Parse disambiguation for a rich HPSG grammar. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, pages 253–263.

Van Noord, G. 1997. An efficient implementation of the head-corner parser. *Computational Linguistics*, 23(3):425–456.

van Noord, G. 2009. Learning efficient parsing. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 817–825, Athens, Greece, March. Association for Computational Linguistics.

Zhang, Y., V. Kordoni, and E. Fitzgerald. 2007a. Partial parse selection for robust deep processing. In *Proceedings of ACL 2007 Workshop on Deep Linguistic Processing*, pages 128–135, Prague, Czech.

Zhang, Y., S. Oepen, and J. Carroll. 2007b. Efficiency in Unification-Based N-Best Parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 48–59. Association for Computational Linguistics.

Local lexical adaptation in Machine Translation through triangulation: SMT helping SMT

Josep Maria Crego
LIMSI-CNRS
jmcrego@limsi.fr

Aurélien Max
LIMSI-CNRS
Univ. Paris Sud
amax@limsi.fr

François Yvon
LIMSI-CNRS
Univ. Paris Sud
yvon@limsi.fr

Abstract

We present a framework where auxiliary MT systems are used to provide lexical predictions to a main SMT system. In this work, predictions are obtained by means of pivoting via auxiliary languages, and introduced into the main SMT system in the form of a low order language model, which is estimated on a sentence-by-sentence basis. The linear combination of models implemented by the decoder is thus extended with this additional language model. Experiments are carried out over three different translation tasks using the European Parliament corpus. For each task, nine additional languages are used as auxiliary languages to obtain the triangulated predictions. Translation accuracy results show that improvements in translation quality are obtained, even for large data conditions.

1 Introduction

Important improvements are yet to come regarding the performance of Statistical Machine Translation systems. Dependence on training data and limited modelling expressiveness are the focus of many research efforts, such as using monolingual corpora for the former and syntactic models for the latter.

Another promising approach consists in exploiting complementary sources of information in order to build better translations, as done by consensus-based system combination (e.g. (Matusov et al., 2008)). This, however, requires to have several systems available for the same language pair. Considering that the same training data would be available to all systems, differences in translation modelling are expected to produce redundant and complementary hypotheses. Multisource translation (e.g. (Och and Ney, 2001; Schwartz, 2008)) is a variant, involving source texts available in several languages which can be translated by systems for different language pairs and whose outputs can be successfully combined into better translations (Schroeder et al., 2009). One theoretical expectation of multisource translation is that it can successfully reduce ambiguity of the original source text, but does so under the rare conditions of availability of existing (accurate) translations. In contrast, pivot-based system combination (e.g. (Utiyama and Isahara, 2007; Wu and Wang, 2007)) aims at compensating the lack of training data for a given language pair by producing translation hypotheses obtained by pivoting via an intermediary language for which better systems are available.

These techniques generally produce a search space that differs from that of the direct translation systems. As such, they create a new translation system out of various systems for which diagnosis becomes more difficult.

This paper instead focusses on improving a single system, which should be state-of-the-art as regards data and models. We propose a framework in which information coming from external sources is used to boost lexical choices and guide the decoder into making more informed choices.[1]

[1] We performed initial experiments where the complementary information was exploited during n-best list reranking (Max et al., 2010), but except for the multisource condition the list of hypotheses contained too little useful variation

Complementary sources can be of different nature: they can involve other automatic systems (for the same or different language pairs) and/or human knowledge. Furthermore, complementary information is injected at the lexical level, thus making targeted fine-grained lexical predictions useful. Importantly, those predictions are exploited at the sentence level[2], so as to allow for efficient use of source contextual information.

The second contribution of this paper is an instantiation of the proposed framework. Automatically pivoting via auxiliary languages is used to make complementary predictions that are exploited through language model adaptation by the decoder for a given language pair. For this apparently difficult condition, where predictions result from automatic translations involving two systems, we manage to report significant improvements, measured with respect to the target and the source text, under various configurations.

This paper is organized as follows. We first review related work in section 2.1, and describe the distinctive characteristics of our approach in Section 2.2. Section 2.3 presents our instantiation of the framework based on lexical boosting via auxiliary language triangulation. Experiments involving three language pairs of various complexity and different amounts of training data are described in Section 3. We finally conclude by discussing the prospects offered by our proposed framework in Section 4.

2 A framework for sentence-level lexical boosting

2.1 Related work

The idea of using more than one translation system to improve translation performance is not new and has been implemented in many different ways which we briefly review here.

System combination An often used strategy consists in *combining the output of several systems* for a fixed language pair, and to rescore the resulting set of hypotheses taking into account all the available translations and scores. Various proposals have been made to efficiently perform such a combination, using auxiliary data structures such as n-best lists, word lattices or consensus networks (see for instance (Kumar and Byrne, 2004; Rosti et al., 2007; Matusov et al., 2008; Hildebrand and Vogel, 2008; Tromble et al., 2008)). Theses techniques have proven extremely effective and have allowed to deliver very significant gains in several recent evaluation campaigns (Callison-Burch et al., 2008).

Multisource translation A related, yet more resourceful approach, consists in trying to combine several systems providing translations *from different sources into the same target*, provided such *multilingual sources* are available. (Och and Ney, 2001) propose to select the most promising translation amongst the hypotheses produced by several Foreign→English systems, where output selection is based on the translation scores. The intuition that if a system assigns a high figure of merits to the translation of a particular sentence, then this translation should be preferred, is implemented in the MAX combination heuristics, whose relative (lack of) success is discussed in (Schwartz, 2008). A similar idea is explored in (Nomoto, 2004), where the sole target language model score is used to rank competing outputs. (Schroeder et al., 2009) propose to combine the available sources prior to translation, under the form of a multilingual lattice, which is decoded with a multisource phrase table. (Chen et al., 2008) integrate the available auxiliary information in a different manner, and discuss how to improve the translation model of the primary system: the idea is to use the entries in the phrase table of the auxiliary system to filter out those accidental correspondences that pollute the main translation model. The most effective implementation of multisource translation to date however consists in using mono-source system combination techniques (Schroeder et al., 2009).

Translation through pivoting The use of auxiliary systems has also been proposed in another common situation, as a possible remedy to the lack of parallel data for a particular language pair, or for a particular domain. Assume, for instance, that one wishes to build a translation system for

to lead to measurable improvements.

[2] We plan to experiment next on using predictions at the document level.

the pair $A \rightarrow B$, for which the parallel data is sparse; assuming further that such parallel resources exist for pairs $A \rightarrow C$ and for $C \rightarrow B$, it is then tempting to perform the translation indirectly through *pivoting*, by first translating from A to C, then from C to B. Direct implementations of this idea are discussed e.g. in (Utiyama and Isahara, 2007). Pivoting can also intervene earlier in the process, for instance as a means to *automatically generate* the missing parallel resource, an idea that has also been considered to adapt an existing translation systems to new domains (Bertoldi and Federico, 2009). Pivoting can finally be used to fix or improve the translation model: (Cohn and Lapata, 2007) augments the phrase table for a baseline bilingual system with supplementary phrases obtained by pivoting into a third language.

Triangulation in translation Triangulation techniques are somewhat more general and only require the availabily of *one* auxiliary system (or one auxiliary parallel corpus). For instance, the authors of (Chen et al., 2008) propose to use the translation model of an auxiliary $C \rightarrow B$ system to filter-out the phrase-table of a primary $A \rightarrow B$ system.

2.2 Our framework

As in other works, we propose to make use of several MT systems (of any type) to improve translation performance, but contrarily to these works we concentrate on *improving one particular system*. Our framework is illustrated on Figure 1. The main system (henceforth, *direct* system), corresponding to configuration **1**, is a SMT system, translating from German to English in the example. Auxiliary information may originate from various sources (**2**-**6**) and enter into the decoder. A new model is dynamically built and is used to guide the exploration of the search space to the best hypothesis. Several auxiliary models can be used at once and can be weighted by standard optimization techniques using development data, so that bad sources are not used in practice, or by exploiting *a priori* information. In the implementation described in section 2.3, this information is updated by the auxiliary source at each sentence.

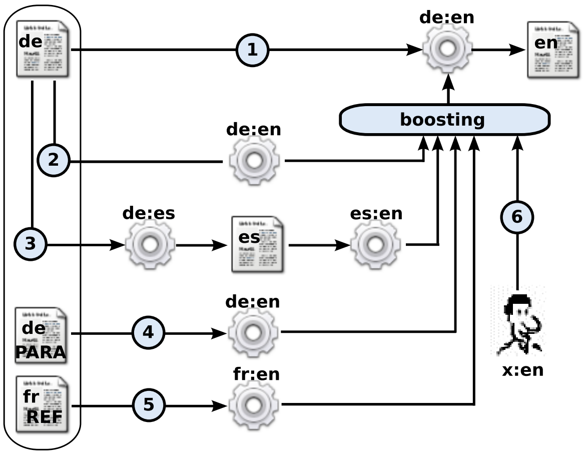


Figure 1: Lexical boosting framework with various configurations for auxiliary predictions

We now briefly describe various possible configurations to make some links to previous works explicit. Configuration **2** translates the same source text by means of another system for the same language pair, as would be done in system combination, except that here a new complete decoding is performed by the direct system. Configuration **3**, which will be detailed in section 2.3, uses translations obtained by triangulating via an auxiliary language (Spanish in the example). Using this two-step translation is common to pivot approaches, but our approach is different in that the result of the triangulation is only used as auxiliary information for the decoding of the direct system. Configurations **4** and **5** are instances of multisource translation, where a paraphrase or a translation of the source text is available. Lastly, configuration **6** illustrates the case where a human translator, with knowledge of the target language and at least of one of the available source languages, could influence the decoding by providing *desired*[3] words (e.g. only for source words or phrases that would be judged difficult to translate). This human supervision through a feedback text in real time is similar to the proposal of (Dymetman et al., 2003).

Given this framework, several questions arise,

[3] The proposal as it is limits the hypotheses produced by the system to those that are attainable given its training data. It is conceivable, however, to find ways of introducing new knowledge in this framework.

the most important underlying this work being whether the performance of SMT systems can be improved by using other SMT systems. Another point of interest is whether improvements made to *auxiliary* systems can yield improvement to the *direct* system, without the latter undergoing any modification.

2.3 Lexical boosting via triangulation

Auxiliary translations obtained by pivoting can be viewed as a source of adaptation data for the target language model of the direct system. Assuming we have computed n-best translation hypotheses of a sentence in the target language, we can then boost the likeliness of the words and phrases occurring in these hypotheses by deriving an auxiliary language model for each test sentence. This allows us to integrate this auxiliary information during the search and thus provides a tighter integration with the direct system. This idea has successfully been used in speech recognition, using for instance close captions (Placeway and Lafferty, 1996) or an imperfect translation (Paulik et al., 2005) to provide auxiliary in-domain adaptation data for the recognizer's language model. (Simard and Isabelle, 2009) proposed a similar approach in Machine Translation in which they use the target-side of an exact match in a translation memory to build language models on a per sentence basis used in their decoder.

This strategy can be implemented in a straightforward manner, by simply training a language model using the n-best list as an adaptation corpus. Being automatically generated, hypotheses in the n-best list are not entirely reliable: in particular, they may contain very unlikely target sequences at the junction of two segments. It is however straightforward to filter these out using the available phrase alignment information.

This configuration is illustrated on Figure 2: the direct system (configuration **1**) makes use of predictions from pivoting through an auxiliary language (configuration **2**), where n-best lists can be used to produce several hypotheses. In order to get a upper bound on the potential gains of this approach, we can run the artificial experiment (configuration **3**) where a reference in the target language is used as a "perfect" source of information. Furthermore, we are interested in the performance of the simple pivot system alone (configuration **4**), as it gives an indication of the quality of the data used for LM adaptation.

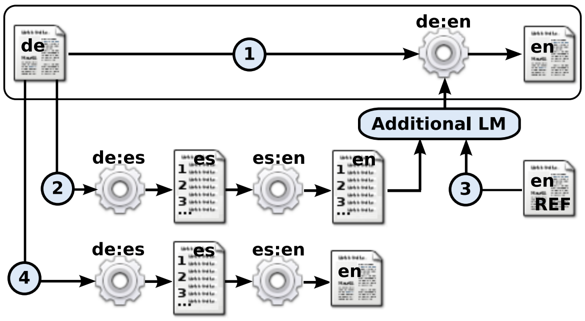


Figure 2: Architecture of a German→English system for lexical boosting via triangulation through Spanish

3 Experiments and results

3.1 Translation engine

In this study, we used our own machine translation engine, which implements the n-gram-based approach to statistical machine translation (Mariño et al., 2006). The translation model is implemented as a stochastic finite-state transducer trained using a n-gram language model of (source,target) pairs.

In addition to a bilingual n-gram model, our SMT system uses six additional models which are linearly combined following a discriminative modeling framework: two *lexicalized reordering* (Tillmann, 2004) models,a *target-language model*, two *lexicon models*, a 'weak' distance-based *distortion model*, a *word bonus model* and a *translation unit bonus model*. Coefficients in this linear combination are tuned over development data with the MERT optimization toolkit[4], slightly modified to use our decoder's n-best lists.

For this study, we used 3-gram bilingual and 3-gram target language models built using modified Kneser-Ney smoothing (Chen and Goodman, 1996); model estimation was performed with the SRI language modeling toolkit.[5] Target language

[4] `http://www.statmt.org/moses`

[5] `http://wwww.speech.sri.com/projects/srilm`

models were trained on the target side of the bitext corpora.

After preprocessing the corpora with standard tokenization tools, word-to-word alignments are performed in both directions, source-to-target and target-to-source. In our system implementation, the *GIZA++* toolkit[6] is used to compute the word alignments. Then, the *grow-diag-final-and* heuristic is used to obtain the final alignments from which translation units are extracted. Convergent studies have showed that systems built according to these principles typically achieve a performance comparable to that of the widely used MOSES phrase-based system for the language pairs under study.

3.2 Corpora

We have used the Europarl corpus[7] for our main and auxiliary languages. The eleven languages are: Danish (da), German (de), English (en), Spanish (es), Finnish (fi), French (fr), Greek (el), Italian (it), Dutch (nl), Portuguese (pt) and Swedish (sv).

We focussed on three translation tasks: one for which translation accuracy, as measured by automatic metrics, is rather high ($fr \rightarrow en$), and two for which translation accuracy is lower ($de \rightarrow en$) and ($fr \rightarrow de$). This will allow us to check whether the improvements provided by our method carry over even in situations where the baseline is strong; conversely, it will allow us to assess whether the proposed techniques are applicable when the baseline is average or poor.

In order to measure the contribution of each of the auxiliary languages we used a subset of the training corpus that is common to all language pairs, hereinafter referred to as the *intersection* data condition. We used the English side of all training language pairs to collect the same sentences in all languages, summing up to $320,304$ sentence pairs. Some statistics on the data used in this study are reported in Table 1. Finally, in order to assess the impact of the training data size over the results obtained, we also considered a much more challenging condition for the $fr \rightarrow de$ pair, where we used the entire Europarl data (V5) made available for the fifth Workshop on Statistical Machine Translation[8] for training, and test our system on out-of-domain news data. The training corpus in this condition contains 43.6M French words and 37.2M German words.

Development and test data for the first condition (*intersection*) were obtained by leaving out respectively 500 and 1000 sentences from the common subset (same sentences for all languages), while the first 500 sentences of *newstest2008* and the entire *newstest2009* official test sets were used for the *full* data condition.

	Train		*Dev*			*Test*		
	Words	*Voc.*	*Words*	*Voc.*	*OOV*	*Words*	*Voc.*	*OOV*
da	8.5M	133.5k	13.4k	3.2k	104	25.9k	5.1k	226
de	8.5M	145.3k	13.5k	3.5k	120	26.0k	5.5k	245
en	8.9M	53.7k	14.0k	2.8k	39	27.2k	4.0k	63
es	9.3M	85.3k	14.6k	3.3k	56	28.6k	5.0k	88
fi	6.4M	274.9k	10.1k	4.3k	244	19.6k	7.1k	407
fr	10.3M	67.8k	16.1k	3.2k	47	31.5k	4.8k	87
el	8.9M	128.3k	14.1k	3.9k	72	27.2k	6.2k	159
it	9.0M	78.9k	14.3k	3.4k	61	28.1k	5.1k	99
nl	8.9M	105.0k	14.2k	3.1k	76	27.5k	4.8k	162
pt	9.2M	87.3k	14.5k	3.4k	49	28.3k	5.2k	118
sv	8.0M	140.8k	12.7k	3.3k	116	24.5k	5.2k	226

Table 1: *Statistics for the training, development and test sets of the intersection data condition*

3.3 Results

In this section, we report on the experiments carried out to assess the benefits of introducing an auxiliary language model to the linear combination of models implemented in our SMT system.

Table 2 reports translation accuracy (BLEU) results for the main translation tasks considered in this work ($fr \rightarrow de$), ($fr \rightarrow en$) and ($de \rightarrow en$), as well as for multiple intermediate tasks needed for pivoting via auxiliary systems.

For each triplet of languages (*src*, *aux*, *trg*), columns 4^{th} to 6^{th} show BLEU scores for systems performing ($src \rightarrow aux$), ($aux \rightarrow trg$) and *pivot* translations using *aux* as the bridge language.

The last two columns display BLEU scores for the main translation tasks ($fr \rightarrow de$), ($fr \rightarrow en$) and ($de \rightarrow en$). Column *src-trg* refers to the baseline (direct) systems, for which no additional lan-

[6] http://www.fjoch.com/GIZA++.html

[7] http://www.statmt.org/europarl

[8] http://www.statmt.org/wmt10

src aux trg	*src-aux*	*aux-trg*	*pivot*	*src-trg*	*+auxLM*
Intersection data condition					
fr - de	-	-	-	18.02	
da	22.78	20.02	16.27		+0.44
el	24.54	18.51	15.86		+0.76
en	29.53	17.31	15.69		+0.50
es	**34.94**	18.31	**16.76**		**+0.96**
fi	10.71	14.15	11.39		+0.65
it	31.60	16.86	16.54		-0.05
nl	22.71	**21.44**	**16.76**		+0.55
pt	33.61	17.47	16.34		-0.12
sv	20.73	19.59	13.73		-0.14
average					*+0.39*
- - ref	-	-	-	-	+6.46
fr - en	-	-	-	29.53	
da	22.78	29.54	25.48		+0.02
de	18.02	24.66	23.50		+0.05
el	24.54	29.37	25.31		+0.07
es	**34.94**	**31.05**	**27.76**		**+0.61**
fi	10.71	20.56	19.15		+0.44
it	31.60	25.75	25.79		+0.32
nl	22.71	24.49	25.15		+0.01
pt	33.61	29.44	27.27		+0.01
sv	20.73	30.98	23.74		+0.50
average					*+0.22*
- - ref	-	-	-	-	+11.30
de - en	-	-	-	24.66	
da	24.59	29.54	**22.73**		**+0.96**
el	19.72	29.37	20.88		**+1.02**
es	**25.48**	**31.05**	21.23		+0.77
fi	12.42	20.56	18.02		**+0.94**
fr	25.93	29.53	21.55		+0.19
it	18.82	25.75	18.05		+0.19
nl	24.97	24.49	22.62		+0.64
pt	23.15	29.44	21.93		+0.87
sv	19.80	**30.98**	21.35		+0.69
average					*+0.69*
- - ref	-	-	-	-	+9.53
Full data condition					
fr - de	-	-	-	19.94	
es	38.76	20.18	19.36		+0.61

Table 2: *Translation accuracy (BLEU) results.*

guage model is used; column *+auxLM* refers to the same system augmented with the additional language model. Additional language models are built from hypotheses obtained by means of *pivot* translations, using *aux* as auxiliary language. The last score is shown in the form of the difference (improvement) with respect to the score of the baseline system.

This table additionally displays the BLEU results obtained when building the additional language models directly from the English reference translations (see last row of each translation task). These numbers provide an upper-bound of the expected improvements. Note finally that numbers in boldface correspond to the best numbers in their column for a given language pair.

As detailed above, the additional language models are built using *trg* hypotheses obtained by pivoting via an auxiliary language: $(src \rightarrow aux)$ + $(aux \rightarrow trg)$. Hence, column *pivot* shows the quality (measured in terms of BLEU) of the hypotheses used to estimate the additional model. Note that we did not limit the language model to be estimated from the 1-best *pivot* hypotheses. Instead, we uses n-best translation hypotheses of the $(src \rightarrow aux)$ system and m-best hypotheses of the $(aux \rightarrow trg)$ system. Hence, $n \times m$ target hypotheses were used as training data to estimate the additional models. Column *+auxLM* shows BLEU scores over the test set after performing four system optimizations on the development set to select the best combination of values used for n and m among: $(1, 1)$, $(10, 1)$, $(10, 1)$ and $(10, 10)$. All hypotheses used to estimate a language model are considered equally likely. Language models are learnt using Witten-Bell discounting. Approximately ± 1.0 point must be added to BLEU scores shown in the last 2 columns for 95% confidence levels.

As expected, *pivot* translations yield lower quality scores than the corresponding direct translations hypotheses. However, *pivot* hypotheses may contain better lexical predictions, that the additional model helps transfer into the baseline system, yielding translations with a higher quality, as shown in many cases the *+auxLM* systems results. The case of using Finnish as an auxiliary language is particularly remarkable. Even though *pivot* hypotheses obtained through Finnish have the lowest scores[9], they help improve the baseline performance as additional language models.

As expected, the translation results of the pair

[9] Given the agglutinative nature of morphological processes in Finnish, reflected in a much lower number of words per sentence, and a higher number of types (see Table 1), BLEU scores for this language do not compare directly with the ones obtained for other languages.

with a highest baseline ($fr \rightarrow en$) were on average less improved than those of the pairs with lower baselines.

As can also be seen, the contribution of each auxiliary language varies for each of the three translation tasks. For instance, Danish (da) provides a clear improvement to ($de \rightarrow en$) translations, while no gain is observed for ($fr \rightarrow en$). No clear patterns seems to emerge, though, and the correlation between the quality of the pivot translation and the boost provided by using these pivot hypotheses remains to be better analyzed.

In order to assess whether the improvements obtained carry over larger data conditions, we trained our ($fr \rightarrow de$), ($fr \rightarrow es$) and ($es \rightarrow de$) systems over the entire EPPS data. Results are reported in the bottom part of Table 2. As can be seen, the ($fr \rightarrow de$) system is still improved by using the additional language model. However, the absolute value of the gain under the *full* condition ($+0.61$) is lower than that of the *intersection* data condition ($+0.96$).

3.4 Contrastive evaluation of lexical translation

In some cases, automatic metrics such as BLEU cannot show significant differences that can be revealed by fine-grained focussed human evaluation (e.g. (Vilar et al., 2006)). Furthermore, computing some similarity between a system's hypotheses and *gold standard* references puts a strong focus on the target side of translation, and does not allow evaluating translation performance from the source words that were actually translated. We therefore use the evaluation methodology described in (Max et al., 2010) for a complementary measure of translation performance that focuses on the contrastive ability of two systems to adequately translate source words.

Source words from the test corpus were first aligned with target words in the reference, by automatically aligning the union of the training and test corpus using GIZA++.[10] The test corpus was analyzed by the TREETAGGER[11] so as to identify content words, which have a more direct impact on translation adequacy. When source words are aligned to several target words, each target word should be individually searched for in the candidate translation, and words from the reference can only be matched once.

		Source words' part-of-speech						
aux		**ADJ**	**ADV**	**NOM**	**PRO**	**VER**	all	**+Bleu**
el	-	27	21	114	25	99	286	+0.07
	+	62	29	136	27	114	368	
es	-	33	25	106	26	110	300	+0.61
	+	64	38	136	22	117	377	
fi	-	44	40	106	20	92	302	+0.44
	+	49	31	120	23	106	329	
it	-	55	39	128	35	119	376	+0.32
	+	55	39	145	36	121	396	
sv	-	40	30	138	29	109	346	+0.50
	+	69	46	144	23	134	416	

Table 3: Contrastive lexical evaluation results per part-of-speech between the baseline French→English system and our systems using various auxiliary languages. '-' (resp. '+') values indicate numbers of words that only the baseline system (resp. our system) correctly translated with respect to the reference translation.

Table 3 shows contrastive results per part-of-speech between the baseline fr→en system and systems using various auxiliary languages. Values in the '-' row indicate the number of words that only the baseline system translated as in the reference translation, and values in the '+' row the number of words that only our corresponding system translated as in the reference. The most striking result is the contribution of Greek, which, while giving no gain in terms of BLEU, improved the translation of 82 content words. This could be explained, in addition to the lower Bleu3 and Bleu4 precision, by the fact that the quality of the translation of grammatical words may have decreased. On the contrary, Italian brings little improvement for content words save for nouns. The mostly negative results on the translation of pronouns were expected, because this depends on their antecedent in English and is not the object of specific modelling from the systems. The translation of nouns and adjectives benefits the most from auxiliary translations.

[10] The obtained alignments are thus strongly influenced by alignments from the training corpus. It could be noted that alignments could be manually corrected.

[11] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger`

Figure 3 illustrates this evaluation by means of two examples. It should be noted that a recurrent type of improvement was that of avoiding missing words, which is here a direct result of their being boosted in the auxiliary hypotheses.

4 Conclusions and future work

We have presented a framework where auxiliary MT systems are used to provide useful information to a main SMT system. Our experiments on auxiliary language triangulation have demonstrated its validity on a difficult configuration and have shown that improvements in translation quality could be obtained even under large training data conditions.

The fact that low quality sources such as pivot translation can provide useful complementary information calls for a better understanding of the phenomena at play. It is very likely that, looking at our results on the contribution of auxiliary languages, improving the quality of an auxiliary source can also be achieved by identifying what a source is good for. For example, in the studied language configurations predictions of translations for pronouns in the source text by auxiliary triangulation does not give access to useful information. On the contrary, triangulation with Greek when translating from French to English seems to give useful information regarding the translation of adjectives, a result which was quite unexpected.

Also, it would be interesting to use richer predictions than short *n*-grams, such as syntactic dependencies, but this would require significant changes on the decoders used. Using dynamic models at the discourse level rather than only at the sentence level would also be a useful improvement. Besides the improvements just mentioned, our future work includes working on several configurations of the framework described in section 2.2, in particular investigating the new type of system combination.

Acknowledgements

This work has been partially funded by OSEO under the Quaero program.

References

Bertoldi, Nicola and Marcello Federico. 2009. Domain adaptation for statistical machine translation with monolingual resources. In *Proceedings of WMT*, Athens, Greece.

Callison-Burch, Chris, Cameron Shaw Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proceedings of WMT*, Columbus, USA.

Chen, Stanley F. and Joshua T. Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of ACL*, Santa Cruz, USA.

Chen, Yu, Andreas Eisele, and Martin Kay. 2008. Improving statistical machine translation efficiency by triangulation. In *Proceedings of LREC*, Marrakech, Morocco.

Cohn, Trevor and Mirella Lapata. 2007. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *Proceedings of ACL*, Prague, Czech Republic.

Dymetman, Marc, Aurélien Max, and Kenji Yamada. 2003. Towards interactive text understanding. In *Proceedings of ACL, short paper session*, Sapporo, Japan.

Hildebrand, Almut Silja and Stephan Vogel. 2008. Combination of machine translation systems via hypothesis selection from combined n-best lists. In *Proceedings of AMTA*, Honolulu, USA.

Kumar, Shankar and William Byrne. 2004. Minimum bayes-risk decoding for statistical machine translation. In *Proceedings of NAACL-HLT*, Boston, USA.

Mariño, José, Rafael E. Banchs, Josep Maria Crego, Adria de Gispert, Patrick Lambert, J.A.R. Fonollosa, and Martha Costa-jussà. 2006. N-gram based machine translation. *Computational Linguistics*, 32(4):527–549.

Matusov, Evgeny, Gregor Leusch, Rafael E. Banchs, Nicola Bertoldi, Daniel Dechelotte, Marcello Federico, Muntsin Kolss, Young-Suk Lee, Jose Mariño, Matthias Paulik, Salim Roukos, Holger Schwenk, and Hermann Ney. 2008. System combination for machine translation of spoken and written language. *IEEE Transactions on Audio, Speech and Language Processing*, 16(7):1222–1237, September.

Max, Aurélien, Josep M. Crego, and François Yvon. 2010. Contrastive Lexical Evaluation of Machine Translation. In *Proceedings of LREC*, Valletta, Malta.

ref #357	this concession to the unions ignores the reality that all airlines have different safety procedures which even differ between aircrafts within each airline .
bas	this **concession unions ignores** the *fact* that all **airlines have different safety procedures** which are **even within each** of the *companies* in accordance with the types of equipment .
w.r.t. src	cette **concession** aux **syndicats ignore** la *réalité* selon laquelle toutes les **compagnies aériennes ont** des **procédures de sécurité différentes** qui diffèrent **même** au **sein** de chacune des *compagnies* en fonction des types d ' *appareils* .
+aux	this **concession** to the trade **unions ignores** the **reality** according to which all the **airlines have different safety procedures** which **differ even within each** of the *companies* in accordance with the types of equipment .
w.r.t. src	cette **concession** aux **syndicats ignore** la **réalité** selon laquelle toutes les **compagnies aériennes ont** des **procédures de sécurité différentes** qui **diffèrent même** au **sein** de chacune des *compagnies* en fonction des types d ' *appareils* .

Figure 3: Example of automatic translations from French to English for the baseline system and when using Spanish as the auxiliary language. Bold marking indicates source/target words which were correctly translated according to the reference translation.

Nomoto, Tadashi. 2004. Multi-engine machine translation with voted language model. In *Proceedings of ACL*, Barcelona, Catalunya, Spain.

Och, Franz Josef and Hermann Ney. 2001. Statistical multi-source translation. In *Proceedings of MT Summit*, Santiago de Compostela, Spain.

Paulik, Matthias, Christian Fügen, Thomas Schaaf, Tanja Schultz, Sebastian Stüker, and Alex Waibel. 2005. Document driven machine translation enhanced automatic speech recognition. In *Proceedings of InterSpeech*, Lisbon, Portugal.

Placeway, Paul and John Lafferty. 1996. Cheating with imperfect transcripts. In *Proceedings of ICSLP*, Philadelphia, USA.

Rosti, Antti-Veikko, Necip Fazil Ayan, Bin Xiang, Spyros Matsoukas, Richard Schwatz, and Bonnie J. Dorr. 2007. Combining outputs from multiple machine translation systems. In *Proceedings of NAACL-HTL*, Rochester, USA.

Schroeder, Josh, Trevor Cohn, and Philipp Koehn. 2009. Word lattices for multi-source translation. In *Proceedings of EACL*, Athens, Greece.

Schwartz, Lane. 2008. Multi-source translation methods. In *Proceedings of AMTA*, Honolulu, USA.

Simard, Michel and Pierre Isabelle. 2009. Phrase-based machine translation in a computer-assisted translation environment. In *Proceedings of Machine Translation Summit XII*, Ottawa, Canada.

Tillmann, Christoph. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of NAACL-HLT*, Boston, USA.

Tromble, Roy, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *Proceedings of EMNLP*, Honolulu, USA.

Utiyama, Masao and Hitoshi Isahara. 2007. A comparison of pivot methods for phrase-based statistical machine translation. In *Proceedings of NAACL-HLT*, Rochester, USA.

Vilar, David, Jia Xu, Luis Fernando d'Haro, and Hermann Ney. 2006. Error Analysis of Statistical Machine Translation Output. In *Proceedings of LREC*, Genoa, Italy.

Wu, Hua and Haifeng Wang. 2007. Pivot language approach for phrase-based statistical machine translation. In *Proceedings of ACL*, Prague, Czech Republic.

Automated Translation of Semantic Relationships

Dmitry Davidov
ICNC
Hebrew University of Jerusalem

Ari Rappoport
Institute of Computer Science
Hebrew University of Jerusalem
arir@cs.huji.ac.il

Abstract

We present a method for translating semantic relationships between languages where relationships are defined as pattern clusters. Given a pattern set which represents a semantic relationship, we use the web to extract sample term pairs of this relationship. We automatically translate the obtained term pairs using multilingual dictionaries and disambiguate the translated pairs using web counts. Finally we discover the set of most relevant target language patterns for the given relationship. The obtained pattern set can be utilized for extraction of new relationship examples for the target language.

We evaluate our method on 11 diverse target languages. To assess the quality of the discovered relationships, we use an automatically generated cross-lingual SAT analogy test, WordNet relationships, and concept-specific relationships, achieving high precision. The proposed framework allows fully automated cross-lingual relationship mining and construction of multilingual pattern dictionaries without relying on parallel corpora.

1 Introduction

Acquiring and understanding semantic relationships is crucial for many NLP applications. In many cases, we would like to know if a given term pair participates in a specified semantic relationship or if two different term pairs encode the same (possibly unspecified) type of relationship. Beyond the well-known major relationship types such as hyponymy (is-a) and meronymy (part-of), there is a huge number of other relationships between objects and concepts. Examples include general relations such as larger-than, contained-in, liked-by and domain specific ones such as country-language, product-manufacturer, product-seller, drug-disease etc.

The vast majority of NLP research is done in a few languages for which extensive corpora (including the web) are available. As a result, most relationship retrieval studies and lexical database compilation efforts target only a few languages. However, due to the substantial growth of the multilingual web[1] and a growing demand for NLP application coverage for less common languages, there is a need for relationship data in many less studied languages.

In this paper we address the task of translating relationships between languages, which has two obvious benefits. First, it can directly help applications such as machine translation, cross-lingual information retrieval, cross-lingual web mining and the construction and enrichment of semantic databases. Second, it can assist applications in a single language, especially when compensating for a relative scarcity of resources in that language. We focus on relations between two entities, which are the most common type.

When discussing the translation of relationships, it is important to define how these are represented and in what way the task differs from MT. While there is wide agreement on the definition and representation of major relationship types such as hypernymy and (to a lesser extent) meronymy, there is no single accepted method (or

[1] http://www.internetworldstats.com/stats7.htm

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 241–249,
Beijing, August 2010

resources) for other less common relationships. Among the methods that have been proposed for specifying lexical relationships are natural language description and rules (Girju et al., 2007), distributional means (Turney, 2005), sample term pairs (Pasca et al, 2006), relationship instances (Banko et al., 2007) and pattern clusters (Davidov and Rappoport, 2008a).

In this paper we utilize the last definition. Following (Davidov and Rappoport, 2008a) each semantic relationship can be defined and represented by a set of lexical patterns such that the represented relation holds between entities filling the patterns' slots. We focus on pattern clusters relationship definition due to several reasons. First, as opposed to natural language descriptions, pattern clusters are formal. Second, as opposed to the other methods above, pattern clusters provide a 'generative' model for the represented relationship – it is possible to obtain from them relationship instances and term pairs, as we indeed utilize in this paper. Third, pattern clusters can be mined in a fully unsupervised manner, or in a focused manner when the relationship desired is known. Finally, pattern methods have proven to be highly efficient and effective for lexical acquisition tasks (Pantel et al, 2004; Davidov and Rappoport, 2006).

The proposed framework comprises the following stages. First, given a set of patterns defining a relationship in a source language, we obtain from the web a set of corresponding term pairs. Next, for each of the terms in the obtained term pairs, we retrieve sets of their translations to the target language using available multilingual dictionaries. Now that we have a set of translations for each term in each pair, we retrieve search engine snippets with the translated term pairs. We then select appropriate word senses using web counts, and extract a set of patterns which connect these disambiguated terms. As a result we get a set of relation-specific target language patterns, effectively obtaining the desired relationship definition. We can optionally use the retrieved pattern sets to obtain term pairs of target language relationships from the web.

We performed a thorough evaluation for various relationships involving 11 languages. We tested our framework on major relationships like meronymy, specific relationships like country-capital and unspecified unsupervisedly discovered English relationships. The obtained relationships were manually verified by human judges using cross-lingual SAT analogy questions, and a few specific factual relationships were evaluated using a gold standard.

Our main contribution is a novel framework for automated relationship translation across languages, where relationships are defined as pattern clusters or as term pairs. This framework allows fully automated cross-lingual relationship mining and construction of multilingual pattern dictionaries without relying on parallel corpora.

In Section 2 we discuss related work. Section 3 details the algorithm. Section 4 describes the evaluation, and Section 5 concludes.

2 Related work

Recently, with the development of practical applications which utilize WN-like databases in dozens of languages, great effort has been made to manually construct and interconnect such databases for different languages (Pease et al, 2008; Charoenporn et al., 2007). Some studies (e.g., (Amasyali, 2005)) use semi-automated methods based on language-specific heuristics and dictionaries.

At the same time, much work has been done on automated lexical acquisition for a single language, and in particular, on the web-based acquisition of various types of semantic relationships. There is a substantial amount of related studies which deal with the discovery of various relationship types represented in useful resources such as WordNet, including hypernymy (Pantel et al, 2004; Snow et al., 2006), synonymy (Davidov and Rappoport, 2006; Widdows and Dorow, 2002) and meronymy (Berland and Charniak, 1999; Girju et al, 2006). Since named entities are very important in NLP, many studies define and discover relations between named entities (Hassan et al., 2006). Work was also done on relations between verbs (Chklovski and Pantel, 2004). There is growing research on relations between nominals (Girju et al., 2007).

While the majority of studies focus on extracting pre-specified semantic relationships, several recent studies were done on the automated discovery of unspecified relationship types. Thus Turney (2006) provided a pattern distance measure that allows a fully unsupervised measurement of relational similarity between two pairs of words on the same language. Banko et al. (2007) and Rosenfeld and Feldman (2007) find relationship instances where the relationships are not specified in advance. (Davidov and Rappoport, 2008a) introduced the idea that salient semantic relationships can be defined as pattern clusters, confirming it with SAT analogy test. As explained above, we use this definition in the present study. We also use pattern clusters given by (Davidov and Rappoport, 2008a) as input in our evaluation.

Most of the relationship acquisition studies were done in a single language. Those that experiment in several languages usually treat each language separately, while we extract a relationship definition for one language using the provided definition for the other language.

Our study is related to cross-language information retrieval (CLIR) frameworks. Both deal with multilingual information extracted from the Web. However, the majority of CLIR studies pursue different targets. Thus, one of the main CLIR goals is the retrieval of *documents* based on explicit queries, when the document language is not the query language (Volk and Buitelaar, 2002). These frameworks usually develop language-specific tools and algorithms including parsers, taggers and morphology analyzers in order to integrate multilingual *queries* and *documents* (Jagarlamudi and Kumaran, 2007). Our goal is to develop and evaluate a *language-independent* algorithm for the *cross-lingual translation of relationship-defining structures*. While our targets are different from those of CLIR, CLIR systems can greatly benefit from our framework, since we can translate the relationships in CLIR queries and subsequently check if the same relationships are present in the retrieved documents.

Another field indirectly related to our research is Machine translation (MT). Many MT tasks require automated creation or improvement of dictionaries (Koehn and Knight, 2001). However, MT mainly deals with translation and disambiguation of words at the sentence or document level, while we translate relationship structures as a set of patterns, defined independently of contexts. We also perform pattern-set to pattern-set translation rather than the pattern-to-pattern or pair-to-pair translation commonly explored in MT studies. This makes it difficult to perform meaningful comparison to existing MT frameworks. However, the MT studies benefit from the proposed framework by enhancement and verification of translated relationship instances.

In (Davidov and Rappoport, 2009), we proposed a framework for automated cross-lingual concept mining. We incorporate several principles from this study including concept extension and disambiguation of query language (See Section 3.3). However our goals here are different since we target cross-lingual acquisition of relationship structures rather then concept term lists.

3 Relationship Translation Framework

Our framework has the following stages: (1) given a set of patterns in a source language defining some lexical relationship, we use the web to obtain source language term pairs participating in this relationship; (2) we automatically translate the obtained terms in each pair to the target language using available multilingual dictionaries; (3) we retrieve web snippets where these translations co-appear, disambiguating translations with web counts and extracting the corresponding patterns. As an optional final stage, the translated pattern cluster can be used to extract and extend a set of target language term pairs. Now we describe each of these stages in detail.

3.1 Acquisition of representative term pairs

We are provided with a pattern cluster, a set of patterns representing a specific lexical relationship in some language. The goal of the first stage is to discover the most representative term pairs for this cluster and language from the web. If the relationship is already specified by a representative set of term pairs, we skip this stage and continue to the next stage. Note that the method described below can also be used at the final stage to obtain representative *target* language term pairs once we obtain a *target* language pattern cluster.

The input lexical patterns are surface patterns which include several fixed words or punctuation symbols and two slots for content words, e.g. *"the [X] of the [Y],"*. Given a cluster of patterns defining a semantic relationship, we would like to obtain from the web the most representative and frequent examples of the represented relationship. In order to do that we construct search engine queries[2] from the given patterns using wildcard symbols to represent pattern slots. For example, given a pattern *"the [X] of the [Y],"* we construct queries such as *"the * of the"; "the * * of the"*[3]. We collect all the retrieved search engine snippets and extract the appropriate term pairs found in these snippets.

Now we would like to select the most useful of the extracted pairs. Since the obtained pairs are only useful if we can translate them into the target language, we dismiss all pairs in which one or both terms have no translations to the target language in our dictionaries (see Section 3.2). Since each particular pattern can be ambiguous, we also dismiss pairs which were found for only a single pattern in the given cluster.

For the remaining term pairs we would like to estimate their specificity for the given pattern cluster. For each pattern, we retrieve and use two web hit counts: $F_{terms}(p, T1, T2)$, a hit count for co-appearance of the pair in a way similar to that in the pattern, and $F_{all}(p, T1, T2)$, the hit count of the full pattern instance.

For example, if for the pattern *p="the * of the"* we obtain a term pair (CEO, company), then $F_{all}(p)$=*Hits("the CEO of the company")* and $F_{terms}(CEO, company)$= *Hits("CEO * * company")*. Given a pattern cluster C with patterns $\{p_1 \dots p_n\} \in C$, we estimate the specificity of a term pair $(T1, T2)$ using the following simple probabilistic metric, giving to all patterns in the cluster an equal weight:

$$Spec(T1, T2) = \frac{1}{n} \sum_{p_i \in C} \frac{F_{all}(p_i, T1, T2)}{F_{terms}(p_i, T1, T2)}$$

We select the top 15 pairs with the highest specificity and use them in the next stage.

3.2 Translation of the term pairs

After the previous stage we have a good representative set of term pairs for the desired source language relationship. Now we would like to translate the words in these pairs to the target language. In order to do that we use an extensive set of 1067 multilingual dictionaries developed for StarDict[4], including Wikipedia cross-language links and Wiktionary. For each term we obtain a set of its translations to the target language. If we get more than five different translations, we select the five having the highest number of dictionaries where this translation appears.

As discussed in Section 3.1, we dismissed terms for which no translation was found in any of the available dictionaries, so each term in each of the obtained pairs has at least a single translation to the target language. However, in many cases the available translations represent the wrong word sense, since both the source terms and their translations can be ambiguous. Thus at this stage many of the obtained term translations are irrelevant for the given relationship and require disambiguation.

3.3 Web mining for translation contexts

For this stage, we need to restrict web mining to specific target languages. This restriction is straightforward if the alphabet or term translations are language-specific or if the search API supports restriction to this language. In case where there is no such natural restrictions, we attempt to detect and add to our queries a few language-specific frequent words. Following (Davidov and Rappoport, 2009), we use our dictionaries to find 1–3 of the 15 most frequent words in a desired language[5] that are unique to that language and 'and' them with the queries to ensure proper language selection. This allows applying our algorithm to more than 60 diverse languages. The only data required for each language is at least a partial coverage of the obtained term pairs by some available dictionary.

Given a term pair $(T1, T2)$ we obtain a set of translations $(T1'_{i \in 1 \dots n}, T2'_{j \in 1 \dots m})$. For each combination $T1'_i, T2'_j$ of the obtained term translations, we construct and execute the following

[2] We use Yahoo! Boss.

[3] Since the search engine API doesn't allow punctuation, we omit the punctuation in queries, but require a proper punctuation when processing the obtained snippet data.

[4] http://stardict.sourceforge.net/

[5] We estimated the word frequencies from text available in the corresponding multilingual dictionaries.

four queries: {"$T1'_i$ $*$ $T2'_j$", "$T2'_j$ $*$ $T1'_i$", "$T1'_i$ $*$ $*$ $T2'_j$", "$T2'_j$ $*$ $*$ $T1'_i$"}[6]. Since $Yahoo!Boss$ allows retrieval of up to the 1000 first results, we can collect up to four thousand snippets for each combination. However, the majority of these combinations return no snippets at all, effectively generating an average of a dozen snippets per query.

3.4 Pattern extraction

Now for each pair of term translations we would like to extract from the snippets all surface patterns which connect the terms in this pair. We use the basic two-slot meta-pattern type:

$$[Prefix]\ X\ [Infix]\ Y\ [Postfix]$$

X and Y should be the translated terms, Infix may contain punctuation, spaces, and up to four words (or up to eight symbols in languages without space-separated words like Chinese). Prefix and Postfix are limited to contain one or zero punctuation characters and/or up to two words. We do not allow empty Infix, Prefix of Postfix. If there are several possible combinations of Prefix and Postfix we generate a pattern set for all possible combinations (e.g., if we retrieve a snippet …", *consider using [plexiglass] for [kitchen].*"…, we create patterns "using X for Y.", "consider using X for Y." and ", consider using X for Y.").

Now we would like to find the patterns representing the relationship in the target language. We do this in two stages. First we would like to detect the most common patterns for the given relationship. Let S_k be the union set of all patterns obtained for all combinations of the extracted translations for a specific source language term pair $k \in 1 \ldots K$. Let $Salience(p) = \frac{1}{K}|\{k|p \in S_k\}|$ be the portion of source language term pairs which lead to detection of the target language pattern p. We compute salience for each pattern, and select a subset of *salient patterns*, defined to be those whose Salience exceeds a predefined threshold (we used $1/3$). If one salient pattern is a substring of another salient pattern, we only select the longer one.

In our salience estimation we mix data from all combinations of translations including incorrect senses and wrong translations of ambiguous terms. Now we would like to select a single correct target language pair for each source language pair in order to find more refined relationship representing patterns. For each source language term pair, we select the target language translated pair which captured the highest number of salient patterns. In case there are several pairs with the same number of salient patterns, we select a pair with the greatest web hit count. We drop term pairs with zero salient patterns.

Finally we would like to enhance the obtained set of salient patterns with more precise and representative relationship-specific patterns. Since we disambiguated the translated pairs, target language patterns captured by the remaining term pairs should be more trusted. We compare the target language pattern sets obtained for different remaining term pairs, and collect all patterns that were captured by at least three different term pairs. As before, if one pattern is a substring of another we retain only the longer one. As a result we get a comprehensive target language pattern cluster for the desired relationship.

3.5 Retrieval of target language term pairs

As an optional final stage, we can utilize the retrieved target language pattern clusters in order to discover target language term pairs for the desired relationship. We do this by utilizing the strategy described in Section 3.1 on the obtained target language pattern clusters. We do not dismiss obtained terms having no available dictionary translations, and we do not limit our search to the 15 terms with highest specificity. Instead we either select N term pairs with top specificity (where N is provided by user as in our evaluation), or we select all term pairs with specificity above some threshold.

4 Evaluation

In order to test the quality of the translated pattern clusters and the corresponding translated term pairs, we need to check both flexibility and correctness. Flexibility measures how well the retrieval works well *across languages* and for *many*

[6] These are Yahoo! queries where enclosing words in "" means searching for an exact phrase and "*" means a wildcard for exactly one arbitrary word.

types of semantic relationships. To do that, we tested our framework on both generic and specific relationships for 11 languages. Correctness verifies that the retrieved set of target language patterns and the corresponding term pairs represent *the same semantic relationship* as the given set of source language term pairs or patterns. To do that, we used both manual cross-lingual analogy-based correctness evaluation and evaluation based of factual data.

4.1 Languages and relationships

One of the main goals in this research was to provide a fully automated and flexible framework, which requires minimal modifications when applied to different languages and relationships.

We examined an extensive set of target languages using English as a source language. Table 1 shows 11 languages used in our experiments. We included west European languages, Slavic languages like Russian, Semitic languages like Hebrew, and Asian languages such as Chinese. We developed a set of tools for automatic off-line access to an extensive set of 1067 multilingual dictionaries created for the StarDict platform. These dictionaries include recent dumps of Wikipedia cross-language links and Wiktionary data.

In our experiments we used three sets of relationships: **(1) Generic:** 15 unsupervisedly discovered English pattern clusters representing various generic relationships. **(2) H-M-C:** The three most studied relationships: hypernymy, meronymy and co-hyponymy.**(3) Specific:** Three factual relationships: country-capital, country-language and dog breed-origin. Below we describe the evaluation of each of these sets in detail. Note that our framework allows two ways of specifying a source language relationship – a pattern cluster and a set of term pairs.

4.2 Evaluation of generic pattern clusters

In our **Generic** evaluation setting, we utilized as input a random sample of 15 automatically discovered relationship definitions. We started from a set of 508 English pattern clusters, unsupervisedly discovered using the method of (Davidov and Rappoport, 2008a). Each of these clusters is assumed to represent a distinct semantic relationship. We randomly selected 15 pattern clusters from this set and executed our framework on these clusters to obtain the corresponding target language pattern clusters for each of the 11 tested languages. An example of a partial set of patterns in a cluster is: *‘this [X] was kept in [Y],”;‘the X that he kept in [Y],”;‘the [X] in the [Y] and”;‘the [Y] containing the [X]”.*. . .

We then used the term pair selection algorithm described in Section 3.1 to select the most specific term pair for each of the 15 source language clusters and 10 pairs for each of the corresponding translated target language clusters. Thus for each of the 15 pattern clusters and for each of the 11 languages we produced a single source language term pair and up to 10 corresponding target language term pairs.

In order to check the correctness of translation of an unspecified semantic relationship we need to compare source and target language relationships. Comparison of relationships is a challenging task, since there are no relationship resources for most relationship types even in a single language, and certainly so for their translations across languages. Thus various studies define and split generic relationships differently even when describing relatively restricted relationship domains (e.g., relationships holding between parts of noun phrases (Nastase and Szpakowicz, 2003; Moldovan et al., 2004)). In order to compare generic relationships we used a manual cross-lingual SAT-like analogy human judgment evaluation[7]. This allowed us to assess the quality of the translated pattern clusters, in a similar way as (Davidov and Rappoport, 2008a) did for testing clusters in a single language.

For each of the 15 clusters we constructed a cross-lingual analogy question in the following manner. The header of the question was a term pair obtained for the source language pattern cluster. The six multiple choice items included: (1) one of the 10 discovered translated term pairs of the same cluster (the ‘correct’ answer)[8]; (2) three

[7] Using Amazon’s Mechanical Turk.

[8] We avoid selection of the target language pairs which were obtained through direct translation of the source language pair given at the header of the question. This is crucial so that subjects will not judge correctness of translation but correctness of the relationship.

of the translated pairs of the other clusters among the 15; (3) a pair constructed by randomly selecting terms from different translated clusters; (4) the 6th option states that either the given options include broken words or incorrect language, or none of the presented pairs even remotely exemplifies the relationship in question. An example question for English-Italian:

The English pair: *(kennel, dog)*; (1) “correct” pair: *(acquario, pesce)*; (2)-(4) “wrong” pairs: *(topo, orecchio), (mela, rossa), (occhio, grande)*; (5) “random”: *(scodella, scatola)*; (6) Pairs comprise non-Italian/broken words or no pair exemplifies the relationship

In order to check the English proficiency of the subjects we added 5 “easy” monolingual English SAT analogy questions. We also added a single hand-crafted cross-lingual question of an obvious analogy case, making a total of 16 cross-lingual questions. Subjects who failed more than one of the easy English SAT questions or failed the obvious cross-lingual question were rejected from the evaluation. Finally we have three subjects for each of the tested languages. We also asked the subjects to assign a confidence score from 0 (worst) to 10 (best) to express how well the selected term pair represents the source language relationship in question.

Language	P	% 6th	$Score_c$	$Score_w$
Chinese	71	9	9.1	1.8
Czech	73	9	8.3	2.0
French	80	10	8.4	1.9
German	68	9	8.3	1.5
Greek	72	11	8.7	2.0
Hebrew	69	11	9.0	2.5
Hindi	62	12	7.4	1.9
Italian	70	10	8.5	1.5
Russian	75	8	9.0	1.6
Turkish	61	13	9.1	2.0
Ukrainian	73	11	9.3	2.3
Average	70	10	9.1	1.9

Table 1: Averaged results for manual evaluation of 15 pattern clusters. P: precision (% of correct answers); % 6th: percentage of 6th selection; $Score_c$: averaged confidence score for correct selections; $Score_w$: confidence score for wrong selections.

We computed accuracy and agreement for the given answers (Table 1). We can see that for all languages above 61% of the choices were correct (comparing to 75% reported by (Davidov and Rappoport, 2008a) for a similar *monolingual* analogy test for the same set of pattern clusters). While the results are obviously lower than the corresponding single-language test, they are significantly above the random baseline of 20%[9]. Also note that as reported in (Turney, 2006), an average single-language highschool SAT grade is 57, which is lower than the scores obtained for our cross-lingual test. We can also see that for the correctly selected pairs the confidence score was very high, while the score for wrongly selected pairs was significantly lower.

4.3 Evaluation of the H-M-C relationships

In order to test how well our algorithm performs on the most common and useful relationships, hypernymy, meronymy and co-hyponymy, we automatically sampled from WordNet a set of 10 source language term pairs for each of these relationships and applied our framework to extract up to 100 target language term pairs for each of the three relationships as done above.

For each of the tested languages we presented to three human subjects for each language a short English definition of hypernymy, meronymy and co-hyponymy, along with the corresponding randomly selected 10 of 100 extracted pairs, and asked them to rank how well (0 (worst) to 10 (best)) each pair represents the described relationship. In order to reduce possible bias, we mixed in each set 3 randomly selected term pairs obtained for the other two relationships. Table 2 shows the average scores for this task.

Language	Hypernymy	Meronymy	Co-hyponymy	Random
Chinese	8.0	7.1	8.1	1.9
Czech	8.4	7.0	8.5	2.3
French	8.1	7.5	8.4	1.8
German	8.4	7.1	8.6	2.4
Greek	8.7	7.5	8.6	1.8
Hebrew	8.6	7.9	8.3	1.6
Hindi	7.5	7.1	7.8	2.2
Italian	7.9	7.8	8.2	1.5
Russian	8.6	8.1	8.9	1.7
Turkish	8.3	7.2	8.6	1.7
Ukrainian	8.2	7.7	8.2	1.7
Average	8.3	7.5	8.4	1.9

Table 2: Averaged results for hypernymy, meronymy and co-hyponymy translations. The three first columns show average scores for hypernymy, meronymy and co-hyponymy relationships. The last column shows scores for the random baseline.

We can see that our algorithm successfully detects the common relationships, achieving high scores. Also the results indicate that the patterns

[9] A reasonable random baseline omits the 6th option.

are sufficiently precise to extract at least 100 of the instances for the given salient relationships.

4.4 Evaluation of the specific relationships

To check how well our algorithm performs on some specific relationships, we examined its performance on three specific relationships explored in previous studies. We provided it with 10 source language (English) term pair examples for each of the (country, capital), (country, language) and (dog breed, origin) relationships. For each of these relationships we have factual information for every tested target language available through Wikipedia list articles. This allows us to perform an unbiased automated evaluation of the quality of the obtained target language data.

We applied our framework on these examples and generated 30 target language pairs with highest specificity for each of these relationships and languages. We compared the retrieved pairs to the factual data. Table 3 shows the precision of the results obtained for these patterns.

Language	Capital	Language	Dog breed
Chinese	0.87	0.83	0.8
Czech	0.93	0.83	0.77
French	0.97	0.9	0.87
German	0.93	0.9	0.83
Greek	0.87	0.83	0.77
Hebrew	0.83	0.8	0.8
Hindi	0.83	0.8	0.77
Italian	0.93	0.87	0.83
Russian	0.97	0.9	0.87
Turkish	0.87	0.83	0.83
Ukrainian	0.93	0.87	0.8
Average	0.9	0.85	0.81

Table 3: Precision for three specific relationship types: (country, capital), (country, language) and (dog breed,origin).

The precision observed for this task is comparable to precision obtained for Country-Capital and Country-Language in a previous single-language acquisition study (Davidov et al., 2007)[10]. The high precision observed for this task indicates that the obtained translated patterns are sufficiently good as a seed for pattern-based mining of specific relationships.

5 Conclusion

We proposed a framework which given a set of patterns defining a semantic relationship in a specific source language uses multilingual dictionaries and the web to discover a corresponding pattern cluster for a target language. In the evaluation we confirmed the applicability of our method for different languages and relationships.

The obtained set of target language pattern clusters can be used for acquisition of relationship instances as shown in our evaluation. An interesting direction for future work is to use the discovered target language pattern clusters in NLP tasks like textual entailment which require distinguishing between semantic relationships.

Applying our framework to the set of unsupervisedly discovered relationships allows a fully automated construction of a relationship dictionary, where pattern clusters in one language correspond to patten clusters in many other languages. Unlike the majority of existing machine translation systems, construction of this dictionary does not require parallel corpora. Such a dictionary can be useful for machine translation, cross-lingual textual entailment and query translation, to name just a few applications. In the future we plan to create a multilingual pattern cluster dictionary which interconnects pattern clusters from many languages and allows cross-lingual definition of lexical relationships.

References

Amasyali Fatih, 2005. Automatic Construction of Turkish Wordnet. *Signal Processing and Communications Applications Conference.*

Mishele Banko, Michael Cafarella , Stephen Soderland, Matt Broadhead, Oren Etzioni, 2007. Open information extraction from the Web. *IJCAI '07.*

Matthew Berland, Eugene Charniak, 1999. Finding parts in very large corpora. *ACL '99.*

Thatsanee Charoenporn, Virach Sornlertlamvanich, Chumpol Mokarat, and Hitoshi Isahara, 2008. Semi-automatic Compilation of Asian WordNet. *Proceedings of the 14th NLP-2008, University of Tokyo, Komaba Campus, Japan.*

Timothy Chklovski, Patrick Pantel, 2004. VerbOcean:

[10] It should be noted however that unlike previous work, we only examine the first 30 pairs and we do not use additional disambiguating words as input.

mining the web for fine-grained semantic verb relations. *EMNLP '04*.

Dmitry Davidov, Ari Rappoport, 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. *COLING-ACL '06*.

Dmitry Davidov, Ari Rappoport and Moshe Koppel, 2007. Fully Unsupervised Discovery of Concept-Specific Relationships by Web Mining. *ACL '07*.

Dmitry Davidov, Ari Rappoport. 2008a. Unsupervised Discovery of Generic Relationships Using Pattern Clusters and its Evaluation by Automatically Generated SAT Analogy Questions. *ACL '08*.

Dmitry Davidov and Ari Rappoport, 2008b. Classification of relationships between nominals using pattern clusters. *ACL '08*.

Dmitry Davidov and Ari Rappoport, 2009. Translation and Extension of Concepts Across Languages. *EACL '09*.

Roxana Girju, Adriana Badulescu, and Dan Moldovan, 2006. Automatic discovery of part-whole relations. *Computational Linguistics*, 32(1).

Roxana Girju, Marthy Hearst, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney and Yuret, D., 2007. Task 04: Classification of semantic relations between nominal at SemEval 2007. *4th Intl. Workshop on Semantic Evaluations (SemEval '07), in ACL '07*.

Hany Hassan, Ahmed Hassan and Ossama Emam, 2006. Unsupervised information extraction approach using graph mutual reinforcement. *EMNLP '06*.

Jagadeesh Jagarlamudi, A Kumaran, 2007 Cross-Lingual Information Retrieval System for Indian Languages *Working Notes for the CLEF 2007 Workshop.*

Philipp Koehn and Kevin Knight. 2001. Knowledge sources for word-level translation models. *EMNLP '01*.

Dan Moldovan, Adriana Badulescu, Marta Tatu, Daniel Antohe, and Roxana Girju, 2004. Models for the semantic classification of noun phrases. *HLT-NAACL '04 Workshop on Computational Lexical Semantics*.

Vivi Nastase, Stan Szpakowicz, 2003. Exploring noun-modifier semantic relations. *In Fifth Intl. Workshop on Computational Semantics (IWCS-5)*.

Patrick Pantel, Deepak Ravichandran, Eduard Hovy, 2004. Towards terascale knowledge acquisition. *COLING '04*.

Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, Alpa Jain, 2006. Names and similarities on the web: fact extraction in the fast lane. *COLING-ACL 06*.

Adam Pease, Christiane Fellbaum, Piek Vossen, 2008. Building the Global WordNet Grid. *CIL18*.

Benjamin Rosenfeld , Ronen Feldman, 2007. Clustering for unsupervised relation identification. *CIKM '07*.

Rion Snow, Daniel Jurafsky, Andrew Ng, 2006. Semantic taxonomy induction from heterogeneous evidence. *COLING-ACL '06*.

Peter Turney, 2005. Measuring semantic similarity by latent relational analysis, *IJCAI '05*.

Peter Turney, 2006. Expressing implicit semantic relations without supervision. *COLING-ACL '06*.

Martin Volk, Paul Buitelaar, 2002 A Systematic Evaluation of Concept-Based Cross-Language Information Retrieval in the Medical Domain. *In: Proc. of 3rd Dutch-Belgian Information Retrieval Workshop.* Leuven.

Dominic Widdows, Beate Dorow, 2002. A graph model for unsupervised Lexical acquisition. *COLING '02*.

Comparison of different algebras for inducing the temporal structure of texts

Pascal Denis[†]
† Alpage Project-Team
INRIA & Université Paris 7
pascal.denis@inria.fr

Philippe Muller[†,◇]
◇ IRIT
Université de Toulouse
muller@irit.fr

Abstract

This paper investigates the impact of using different temporal algebras for learning temporal relations between events. Specifically, we compare three interval-based algebras: Allen (1983) algebra, Bruce (1972) algebra, and the algebra derived from the TempEval-07 campaign. These algebras encode different granularities of relations and have different inferential properties. They in turn behave differently when used to enforce global consistency constraints on the building of a temporal representation. Through various experiments on the TimeBank/AQUAINT corpus, we show that although the TempEval relation set leads to the best classification accuracy performance, it is too vague to be used for enforcing consistency. By contrast, the other two relation sets are similarly harder to learn, but more useful when global consistency is important. Overall, the Bruce algebra is shown to give the best compromise between learnability and expressive power.

1 Introduction

Being able to recover the temporal relations (e.g., precedence, inclusion) that hold between events and other time-denoting expressions in a document is an essential part of natural language understanding. Success in this task has important implications for other NLP applications, such as text summarization, information extraction, and question answering.

Interest for this problem within the NLP community is not new (Passonneau, 1988; Webber, 1988; Lascarides and Asher, 1993), but has been recently revived by the creation of the TimeBank corpus (Pustejovsky et al., 2003), and the organization of the TempEval-07 campaign (Verhagen et al., 2007). These have seen the development of machine learning inspired systems (Bramsen et al., 2006; Mani et al., 2006; Tatu and Srikanth, 2008; Chambers and Jurafsky, 2008).

Learning the temporal stucture from texts is a difficult problem because there are numerous information sources at play (in particular, semantic and pragmatic ones) (Lascarides and Asher, 1993). An additional difficulty comes from the fact that temporal relations have logical properties that restrict the consistent graphs that can be built for a set of temporal entities (for instance the transitivity of inclusion and temporal precedence). Previous work do not attempt to directly predict globally coherent temporal graphs, but instead focus on the the simpler problem of labeling pre-selected pairs of events (i.e., a task that directly lends itself to the use of standard classification techniques). That is, they do not consider the problem of *linking* pairs of events (i.e., of determining which pairs of events are related).

Given the importance of temporal reasoning for determining the temporal structure of texts, a natural question is how to best use it within a machine-based learning approach. Following (Mani et al., 2006), prior approaches exploit temporal inferences to enrich the set of training instances used for learning. By contrast, (Bramsen et al., 2006) use temporal relation compositions to provide constraints in a global inference problem (on the slightly different task of ordering passages in medical history records). (Tatu and Srikanth, 2008) and (Chambers and Jurafsky, 2008) combine both approaches and use temporal reasoning both during training and decoding. Interestingly, these approaches use different inventories of relations: (Mani et al., 2006) use the TimeML 13 relation set, while (Chambers and Jurafsky, 2008;

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 250–258,
Beijing, August 2010

Bramsen et al., 2006) use subset of these relations, namely precedence and the absence of relation.

This paper adopts a more systematic perspective and directly assesses the impact of different relation sets (and their underlying algebras) in terms of learning and inferential properties. Specifically, we compare three interval-based algebras for building classification-based systems, namely: Allen (1983)'s 13 relation algebra, Bruce (1972)'s 7 relations algebra, and the algebra underlying Tempeval-07 3 relations (henceforth, TempEval algebra). We wish to determine the best trade-off between: (i) how easy it is to learn a given set of relations, (ii) how informative are the representations produced by each relation set, and (iii) how much information can be drawn from the predicted relations using knowledge encoded in the representation. These algebras indeed differ in the number of relations they encode, and in turn in how expressive each of these relations is. From a machine learning point of view of learning, it is arguably easier to learn a model that has to decide among fewer relations (i.e., that has fewer classes). But from a representational point of view, it is better to predict relations that are as specific as possible, for composing them may restrict the prediction to more accurate descriptions of the situation. However, while specific relations potentially trigger more inferences, they are also more likely to predict inconsistent constraints. In order to evaluate these differences, we design a set of experiments on the Timebank/AQUAINT corpus, wherein we learn precise relations and vaguer ones, and evaluate them with respect to each other (when a correspondence is possible).

Section 2 briefly presents the Timebank/AQUAINT corpus. In section 3, we describe the task of temporal ordering through an example, and discuss how it should be evaluated. Section 4 then goes into more detail about the different representation possibilities for temporal relations, and some of their formal properties. Section 5 presents our methods for building temporal structures, that combines relation classifiers with global constraints on whole documents. Finally, we discuss our experimental results in section 6.

2 The Timebank/AQUAINT corpus

Like (Mani et al., 2006) and (Chambers and Jurafsky, 2008), we use the so-called OTC corpus, a corpus of 259 documents obtained by combining the Timebank corpus (Pustejovsky et al., 2003) (we use version 1.1 of the corpus) and the AQUAINT corpus.[1] The Timebank corpus consists of 186 newswire articles (and around 65,000 words), while AQUAINT has 73 documents (and around 40,000 words).

Both corpora are annotated using the TimeML scheme for tagging eventualities (events and states), dates/times, and their temporal relations. Eventualities can be denoted by verbs, nouns, and some specific constructions. The temporal relations (i.e., the so-called TLINKS) encode topological information between the time intervals of occurring eventualities. TimeML distinguishes three types of TLINKS: event-event, event-time, and time-time, giving rise to different subtasks. In this paper, we will focus on predicting event-event relations (see (Filatova and Hovy, 2001; Boguraev and Ando, 2005) for work on the other tasks). The set of temporal relations used in TLINKS mirrors the 13 Allen relations (see next section), and includes the following six relations: *before*, *begins*, *ends*, *ibefore*, *includes*, *simultaneous* and their inverses. The combined OTC corpus comprises a total of 6,139 annotated event-event TLINKS. We also make use of the additional TLINKS independently provided by (Bethard et al., 2007) for 129 of the 186 Timebank documents.

3 Task presentation and evaluation

3.1 An example

We illustrate the task of event ordering using a small fabricated, simplified example:

> Fortis bank invested$_{e_1}$ in junk bonds before the financial crisis$_{e_2}$, but got rid$_{e_3}$ of most of them during the crisis$_{e_{2bis}}$. However, the institution still went bankrupt$_{e_4}$ a year later.

[1] Both corpora are freely available from `http://www.timeml.org/site/timebank/timebank.html`.

The annotation for this temporal structure would include the following relations: e_1 is temporally before e_2, e_3 is temporally included in e_2, and e_3 is before e_4. The coreference relation between e_2 and e_{2bis} implies the equality of their temporal extension. Of course all these events may in theory be related temporally to almost any other event in the text. Events are also anchored to temporal expressions explicitly, and this is usually considered as a separate, much easier task. We will use this example throughout the rest of our presentation.

3.2 Comparing temporal annotations

Due to possible inferences, there are often many equivalent ways to express the same ordering of events, so comparisons between annotation and reference event-event pairs cannot rely on simple precision/recall measures.

Consider the above example and assume the following annotation: e_1 is before e_2, e_3 is included in e_2, and e_3 is before e_3. Without going into too much detail about the semantics of the relations used, one expects annotators to agree with the fact that it entails that e_1 is before e_3, among other things. So the annotation is equivalent to a larger set of relations. In some cases, the inferred information is disjunctive (the relation holding between two events is a subset of possible "simple" relations, such as "before or included").

Nowadays, the given practice is to compute some sort of transitive closure over the network of constraints on temporal events (usually expressed in the well-studied Allen algebra (Allen, 1983)), and compute agreements over the saturated structures. Specifically, we can compare the sets of *simple* temporal relations that are deduced from it (henceforth, the "strict" metric), or measure the agreement between the whole graphs, including disjunctions (Verhagen et al., 2007) (henceforth, the "relaxed" metric).[2] Under this latter metric, precision (resp. recall) of a prediction for a pair of events consisting of a set S of relations with respect to a set of relations R inferred from the reference, is computed as $|S \cap R|/|S|$ (resp. $|S \cap R|/|R|$).

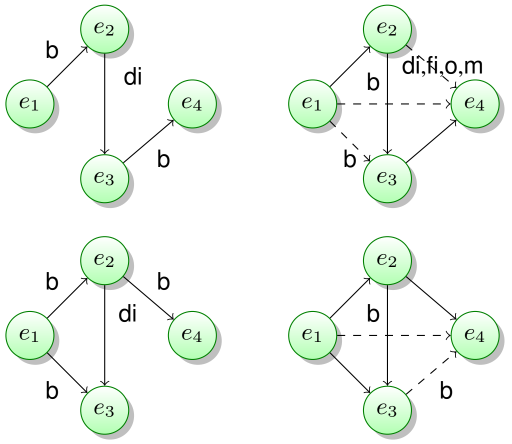


Figure 1: Two non-equivalent annotations of the same situations (left) and their transitive closure in Allen's algebra (right, with new relations only). b stands for Allen's *before* relation, m for *meet*, o for *overlap*, di and fi for the inverses of *during* and *finish*, respectively.

Figure 1 illustrates the point of these "saturated" representations, showing two raw annotations of our example on the left (top and bottom) and their closures on the right. The raw annotations share only 2 relations (between e_1 and e_2, and e_3 and e_4), but their transitive closures agree also on the relations between e_1 and e_3, e_1 and e_4, and e_3 and e_4. They still differ on the relation between e_2 and e_4, but only because one is much more specific than the other, something that can only be taken into account by a partial credit scoring function.

For this example, the "strict" metric yields precision and recall scores of 5/5 and 5/6, when comparing the top annotation against the bottom one. By contrast, the "relaxed" metric (introduced in the TempEval-07) yields precision and recall scores of (5+0.2)/6 and 6/6, respectively.

We now turn to the issue of the set of relations chosen for the task of expressing temporal information in texts.

4 Temporal representations

Because of the inferential properties of temporal relations, we have seen that the same situation can be expressed in different ways, and some relations can be deduced from others. The need for

[2] Taking into account disjunctions means giving partial credit to disjunctions approximating the reference relation (possibly disjunctive itself), see next section.

a precise reasoning framework has been present in previous attempts at the task (Setzer et al., 2006), and people have moved to a set of handmade rules over ad hoc relations to more widely accepted temporal reasoning frameworks, such as algebras of temporal relations, the most famous being Allen's interval algebra.

An algebra of relations can be defined on any set of relations that are mutually exclusive (two relations cannot hold at the same time between two entities) and exhaustive (at least one relation must hold between two given entities). The algebra starts from a set of simple, atomic, relations $U = \{r_1, r_2, ...\}$, and a general relation is a subset of U, interpreted as a disjunction of the relations it contains. From there, we can define union and intersection of relations as classical set union and intersection of the base relations they consist of. Moreover, one can define a composition of relations as follows:

$$(r_1 \circ r_2)(x, z) \leftrightarrow \exists y\ r_1(x, y) \wedge r_2(y, z)$$

In words, a relation between x and z can be computed from what is known between (x and y) and (y and z). By computing beforehand the $n \times n$ compositions of base relations of U, we can compute the composition of any two general relations (because $r \cap r' = \emptyset$ when r, r' are basic and $r \neq r'$):

$$\{r_1, r_2, ...r_k\} \circ \{s_1, s_2, ...s_m\} = \bigcup_{i,j}(r_i \circ s_j)$$

Saturating the graph of temporal constraints means applying these rules to all compatible pairs of constraints in the graph and iterating until a fixpoint is reached. In Allen's algebra there are 13 relations, determined by the different relations that can hold between two intervals endpoints (before, equals, after). These relations are: b (*before*), m (*meet*), o (*overlap*), s (*start*), f (*finish*), d (*during*), their inverses (bi, mi, oi, si, fi, di) and = (*equal*), see figure 2.[3]

It is important to see that a general approach to temporal ordering of events cannot restrict itself to a subset of these and still use the power of inferences to complete a situation, because composition of information is stable only on restricted subsets. And using all of them means generating numerous disjunctions of relations.

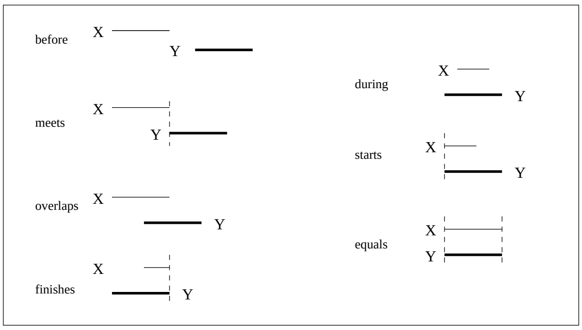


Figure 2: Allen's thirteen relations between two temporal intervals

Allen relations are convenient for reasoning purposes, but might too precise for representing natural language expressions, and that's why recent evaluation campaigns such as TempEval-07 have settled on vaguer representations. TempEval-07 uses three relations called *before*, *overlaps* and *after*, which we note b_t, o_t, and bi_t.[4] These all correspond to disjunctions of Allen relations: $\{\mathsf{b,m}\}_a$, $\{\mathsf{o,d,s,=,f}\}_a$ and its inverse, and $\{\mathsf{bi,mi}\}_a$, respectively. These representations can be converted to Allen relations, over which the same inference procedures can be applied, and then expressed back as (potentially disjunctive) TempEval relations. They thus form a sub-algebra of Allen's algebra, if we add their possible disjunctions.

In fact, starting from the base relations, only $\{\mathsf{b,o}\}_t$, $\{\mathsf{bi,o}\}_t$, and *vague* (i.e., the disjunction of all relations) can be inferred (besides the base relations). This is a consequence of the stability of so-called convex relations in Allen algebra. Note that an even simpler schema is used in (Chambers and Jurafsky, 2008), where only TempEval *before* and *after* and the *vague* relation are used.

We propose to consider yet another set of relation, namely relations from (Bruce, 1972). These provide an intermediate level of representation, since they include 7 simple relations. These are

[3] TimeML uses somewhat different names, with obvious mappings, except *ibefore* ("immediately before") for m, and *iafter* ("immediately after") for mi.

[4] When it is not obvious, we will use subscript symbols to indicate the particular algebra that is used (e.g., b_t is the before relation in TempEval).

also expressible as disjunctions of Allen relations; they are: *before* (b_b), *after* (bi_b) (with the same semantics as TempEval's b_t and bi_t), *equals* ($=_b$, same as $=_a$), *includes* (i, same as Allen's $\{\mathsf{s,d,f}\}_a$), *overlaps* (o_b, same as o_a), *included* (ii) and *is-overlapped* (oi_b), their inverse relations. The equivalences between the three algebras is shown table 1.

Allen	Bruce	Tempeval
before meet	before	before
overlaps	overlaps	overlaps
starts during finishes	included	
overlapsi	is-overlapped	
startsi duringi finishesi	includes	
meeti beforei	after	after
equals	equals	equals

Table 1: Correspondances between temporal algebras. A relation ranging over multiple cells is equivalent to a disjunction of all the relations within these cells.

Considering a vaguer set is arguably more adequate for natural language expressions while at the same time this specific set preserves at least the notions of temporal order and inclusion (contrary to the TempEval scheme), which have strong inferential properties: they are both transitive, and their composition yields simple relations; overlap allows for much weaker inferences. Figure 3 shows part of our example from the introduction expressed in the three cases: with Allen relations, the most precise, with Bruce relations and TempEval relations, with dotted lines showing the extent of the vagueness of the temporal situations in each case (with respect to the most precise Allen description). We can see that TempEval relations lose quickly all information that is not before or after, while Bruce preserves inference combining precedence and temporal inclusion.

Information can be converted from one algebra to the other, since vaguer algebras are based on relations equivalent to disjunctions in Allen algebra. But conversion from a precise relation to a vaguer one and back to a more precise algebra leads to information loss. Hence on figure 3, the original Allen relation: $e_3\ \mathsf{d}_a\ e_2$ is converted to: $e_3\ \mathsf{o}_t\ e_2$ in TempEval, which converts back into the much less informative: $e3\ \{\mathsf{o}, \mathsf{d}, \mathsf{s}, =, \mathsf{f}, \mathsf{oi}, \mathsf{si}, \mathsf{fi}, \mathsf{di}\}_a\ e_2$. We will use these translations during our system evaluation to have a common comparison point between representations.

5 Models

5.1 Algebra-based classifiers

In order to compare the impact of the different algebras described in section 4, we build three event pair classification models corresponding to each relation set. The resulting Allen-based, Bruce-based, and Tempeval-based models therefore contain 13, 7, and 3 class labels, respectively.[5] For obvious sparsity issues, we did not include classes corresponding to disjunctive relations, as there are $2^{|R|}$ possible disjunctions for each relation set R.

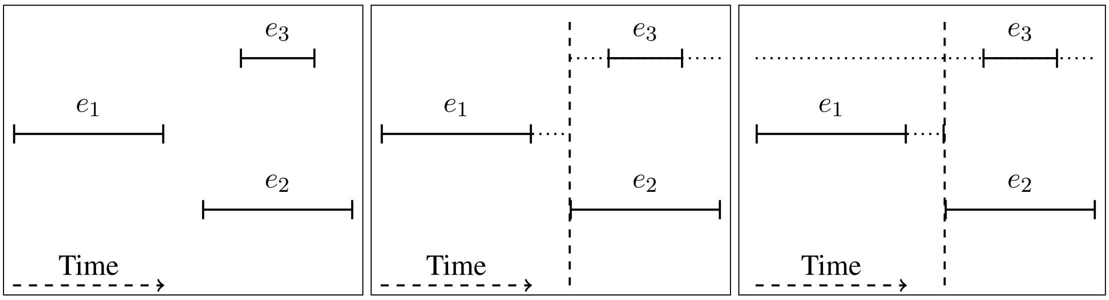
For training our models, we experiment with 4 various configurations that correspond to ways of expanding the set of training examples. Specifically, these configurations vary in: (i) whether or not we added the additional "Bethard relations" to the initial OTC annotations (Bethard et al., 2007), (ii) whether or not we applied saturation over the set of annotated relations.

5.2 Features

Our feature set for the various models is similar to that used by previous work, including binary features that encode event string as well as the five TimeML attributes and their possible values:

- **aspect**: none, prog, perfect, prog perfect
- **class**: report, aspectual, state, I-state I-action, perception, occurrence
- **modality**: none, to, should, would, could can, might
- **polarity**: positive, negative
- **tense**: none, present, past, future

[5] Our TempEval model actually has a fourth label for the *identity* relation. The motivations behind the inclusion of this extra label are: (i) this relation is linguistically motivated and comparatively easy to learn (for a lot of instances of this relation are cases of anaphora, which are often signaled by identical strings) (ii) this relation triggers a lot of specific inferences.



(a) Allen:
$(e_1 \mathsf{b}_a e_2 \wedge e_3 \mathsf{d}_a e_2) \rightarrow e_1 \mathsf{b}_a e_3$

(b) Bruce:
$(e_1 \mathsf{b}_b e_2 \wedge e_3 \mathsf{d}_b e_2) \rightarrow e_1 \mathsf{b}_b e_3$

(c) Tempeval:
$(e_1 \mathsf{b}_t e_2 \wedge e_3 \mathsf{o}_t e_2) \rightarrow e_1 \{\mathsf{b}_t, \mathsf{o}_t\} e_3$

Figure 3: Comparing loss of inferential power in algebras: hard lines show the actual temporal model, exactly expressed in Allen relations (a); dotted lines show the vagueness induced by alternative schemes, and the inference that can or cannot still be made in each algebra, (b) and (c).

Additional binary features check agreement for same attribute (e.g., the same tense). Finally, we add features that represent the distance between two events (in number of sentences, and in number of intervening events). [6]

5.3 Training set generation

Our generic training procedure works as follows. For each document, we scan events in their order of appearance in the text. We create a training instance $inst_{(e_i,e_j)}$ for each *ordered* pair of events (e_i, e_j): if (e_i, e_j) (resp. (e_j, e_i)) corresponds to an annotated relation r, then we label $inst_{(e_i,e_j)}$ with the label r (resp. its inverse r^{-1}).

5.4 Parameter estimation

All of these classifiers are maximum entropy models (Berger et al., 1996). Parameter estimation was performed with the Limited Memory Variable Metric algorithm (Malouf, 2002) implemented in the Megam package.[7]

5.5 Decoding

We consider two different decoding procedures. The first one simply mirrors the training procedure just described, scanning pairs of events in the order of the text, and sending each pair to the classifier. The pair is then labeled with the label outputted by the classifier (i.e., the label receiving the highest probability). No attempt is made to guarantee the consistency of the final temporal graph.

Our second inference procedure works as follows. As in the previous method, we scan the events in the order of the text, and create ordered pairs of events that we then submit to the classifier. But the difference is that we saturate the graph after each classification decision to make sure that the graph created so far is coherent. In case where the classifier predicts a relation whose addition results in an incoherent graph, we try the next highest probability relation, and so on, until we find a coherent graph. This greedy procedure is similar to the Natural Reading Order (NRO) inference procedure described by (Bramsen et al., 2006).

6 Experiments and results

We perform two main series of experiments for comparing our different models. In the first series, we measure the accuracy of the Allen-, Bruce-, and Tempeval-based models on predicting the correct relation for the event-event TLINKS annotated in the corpus. In the second series, we saturate the event pair relations produced by the classifiers (combined with NRO search to enforce global coherence) and compare the predicted graphs against the saturated event-event TLINKS.

6.1 Experiment settings

All our models are trained and tested with 5-fold cross-validation on the OTC documents. For eval-

[6] These were also encoded as binary features, and the various feature values were binned in order to avoid sparseness.

[7] Available from `http://www.cs.utah.edu/~hal/megam/`.

uation, we use simple accuracy for the first series of experiments, and two "strict" and "relaxed" precision/recall measures described in section 3 for the other series. For each type of measures, we report scores with respect to both Allen and TemEval relation sets. All scores are reported using macro-averaging. Out of the 259 temporal graphs present in OTC, we found that 54 of them were actually inconsistent when saturated; the corresponding documents were therefore left out of the evaluation.[8] Given the rather expensive procedure involved in the NRO decoding (saturating an inconsistent graph "erases" all relations), we skipped 8 documents wich were much longer than the rest, leaving us with 197 documents for our final experiments.

6.2 Event-event classification

Table 2 summarizes the accuracy scores of the different classifiers on the event-event TLINKS of OTC. We only report the best configuration for each model. For the TempEval-based model, we found that the best training setting was when Bethard annotations were added to the original TimeML annotations, but with no saturation.[9] For Allen and Bruce models, neither Bethard's relations nor saturation helps improve classification accuracy. In fact, saturation degrades performance, which can be explained by the fact that saturation reinforces the bias towards already over-represented relations.[10] The best accuracy performances are obtained by the Allen-based and TempEval-based classifiers, each one performing better in its own algebra (with 47.0% and 54.0%). This is not surprising, since these classifiers were specifically trained to optimize their respective metrics. The Bruce-based classifier is slightly better than the Allen-based one in TempEval, but also slightly worse than TempEval-based classifier in Allen.

	Allen Acc.	TempEval Acc.
Allen	47.0	48.9
Bruce	N/A	49.3
TempEval	N/A	54.0

Table 2: Accuracy scores for Allen, Bruce, and TempEval classifiers on event-event TLINKS, expressed in Allen or TempEval algebra. Scores for Bruce and TempEval models into Allen are left out, since they predict (through conversion) disjunctive relations for all relations but equality.

Our accuracy scores for Allen, and TempEval-based classifiers are somewhat lower than the ones reported for similar systems by (Mani et al., 2006) and (Chambers and Jurafsky, 2008), respectively. These differences are likely to come from the fact that: (i) (Mani et al., 2006) perform a 6-way classification, and not a 13-way classification[11], and (ii) (Chambers and Jurafsky, 2008) use a relation set that is even more restrictive than TempEval's.

6.3 Saturated graphs

Table 3 summarizes the various precision/recall scores of the graph obtained by saturating the classifiers predictions (potentially altered by NRO) against the event-event saturated graph. These results contrast with the accuracy results presented in table 2: while the TempEval-based model was the best model in classification accuracy in TempEval, it is now outperformed by both the Allen- and Bruce-based systems (this with or with using NRO). The best system in TempEval is actually Bruce-based system, with 52.9 and 62.8 for the strict/relaxed metrics, respectively. The results suggest that this algebra might actually offer the best trade-off between learnanility and expressive power. The use of NRO to restore global coherence yields important gains (10 points) in the relaxed metric for both Allen- and Bruce-based systems (although they do not convert into gains in the strict metric). Unsuprisingly, the best model on the Allen set remains Allen-based model (and this time the use of NRO results in gains on the strict metric). Predictions without

[8] Because there is no way to trace the relation(s) responsible for an inconsistency without analysing the whole set of annotations of a text, and considering that it usually happens on very long texts, we did not attempt to manually correct the annotations.

[9] This is actually consistent with similar findings made by (Chambers and Jurafsky, 2008).

[10] For instance, for Allen relations, there are roughly 50% of *before-after* relations before saturation but 73% of them after saturation.

[11] This is only possible because they order the event-event pairs before submitting them to the classifier.

System	Allen						Tempeval					
	RELAX			STRICT			RELAX			STRICT		
	R	P	F1	R	P	F1	R	P	F1	R	P	F1
Allen	57.5	46.7	**51.5**	49.6	56.2	**52.7**	62.0	50.3	55.5	50.4	57.1	53.6
Bruce	46.0	39.0	42.1	18.0	44.0	25.9	62.9	52.6	**57.3**	50.9	57.0	**53.8**
Tempeval	37.1	35.9	36.5	14.0	44.0	21.2	49.3	47.1	48.2	21.7	44.2	29.1
Allen$_{NRO}$	44.8	60.1	**51.3**	57.2	62.9	**59.9**	63.8	67.0	65.3	45.2	60.6	51.8
Bruce$_{NRO}$	46.3	53.1	49.5	13.9	45.3	21.2	65.5	71.8	**68.5**	46.6	61.1	**52.9**
Tempeval$_{NRO}$	37.1	35.9	36.5	13.9	44.3	21.2	49.3	47.1	48.2	21.7	44.2	29.1

Table 3: Comparing Allen-, Bruce-, Tempeval-based classifiers saturated predictions on saturated event-event graph. The $_{NRO}$ subscript indicates whether the system uses NRO or not. Evaluation are given with respect to both Allen and Tempeval relation sets.

NRO yielded between 7.5 and 9% of inconsistent saturated graphs that were ignored by the evaluation, which means this impacted recall measures only.

7 Related work

Early work on temporal ordering (Passonneau, 1988; Webber, 1988; Lascarides and Asher, 1993) concentrated on studying the knowledge sources at play (such as tense, aspect, lexical semantics, rhetorical relations). The development of annotated resources like the TimeBank corpus (Pustejovsky et al., 2003) has triggered the development of machine learning systems (Mani et al., 2006; Tatu and Srikanth, 2008; Chambers and Jurafsky, 2008).

More recent work uses automatic classification methods, based on the TimeBank and Acquaint corpus, either as is, with inferential enrichment for training (Mani et al., 2006; Chambers et al., 2007), or supplied with the corrections of (Bethard et al., 2007), or are restricted to selected contexts, such as intra-sentential event relations (Li et al., 2004; Lapata and Lascarides, 2006). All of these assume that event pairs are preselected, so the task is only to determine what is the most likely relation between them. The best scores are obtained with the added assumption that the event-event pair can be pre-ordered (thus reducing the number of possible labels by 2).

More recently, (Bramsen et al., 2006) and subsequently (Chambers and Jurafsky, 2008) propose to use an Integer Linear Programming solver to enforce the consistency of a network of constraints while maximizing the score of local classification decisions. But these are restricted to the relations BEFORE and AFTER, which have very strong inference properties that cannot be generalised to other relations. The ILP strategy is not likely to scale up very well for richer relation sets, for the number of possible relations between two events (and thus the number of variables to put in the LP solver for each pair) is the order of $2^{|R|}$ (where R is the relation set), and each transitivity constraints generates an enormous amount of constraints.

8 Conclusion

We have investigated the role played by ontological choices in temporal representations by comparing three algebras with different granularities of relations and inferential powers. Our experiments on the Timebank/AQUAINT reveal that the TempEval relation set provides the best overall classification accuracy, but it provides much less informative temporal structures, and it does not provide enough inferences for being useful for enforcing consistency. By contrast, the other two relation sets are significantly harder to learn, but provide more richer inferences and are therefore more useful when global consistency is important. Bruce's 7 relations-based model appears to perform best in the TempEval evaluation, suggesting that this algebra provides the best trade-off between learnability and expressive power.

References

Allen, James. 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, pages 832–843.

Berger, A., S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Bethard, Steven, James H. Martin, and Sara Klingenstein. 2007. Timelines from text: Identification of syntactic temporal relations. In *International Conference on Semantic Computing*, pages 11–18, Los Alamitos, CA, USA. IEEE Computer Society.

Boguraev, Branimir and Rie Ando. 2005. TimeML-compliant text analysis for temporal reasoning. In Kaelbling, Leslie Pack and Fausto Giunchiglia, editors, *Proceedings of IJCAI05*, pages 997–1003.

Bramsen, Philip, Pawan Deshpande, Yoong Keok Lee, and Regina Barzilay. 2006. Inducing temporal graphs. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 189–198, Sydney, Australia, July. Association for Computational Linguistics.

Bruce, B. 1972. A model for temporal references and its application in a question answering program. *Artificial Intelligence*, 3(1-3):1–25.

Chambers, Nathanael and Daniel Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 698–706, Honolulu, Hawaii, October. Association for Computational Linguistics.

Chambers, Nathanael, Shan Wang, and Daniel Jurafsky. 2007. Classifying temporal relations between events. In *ACL*. The Association for Computer Linguistics.

Filatova, Elena and Eduard Hovy. 2001. Assigning time-stamps to event-clauses. In Mani, I., J. Pustejovsky, and R Gaizauskas, editors, *The Language of Time: A Reader*. Oxford University Press.

Lapata, Maria and Alex Lascarides. 2006. Learning sentence-internal temporal relations. *J. Artif. Intell. Res. (JAIR)*, 27:85–117.

Lascarides, Alex and Nicholas Asher. 1993. Temporal interpretation, discourse relations and common sense entailment. *Linguistics and Philosophy*, 16:437–493.

Li, Wenjie, Kam-Fai Wong, Guihong Cao, and Chunfa Yuan. 2004. Applying machine learning to chinese temporal relation resolution. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 582–588, Barcelona, Spain, July.

Malouf, Robert. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Workshop on Natural Language Learning*, pages 49–55, Taipei, Taiwan.

Mani, Inderjeet, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 753–760, Sydney, Australia, July. Association for Computational Linguistics.

Passonneau, Rebecca J. 1988. A computational model of the semantics of tense and aspect. *Computational Linguistics*, 14(2):44–60.

Pustejovsky, James, Patrick Hanks, Roser Saurí, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. 2003. The TIMEBANK Corpus. In *Proceedings of Corpus Linguistics*, pages 647–656, Lancaster University, UK, March.

Setzer, Andrea, Robert Gaizauskas, and Mark Hepple. 2006. The Role of Inference in the Temporal Annotation and Analysis of Text. *Language Resources and Evaluation*, 39:243–265.

Tatu, Marta and Munirathnam Srikanth. 2008. Experiments with reasoning for temporal relations between events. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 857–864, Manchester, UK, August. Coling 2008 Organizing Committee.

Verhagen, Marc, Robert Gaizauskas, Franck Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 - 15: TempEval Temporal Relation Identification. In *Proceedings of SemEval workshop at ACL 2007*, Prague, Czech Republic, June. Association for Computational Linguistics, Morristown, NJ, USA.

Webber, Bonnie Lynn. 1988. Tense as discourse anaphor. *Computational Linguistics*, 14(2):61–73.

Generating Learner-Like Morphological Errors in Russian

Markus Dickinson
Indiana University
md7@indiana.edu

Abstract

To speed up the process of categorizing learner errors and obtaining data for languages which lack error-annotated data, we describe a linguistically-informed method for generating learner-like morphological errors, focusing on Russian. We outline a procedure to select likely errors, relying on guiding stem and suffix combinations from a segmented lexicon to match particular error categories and relying on grammatical information from the original context.

1 Introduction

Work on detecting grammatical errors in the language of non-native speakers covers a range of errors, but it has largely focused on syntax in a small number of languages (e.g., Vandeventer Faltin, 2003; Tetreault and Chodorow, 2008). In more morphologically-rich languages, learners naturally make many errors in morphology (Dickinson and Herring, 2008). Yet for many languages, there is a major bottleneck in system development: there are not enough error-annotated learner corpora which can be mined to discover the nature of learner errors, let alone enough data to train or evaluate a system. Our perspective is that one can speed up the process of determining the nature of learner errors via semi-automatic means, by generating plausible errors.

We set out to generate linguistically-plausible morphological errors for Russian, a language with rich inflections. Generating learner-like errors has practical and theoretical benefits. First, there is the issue of obtaining training data; as Foster and Andersen (2009) state, "The ideal situation for a grammatical error detection system is one where a large amount of labelled positive *and* negative evidence is available." Generated errors can bridge this gap by creating realistic negative evidence (see also Rozovskaya and Roth, 2010). As for evaluation data, generated errors have at least one advantage over real errors, in that we know precisely what the correct form is supposed to be, a problem for real learner data (e.g., Boyd, 2010).

By starting with a coarse error taxonomy, generating errors can improve categorization. Generated errors provide data for an expert—e.g., a language teacher—to search through, expanding the taxonomy with new error types or subtypes and/or deprecating error types which are unlikely. Given the lack of real learner data, this has the potential to speed up error categorization and subsequent system development. Furthermore, error generation techniques can be re-used, adjusting the errors for different learner levels, first languages, and so forth.

The error generation process can benefit by using linguistic properties to mimic learner variations. This can lead to more realistic errors, a benefit for machine learning (Foster and Andersen, 2009), and can also provide feedback for the linguistic representation used to generate errors by, e.g., demonstrating under which linguistic conditions certain error types are generated and under which they are not.

We are specifically interested in generating Russian morphological errors. To do this, we need a knowledge base representing Russian morphology, allowing us to manipulate linguistic properties. After outlining the coarse error taxonomy

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 259–267,
Beijing, August 2010

(section 2), we discuss enriching a part-of-speech (POS) tagger lexicon with segmentation information (section 3). We then describe the steps in error generation (section 4), highlighting decisions which provide insight for the analysis of learner language, and show the impact on POS tagging in section 5.

2 Error taxonomy

Russian is an inflecting language with relatively free word order, meaning that morphological syntactic properties are often encoded by affixes. In (1a), for example, the verb начина needs a suffix to indicate person and number, and ет is the third person singular form.[1] By contrast, (1b) illustrates a paradigm error: the suffix ит is third singular, but not the correct one. Generating such a form requires having access to individual morphemes and their linguistic properties.

(1) a. начина+ет [*nachina+et*]
begin-3s

b. *начина+ит [*nachina+it*]
begin-3s (diff. verb paradigm)

This error is categorized as a suffix error in figure 1, expanding the taxonomy in Dickinson and Herring (2008). Stem errors are similarly categorized, with *Semantic errors* defined with respect to a particular context (e.g., using a different stem than required by an activity).

For formation errors (#3), one needs to know how stems relate. For instance, some verbs change their form depending on the suffix, as in (2). In (2c), the stem and suffix are morphologically compatible, just not a valid combination. One needs to know that мож is a variant of мог.

(2) a. мог+ут [*mog+ut*]
can-3p

b. мож+ет [*mozh+et*]
can-3s

c. *мож+ут [*mozh+ut*] (#3)
can-3p (wrong formation)

0. Correct: The word is well-formed.
1. Stem errors:
   (a) Stem spelling error
   (b) Semantic error
2. Suffix errors:
   (a) Suffix spelling error
   (b) Lexicon error:
      i. Derivation error: The wrong POS is used (e.g., a noun as a verb).
      ii. Inherency error: The ending is for a different subclass (e.g., inanimate as an animate noun).
   (c) Paradigm error: The ending is from the wrong paradigm.
3. Formation errors: The stem does not follow appropriate spelling/sound change rules.
4. Syntactic errors: The form is correct, but used in an in appropriate syntactic context (e.g., nominative case in a dative context)

- Lexicon incompleteness: The form may be possible, but is not attested.

Figure 1: Error taxonomy

Using a basic lexicon without such knowledge, it is hard to tell formation errors apart from lexicon incompleteness (see section 4.2.2). If можут (2c) is generated and is not in the lexicon, we do not know whether it is misformed or simply unattested. In this paper, we group together such cases, since this allows for a simpler and more quickly-derivable lexicon.

We have added syntactic errors, whereas Dickinson and Herring (2008) focused on strictly morphological errors. Learners make syntactic errors (e.g., Rubinstein, 1995; Rosengrant, 1987), and when creating errors, a well-formed word may result. In the future, syntactic errors can be subdivided (Boyd, 2010).

This classification is of *possible* errors, making no claim about the *actual* distribution of learner errors, and does not delve into issues such as errors stemming from first language interference (Rubinstein, 1995). Generating errors from the possible types allows one to investigate which types are plausible in which contexts.

[1] For examples, we write the Cyrillic form and include a Roman transliteration (SEV 1362-78) for ease of reading.

It should be noted that we focus on inflectional morphology in Russian, meaning that we focus on suffixes. Prefixes are rarely used in Russian as inflectional markers; for example, prefixes mark semantically-relevant properties for verbs of motion. The choice of prefix is thus related to the overall word choice, an issue discussed under *Random stem generation* in section 4.2.4.

3 Enriching a POS lexicon

To create errors, we need a segmented lexicon with morphological information, as in (3). Here, the word могу (*mogu*, 'I am able to') is split into stem and suffix, with corresponding POS tags.[2]

(3) a. мог,Vm-----a-p,у,Vmip1s-a-p
b. мож,Vm-----a-p,ет,Vmip3s-a-p
c. мог,Vm-----a-p,NULL,Vmis-sma-p

The freely-available POS lexicon from Sharoff et al. (2008), specifically the file for the POS tagger TnT (Brants, 2000), contains full words (239,889 unique forms), with frequency information. Working with such a rich database, we only need segmentation, providing a quickly-obtained lexicon (cf. five years for a German lexicon in Geyken and Hanneforth, 2005).

In the future, one could switch to a different tagset, such as that in Hana and Feldman (2010), which includes reflexivity, animacy, and aspect features. One could also expand the lexicon, by adapting algorithms for analyzing unknown words (e.g., Mikheev, 1997), as suggested by Feldman and Hana (2010). Still, our lexicon continues the trend of linking traditional categories used for tagging with deeper analyses (Sharoff et al., 2008; Hana and Feldman, 2010).[3]

3.1 Finding segments/morphemes

We use a set of hand-crafted rules to segment words into morphemes, of the form: if the tag is x and the word ends with y, make y the suffix. Such rules are easily and quickly derivable from a textbook listing of paradigms. For certain exceptional cases, we write word-specific rules. Additionally, we remove word, tag pairs indicating punctuation or non-words (PUNC, SENT, -).

One could use a sophisticated method for lemmatizing words (e.g., Chew et al., 2008; Schone and Jurafsky, 2001), but we would likely have to clean the lexicon later; as Feldman and Hana (2010) point out, it is difficult to automatically guess the entries for a word, without POS information. Essentially, we write precise rules to specify part of the Russian system of suffixes; the lexicon then provides the stems for free.

We use the lexicon for generating errors, but it should be compatible with analysis. Thus, we focus on suffixes for beginning and intermediate learners. We can easily prune or add to the rule set later. From an analysis perspective, we need to specify that certain grammatical properties are in a tag (see below), as an analyzer is to support the provision of feedback. Since the rules are freely available,[4] changing these criteria for other purposes is straightforward.

3.1.1 Segmentation rules

We have written 1112 general morphology rules and 59 rules for the numerals 'one' through 'four,' based on the *Nachalo* textbooks (Ervin et al., 1997). A rule is simply a tag, suffix pair. For example, in (4), Ncmsay (Noun, common, masculine, singular, accusative, animate [yes]) words should end in either а (*a*) or я (*ya*).

(4) a. Ncmsay, а
b. Ncmsay, я

A program consults this list and segments a word appropriately, requiring at least one character in the stem. In the case where multiple suffixes match (e.g., ени (*eni*) and и (*i*) for singular neuter locative nouns), the longer one is chosen, as it is unambiguously correct.

We add information in 101 of the 1112 rules. All numerals, for instance, are tagged as Mc-s (Numeral, cardinal, [unspecified gender], singular). The tagset in theory includes properties such as case; they just were not marked (see footnote 6, though). Based on the ending, we add all

[2] POS tags are from the compositional tagset in Sharoff et al. (2008). A full description is at: http://corpus.leeds.ac.uk/mocky/msd-ru.html.

[3] This lexicon now includes lemma information, but each word is not segmented (Erjavec, 2010).

[4] http://cl.indiana.edu/~boltundevelopment/

possible analyses. Using an optional output tag, in (5), Mc-s could be genitive (g), locative (l), or dative (d) when it ends in и (*i*). These rules increase ambiguity, but are necessary for learner feedback.

(5) a. Mc-s, и, Mc-sg
 b. Mc-s, и, Mc-sl
 c. Mc-s, и, Mc-sd

In applying the rules, we generate stem tags, encoding properties constant across suffixes. Based on the word's tag (e.g., Ncmsay, cf. (4)) a stem is given a more basic tag (e.g., Ncm--y).

3.2 Lexicon statistics

To be flexible for future use, we have only enriched 90% of the words (248,014), removing every 10th word. Using the set of 1112 rules results in a lexicon with 190,450 analyses, where *analyses* are as in (3). For these 190,450 analyses, there are 117 suffix forms (e.g., я, *ya*) corresponding to 808 suffix analyses (e.g., <я, Ncmsay>). On average 3.6 suffix tags are observed with each stem-tag pair, but 22.2 tags are compatible, indicating incomplete paradigms.

4 Generating errors

4.1 Basic procedure

Taking the morpheme-based lexicon, we generate errors by randomly combining morphemes into full forms. Such randomness must be constrained, taking into account what types of errors are likely to occur.

The procedure is given in figure 2 and detailed in the following sections. First, we use the contextually-determined POS tag to restrict the space of possibilities. Secondly, given that random combinations of a stem and a suffix can result in many unlikely errors, we guide the combinations, using a loose notion of likelihood to ensure that the errors fall into a reasonable distribution. After examining the generated errors, one could restrict the errors even further. Thirdly, we compare the stem and suffix to determine the possible types of errors. A full form may have several different interpretations, and thus, lastly, we select the best interpretation(s).

1. Determine POS properties of the word to be generated (section 4.2.1).
2. Generate a full-form, via *guided* random stem and suffix combination (section 4.2.4).
3. Determine possible error analyses for the full form (section 4.2.2).
4. Select the error type(s) from among multiple possible interpretations (section 4.2.3).

Figure 2: Error generation procedure

By trying to determine the best error type in step 4, the generation process can provide insight into error analysis. This is important, given that suffixes are highly ambiguous; for example, ой (*-oj*) has at least 6 different uses for adjectives. Analysis is not simply generation in reverse, though. Importantly, error generation relies upon the context POS tag for the *intended* form, for the whole process. To morphologically analyze the corrupted data, one has to POS tag *corrupted* forms (see section 5).

4.2 Corruption

We use a corpus of 5 million words automatically tagged by TnT (Brants, 2000) and freely available online (Sharoff et al., 2008).[5] Because we want to make linguistically-informed corruptions, we corrupt only the words we have information for, identifying the words in the corpus which are found in the lexicon with the appropriate POS tag.[6] We also select only words which have inflectional morphology: nouns, verbs, adjectives, pronouns, and numerals.[7]

4.2.1 Determining word properties (step 1)

We use the POS tag to restrict the properties of a word, regardless of how exactly we corrupt it. Either the stem and its tag or the suffix and its tag

[5] See http://corpus.leeds.ac.uk/mocky/.

[6] We downloaded the TnT lexicon in 2008, but the corpus in 2009; although no versions are listed on the website, there are some discrepancies in the tags used (e.g., numeral tags now have more information). To accommodate, we use a looser match for determining whether a tag is known, namely checking whether the tags are compatible. In the future, one can tweak the rules to match the newer lexicon.

[7] Adverbs inflect for comparative forms, but we do not consider them here.

can be used as an invariant, to guide the generated form (section 4.2.4). In (6a), for instance, the adjective (`Af`) stem or plural instrumental suffix (`Afp-pif`) can be used as the basis for generation.

(6) a. Original: серыми (*serymi*, 'gray')
↦ сер/Af+ыми/Afp-pif
b. Corrupted: сер+ой (*seroj*)

The error type is defined in terms of the original word's POS tag. For example, when we generate a correctly-formed word, as in (6b), it is a syntactic error if it does not match this POS tag.

4.2.2 Determining error types (step 3)

Before discussing word corruption in step 2 (section 4.2.4), we need to discuss how error types are determined (this section) and how to handle multiple possibilities (section 4.2.3), as these steps help guide step 2. After creating a corrupted word, we elucidate all possible interpretations in step 3 by comparing each suffix analysis with the stem. If the stem and suffix form a legitimate word (in the wrong context), it is a syntactic error. Incompatible features means a derivation or inherency error, depending upon which features are incompatible. If the features are compatible, but there is no attested form, it is either a paradigm error—if we know of a different suffix with the same grammatical features—or a formation/incompleteness issue, if not.

This is a crude morphological analyzer (cf. Dickinson and Herring, 2008), but bases its analyses on what is known about the invariant part of the original word. If we use ыми (*ymi*) from (6a) as an invariant, for instance, we know to treat it as a plural instrumental adjective ending, regardless of any other possible interpretations, because that is how it was used in this context.

4.2.3 Selecting the error type (step 4)

Corrupted forms may have many possible analyses. For example, in (6b), the suffix ой (*oj*) has been randomly attached to the stem сер (*ser*). With the stem fixed as an adjective, the suffix could be a feminine locative adjective (syntactic error), a masculine nominative adjective (paradigm error), or an instrumental feminine noun (derivation error). Given what learners are likely to do, we can use some heuristics to restrict the set of possible error types.

First, we hypothesize that a correctly-formed word is more likely a correct form than a misformed word. This means that correct words and syntactic errors—correctly-formed words in the wrong context—have priority over other error types. For (6b), for instance, the syntactic error outranks the paradigm and derivation errors.

Secondly, we hypothesize that a contextually-appropriate word, even if misformed, is more likely the correct interpretation than a contextually-inappropriate word. When we have cases where there is: a) a correctly-formed word not matching the context (a syntactic error), and b) a malformed word which matches the context (e.g., a paradigm error), we list both possibilities.

Finally, derivation errors seem less likely than the others (a point confirmed by native speakers), giving them lower priority. Given these heuristics, not only can we rule out error types after generating new forms, but we can also split the error generation process into different steps.

4.2.4 Corrupting selected words (step 2)

Using these heuristics, we take a known word and generate errors based on a series of choices. For each choice, we randomly generate a number between 0 and 1 and choose based on a given threshold. Thresholds should be reset when more is known about error frequency, and more decisions added as error subtypes are added.

Decision #1: Correct forms The first choice is whether to corrupt the word or not. Currently, the threshold is set at 0.5. If we corrupt the word, we continue on to the next decision.

Decision #2: Syntactic errors We can either generate a syntactic or a morphological error. On the assumption that syntactic errors are more common, we currently set a threshold of 0.7, generating syntactic errors 70% of the time and morphological form errors 30% of the time.

To generate a correct form used incorrectly, we extract the stem from the word and randomly select a new suffix. We keep selecting a suffix until we obtain a valid form.[8] An example is given in (7): the original (7a) is a plural instrumental adjective, unspecified for gender; in (7b), it is singular nominative feminine.

(7) a. серыми глазами .
gray eyes .
Afp-**pi**f Ncmpin SENT

b. серая глазами .
Afp**fsn**f Ncmpin SENT

One might consider ensuring that each error differs from the original in only one property. Or one might want to co-vary errors, such that, in this case, the adjective and noun both change from instrumental to nominative. While this is easily accomplished algorithmically, we do not know whether learners obey these constraints. Generating errors in a relatively unbounded way can help pinpoint these types of constraints.

While the form in (7b) is unambiguous, syntactic errors can have more than one possible analysis. In (8), for instance, this word could be corrupted with an -ой (*-oj*) ending, indicating feminine singular genitive, instrumental, or locative. We include all possible forms.

(8) серой глазами .
Afpfsg.Afpfsi.Afpfsl Ncmpin SENT

Likewise, considering the heuristics in section 4.2.3, generating a syntactic error may lead to a form which may be contextually-appropriate. Consider (9): in (9a), the verb-preposition combination requires an accusative (Ncns**a**n). By changing -о to -е, we generate a form which could be locative case (Ncns**l**n, type #4) or, since -е can be an accusative marker, a misformed accusative with the incorrect paradigm (#2c). We list both possibilities.

(9) a. ... смотрел в небо
... (he) looked into the sky
... Vmis-sma-p Sp-a Ncnsan

b. ... в небе
... Sp-a Ncns**a**n+2c.Ncns**l**n+4

Syntactic errors obviously conflate many different error types. The taxonomy for German from Boyd (2010), for example, includes selection, agreement, and word order errors. Our syntactic errors are either selection (e.g., wrong case as object of preposition) or agreement errors (e.g., subject-verb disagreement in number). However, without accurate syntactic information, we cannot divvy up the error space as precisely. With the POS information, we can at least sort errors based on the ways in which they vary from the original (e.g., incorrect case).

Finally, if no syntactic error can be derived, we revert to the correct form. This happens when the lexicon contains only one form for a given stem. Without changing the stem, we cannot generate a new form which is verifiably correct.

Decision #3: Morphological errors The next decision is: should we generate a true morphological error or a spelling error? We currently bias this by setting a 0.9 threshold. The process for generating morphological errors (0.9) is described in the next few sections, after which spelling errors (0.1) are described. Surely, 10% is an underestimate of the amount of spelling errors (cf. Rosengrant, 1987); however, for refining a morphological error taxonomy, biasing towards morphological errors is appropriate.

Decision #4: Invariant morphemes When creating a context-dependent morphological error, we have to ask what the unit, or morpheme, is upon which the full form is dependent. The final choice is thus to select whether we keep the stem analysis constant and randomize the suffix or keep the suffix and randomize the stem. Consider that the stem is the locus of a word's semantic properties, and the (inflectional) suffix reflects syntactic properties. If we change the stem of a word, we completely change the semantics (error type #1b). Changing the suffix, on the other hand, creates a morphological error with the same basic semantics. We thus currently randomly generate a suffix 90% of the time.

Random suffix generation Randomly attaching a suffix to a fixed stem is the same procedure used above to generate syntactic errors. Here, however, we force the form to be incorrect, not allowing syntactic errors. If attaching a suffix re-

[8] We ensure that we do not generate the original form, so that the new form is contextually-inappropriate.

sults in a correct form (contextually-appropriate or not), we re-select a random suffix.

Similarly, the intention is to generate inherency (#2bii), paradigm (#2c), and formation (#3) errors (or lexicon incompleteness). All of these seem to be more likely than derivation (#2bi) errors, as discussed in section 4.2.3. If we allow any suffix to combine, we will overwhelmingly find derivation errors. As pointed out in Dickinson and Herring (2008), such errors can arise when a learner takes a Russian noun, e.g., душ (*dush*, 'shower') and attempts to use it as a verb, as in English, e.g., душу (*dushu*) with first person singular morphology. In such cases, we have the wrong stem being used with a contextually-appropriate ending. Derviation errors are thus best served with random stem selection, as described in the next section. To rule out derivation errors, we only keep suffix analyses which have the same major POS as the stem.

For some stems, particular types of errors are impossible to generate. a) Inherency errors do not occur for underspecified stems, as happens with adjectives. For example, нов- (*nov-*, 'new') is an adjective stem which is compatible with any adjective ending. b) Paradigm errors cannot occur for words whose suffixes in the lexicon have no alternate forms; for instance, there is only one way to realize a third singular nominative pronoun. c) Lexicon incompleteness cannot be posited for a word with a complete paradigm. These facts show that the generated error types are biased, depending upon the POS and the completeness of the lexicon.

Random stem generation Keeping the suffix fixed and randomly selecting a stem ties the generated form to the syntactic context, but changes the semantics. Thus, these generated errors are firstly semantic errors (#1b), featuring stems inappropriate for the context, in addition to having some other morphological error. The fact that, given a context, we have to generate two errors lends weight to the idea that these are less likely.

A randomly-generated stem will most likely be of a different POS class than the suffix, resulting in a derivation error (#2bi). Further, as with all morphological errors, we restrict the generated word not to be a correctly-formed word, and we do not allow the stem or the suffix to be closed class items. It makes little sense to put noun inflections on a preposition, for example, and derivation errors involve open class words.[9]

Spelling errors For spelling errors, we create an error simply by randomly inserting, deleting, or substituting a single character in the word.[10] This will either be a stem (#1a) or a suffix (#2a) error. It is worth noting that since we know the process of creating this error, we are able to compartmentalize spelling errors from morphological ones. An error analyzer, however, will have a harder time distinguishing them.

5 Tagging the corpus

Figure 3 presents the distribution of error types generated, where *Word* refers to the number of words with a particular error type, as opposed to the count of error type+*POS* pairs, as each word can have more than one POS for an error type (cf. (9b)). For the 780,924 corrupted words, there are 2.67 error type+POS pairs per corrupted word. Inherency (#2bii) errors in particular have many tags per word, since the same suffix can have multiple similar deviations from the original (cf. (8)). Figure 3 shows that we have generated roughly the distribution we wanted, based on our initial ideas of linguisic plausibility.

Type	Word	POS	Type	Word	POS
1a	19,661	19,661	1b-2bi	11,772	11,772
2a	6,560	6,560	1b-2bii	5,529	5,529
2bii	150,710	749,292	1b-2c	279	279
2c	94,211	94,211	1b-3+	1,770	1,770
4	524,269	721,051			
3+	83,763	208,208	1b-all	19,350	19,350

Figure 3: Distribution of generated errors

Without an error detection system, it is hard to gauge the impact of the error generation process. Although it is not a true evaluation of the error generation process, as a first step, we test a POS

[9] Learners often misuse, e.g., prepositions, but these errors do not affect morphology. Future work should examine the relation between word choice and derivation errors, including changes in prefixes.

[10] One could base spelling errors on known or assumed phonological confusions (cf. Hovermale and Martin, 2008).

tagger against the newly-created data. This helps test the difficulty of tagging corrupted forms, a needed step in the process of analyzing learner language. Note that for providing feedback, it seems desirable to have the POS tagger match the tag of the corrupted form. This is a different goal than developing POS taggers which are robust to noise (e.g., Bigert et al., 2003), where the tag should be of the original word.

To POS tag, we use the HMM tagger TnT (Brants, 2000) with the model from `http://corpus.leeds.ac.uk/mocky/`. The results on the generated data are in figure 4, using a lenient measure of accuracy: a POS tag is correct if it matches any of the tags for the hypothesized error types. The best performance is for uncorrupted known words,[11] but notable is that, out of the box, the tagger obtains 79% precision on corrupted words when compared to the generated tags, but is strongly divergent from the original (no longer correct) tags. Given that 67% $(\frac{524{,}269}{780{,}924})$ of words have a syntactic error—i.e., a well-formed word in the wrong context—this indicates that the tagger is likely relying on the form in the lexicon more than the context.

	Gold Tags		
	Original	Error	# words
Corrupted	3.8%	79.0%	780,924
Unchanged:			
Known	92.1%	92.1%	965,280
Unknown	81.9%	81.9%	3,484,909
Overall	72.1%	83.4%	5,231,113

Figure 4: POS tagging results, comparing tagger output to *Original* tags and *Error* tags

It is difficult to break down the results for corrupted words by error type, since many words are ambiguous between several different error types, and each interpretation may have a different POS tag. Still, we can say that words which are syntactic errors have the best tagging accuracy. Of the 524,269 words which may be syntactic errors, TnT matches a tag in 96.1% of cases. Suffix spelling errors are particularly in need of improvement: only 17.3% of these words are correctly tagged (compared to 62% for stem spelling errors). With an ill-formed suffix, the tagger simply does not have reliable information. To improve tagging for morphological errors, one should investigate which linguistic properties are being incorrectly tagged (cf. sub-tagging in Hana et al., 2004) and what roles distributional, morphological, or lexicon cues should play in tagging learner language (see also Díaz-Negrillo et al., 2010).

6 Conclusions and Outlook

We have developed a general method for generating learner-like morphological errors, and we have demonstrated how to do this for Russian. While many insights are useful for doing error analysis (including our results for POS tagging the resulting corpus), generation proceeds from knowing grammatical properties of the original word. Generating errors based on linguistic properties has the potential to speed up the process of categorizing learner errors, in addition to creating realistic data for machine learning systems. As a side effect, we also added segmentation to a wide-coverage POS lexicon.

There are several directions to pursue. The most immediate step is to properly evaluate the quality of generated errors. Based on this analysis, one can refine the taxonomy of errors, and thereby generate even more realistic errors in a future iteration. Additionally, building from the initial POS tagging results, one can work on generally analyzing the morphology of learner language, including teasing apart what information a POS tagger needs to examine and dealing with multiple hypotheses (Dickinson and Herring, 2008).

Acknowledgements

I would like to thank Josh Herring, Anna Feldman, Jennifer Foster, and three anonymous reviewers for useful comments on this work.

[11] *Known* here refers to being in the enriched lexicon, as these are the cases we specificaly did not corrupt.

References

Bigert, Johnny, Ola Knutsson and Jonas Sjöbergh (2003). Automatic Evaluation of Robustness and Degradation in Tagging and Parsing. In *Proceedings of RANLP-2003*. Borovets, Bulgaria, pp. 51–57.

Boyd, Adriane (2010). EAGLE: an Error-Annotated Corpus of Beginning Learner German. In *Proceedings of LREC-10*. Malta.

Brants, Thorsten (2000). TnT – A Statistical Part-of-Speech Tagger. In *Proceedings of ANLP-00*. Seattle, WA, pp. 224–231.

Chew, Peter A., Brett W. Bader and Ahmed Abdelali (2008). Latent Morpho-Semantic Analysis: Multilingual Information Retrieval with Character N-Grams and Mutual Information. In *Proceedings of Coling 2008*. Manchester, pp. 129–136.

Díaz-Negrillo, Ana, Detmar Meurers, Salvador Valera and Holger Wunsch (2010). Towards interlanguage POS annotation for effective learner corpora in SLA and FLT. *Language Forum* .

Dickinson, Markus and Joshua Herring (2008). Developing Online ICALL Exercises for Russian. In *The 3rd Workshop on Innovative Use of NLP for Building Educational Applications*. Columbus, OH, pp. 1–9.

Erjavec, Tomaž (2010). MULTEXT-East Version 4: Multilingual Morphosyntactic Specifications, Lexicons and Corpora. In *Proceedings of LREC-10*. Malta.

Ervin, Gerard L., Sophia Lubensky and Donald K. Jarvis (1997). *Nachalo: When in Russia ...* . New York: McGraw-Hill.

Feldman, Anna and Jirka Hana (2010). *A Resource-light Approach to Morpho-syntactic Tagging*. Amsterdam: Rodopi.

Foster, Jennifer and Oistein Andersen (2009). GenERRate: Generating Errors for Use in Grammatical Error Detection. In *The 4th Workshop on Innovative Use of NLP for Building Educational Applications*. Boulder, CO, pp. 82–90.

Geyken, Alexander and Thomas Hanneforth (2005). TAGH: A Complete Morphology for German Based on Weighted Finite State Automata. In *FSMNLP 2005*. Springer, pp. 55–66.

Hana, Jirka and Anna Feldman (2010). A Positional Tagset for Russian. In *Proceedings of LREC-10*. Malta.

Hana, Jirka, Anna Feldman and Chris Brew (2004). A Resource-light Approach to Russian Morphology: Tagging Russian using Czech resources. In *Proceedings of EMNLP-04*. Barcelona, Spain.

Hovermale, DJ and Scott Martin (2008). Developing an Annotation Scheme for ELL Spelling Errors. In *Proceedings of MCLC-5 (Midwest Computational Linguistics Colloquium)*. East Lansing, MI.

Mikheev, Andrei (1997). Automatic Rule Induction for Unknown-Word Guessing. *Computational Linguistics* 23(3), 405–423.

Rosengrant, Sandra F. (1987). Error Patterns in Written Russian. *The Modern Language Journal* 71(2), 138–145.

Rozovskaya, Alla and Dan Roth (2010). Training Paradigms for Correcting Errors in Grammar and Usage. In *Proceedings of HLT-NAACL-10*. Los Angeles, California, pp. 154–162.

Rubinstein, George (1995). On Case Errors Made in Oral Speech by American Learners of Russian. *Slavic and East European Journal* 39(3), 408–429.

Schone, Patrick and Daniel Jurafsky (2001). Knowledge-Free Induction of Inflectional Morphologies. In *Proceedings of NAACL-01*. Pittsburgh, PA.

Sharoff, Serge, Mikhail Kopotev, Tomaž Erjavec, Anna Feldman and Dagmar Divjak (2008). Designing and evaluating Russian tagsets. In *Proceedings of LREC-08*. Marrakech.

Tetreault, Joel and Martin Chodorow (2008). The Ups and Downs of Preposition Error Detection in ESL Writing. In *Proceedings of COLING-08*. Manchester.

Vandeventer Faltin, Anne (2003). Syntactic error diagnosis in the context of computer assisted language learning. Thèse de doctorat, Université de Genève, Genève.

Resolving Object and Attribute Coreference in Opinion Mining

Xiaowen Ding
Department of Computer Science
University of Illinois at Chicago
dingxwsimon@gmail.com

Bing Liu
Department of Computer Science
University of Illinois at Chicago
liub@cs.uic.edu

Abstract

Coreference resolution is a classic NLP problem and has been studied extensively by many researchers. Most existing studies, however, are generic in the sense that they are not focused on any specific text. In the past few years, opinion mining became a popular topic of research because of a wide range of applications. However, limited work has been done on coreference resolution in opinionated text. In this paper, we deal with object and attribute coreference resolution. Such coreference resolutions are important because without solving it a great deal of opinion information will be lost, and opinions may be assigned to wrong entities. We show that some important features related to opinions can be exploited to perform the task more accurately. Experimental results using blog posts demonstrate the effectiveness of the technique.

1 Introduction

Opinion mining has been actively researched in recent years. Researchers have studied the problem at the document level (e.g., Pang et al., 2002; Tuney, 2002; Gamon et al., 2005) sentence and clause level (Wilson et al., 2004; Kim and Hovy, 2004), word level (e.g., Andreevskaia and Bergler, 2006; Hatzivassiloglou and McKeown, 1997; Esuli and Sebastiani, 2006; Kanayama and Nasukawa, 2006; Qiu et al., 2009), and attribute level (Hu and Liu 2004; Popescu and Etzioni, 2005; Ku et al., 2006; Mei et al., 2007; Titov and McDonald 2008). Here attributes mean different aspects of an object that has been commented on. Let us use the following example blog to illustrate the problem: "*I bought a Canon S500 camera yesterday. It looked beautiful. I took a few photos last night. They were amazing*". "It" in the second sentence refers to "*Canon S500 camera*", which is called an *object*. "They" in the fourth sentence refers to "photos", which is called an *attribute* of the object "*Canon S500 camera*". The usefulness of coreference resolution in this case is clear. Without resolving them, we lose opinions. That is, although we know that the second and fourth sentences express opinions, we do not know on what. Without knowing the opinion target, the opinion is of limited use. In (Nicolov et al., 2008), it was shown based on manually annotated data that opinion mining results can be improved by 10% if coreference resolution is used (the paper did not provide an algorithm).

In this paper, we propose the problem of *object and attribute coreference resolution* – the task of determining which mentions of objects and attributes refer to the same entities. Note that here *entities* refer to both objects and attributes, not the traditional named entities. To our knowledge, limited work has been done on this problem in the opinion mining context apart from a prior study on resolving opinion sources (or holders) (Stoyanov and Cardie 2006). Opinion sources or holders are the persons or organizations that hold some opinions on objects and attributes. In this paper, we do not deal with source resolution as we are mainly interested in opinion texts on the web, e.g., reviews, discussions and blogs. In such environments opinion sources are usually the authors of the posts, which are displayed in Web pages.

This work follows the *attribute-based opinion mining model* in (Hu and Liu 2004; Popescu and Etzioni, 2005). In their work, attributes are called *features*. We do not use the term "feature" in this paper to avoid confusion with the term "feature" used in machine learning.

Our primary interests in this paper are opi-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 268–276,
Beijing, August 2010

nions expressed on products and services, which are called *objects*. Each object is described by its parts/components and attributes, which are all called *attributes* for simplicity.

This paper takes the supervised learning approach to solving the problem. The key contribution of this paper is the design and testing of two novel opinion related features for learning. The first feature is based on sentiment analysis of normal sentences (non-comparative sentences), comparative sentences, and the idea of sentiment consistency. For example, we have the sentences, "*The Sony camera is better than the Canon camera. It is cheap too*." It is clear that "It" means "Sony" because in the first sentence, the opinion on "Sony" is positive (comparative positive), but negative (comparative negative) on "Canon", and the second sentence is positive. Thus, we can conclude that "It" refers to "Sony" because people usually express sentiments in a consistent way. It is unlikely that "It" refers to "Canon". This is the idea of *sentiment consistency*. As we can see, this feature requires the system to have the ability to determine positive and negative opinions expressed in normal and comparative sentences.

The second feature considers what objects and attributes are modified by what opinion words. Opinion words are words that are commonly used to express positive or negative opinions, e.g., good, best, bad, and poor. Consider the sentences, "*The picture quality of the Canon camera is very good. It is not expensive either*." The question is what "It" refers to, "Canon camera" or "picture quality". Clearly, we know that "It" refers to "Canon camera" because "picture quality" cannot be expensive. To make this feature work, we need to identify what opinion words are usually associated with what objects or attributes, which means that the system needs to discover such relationships from the corpus.

These two features give significant boost to the coreference resolution accuracy. Experimental results based on three corpora demonstrate the effectiveness of the proposed features.

2 Related Work

Coreference resolution is an extensively studied NLP problem (e.g., Morton, 2000; Ng and Cardie, 2002; Gasperin and Briscoe, 2008). Early knowledge-based approaches were domain and linguistic dependent (Carbonell and Brown 1988), where researchers focused on diverse lexical and grammatical properties of referring expressions (Soon et al., 2001; Ng and Cardie, 2002; Zhou et al., 2004). Recent research relied more on exploiting semantic information. For example, Yang et al. (2005) used the semantic compatibility information, and Yang and Su (2007) used automatically discovered patterns integrated with semantic relatedness information, while Ng (2007) employed semantic class knowledge acquired from the Penn Treebank. Versley et al. (2008) used several kernel functions in learning.

Perhaps, the most popular approach is based on supervised learning. In this approach, the system learns a pairwise function to predict whether a pair of noun phrases is coreferent. Subsequently, when making coreference resolution decisions on unseen documents, the learnt pairwise noun phrase coreference classifier is run, followed by a clustering step to produce the final clusters (coreference chains) of coreferent noun phrases. For both training and testing, coreference resolution algorithms rely on feature vectors for pairs of noun phrases that encode lexical, grammatical, and semantic information about the noun phrases and their local context.

Soon et al. (2001), for example, built a noun phrase coreference system based on decision trees and it was tested on two standard coreference resolution data sets (MUC-6, 1995; MUC-7, 1998), achieving performance comparable to the best-performing knowledge based coreference engines at that time. The learning algorithm used 12 surface-level features. Our proposed method builds on this system with additional sentiment related features. The features inherit from this paper includes:

Distance Feature: Its possible values are 0, 1, 2, 3 and so on which captures the sentence distance between two entities.

Antecedent-pronoun feature, anaphor-pronoun feature: If the candidate antecedent or anaphor is a pronoun, it is true; false otherwise.

Definite noun phrase feature: The value is true if the noun phrase starts with "the"; false otherwise.

Demonstrative noun phrase feature: The value is true if the noun phrase starts with the word "this", "that", "these", or "those"; false otherwise.

Number agreement feature: If the candidate antecedent and anaphor are both singular or both plural, the value is true; otherwise false.

Both-proper-name feature: If both the candidates are proper nouns, which are determined by capitalization, return true; otherwise false.

Alias feature: It is true if one candidate is an alias of the other or vice versa; false otherwise.

Ng and Cardie (2002) expanded the feature set of Soon et al. (2001) from 12 to 53 features. The system was further improved by Stoyanov and Cardie (2006) who gave a partially supervised clustering algorithm and tackled the problem of opinion source coreference resolution.

Centering theory is a linguistic approach tried to model the variation or shift of the main subject of the discourse in focus. In (Grosz et al., 1995; Tetreault, 2001), centering theory was applied to sort the antecedent candidates based on the ranking of the forward-looking centers, which consist of those discourse entities that can be interpreted by linguistic expressions in the sentences. Fang et al. (2009) employed the centering theory to replace the grammatical role features with semantic role information and showed superior accuracy performances.

Ding et al. (2009) studied the *entity assignment* problem. They tried to discover the product names discussed in forum posts and assign the product entities to each sentence. The work did not deal with product attributes.

Unsupervised approaches were also applied due to the cost of annotating large corpora. Ng (2008) used an Expectation-Maximization (EM) algorithm, and Poon and Domingos (2008) applied Markov Logic Network (MLN).

Another related work is the indirect anaphora, known as bridging reference. It arises when an entity is part of an earlier mention. Resolving indirect anaphora requires background knowledge (e.g. Fan et al., 2005), and it is thus not in the scope of this paper.

Our work differs from these existing studies as we work in the context of opinion mining, which gives us extra features to enable us to perform the task more effectively.

3 Problem of Object and Attribute Coreference Resolution

In general, opinions can be expressed on anything, e.g., a product, an individual, an organization, an event, a topic, etc. Following (Liu, 2006), we also use the term *object* to denote an named entity that has been commented on. The object has a set of *components* (or parts) and also a set of *attributes*. For simplicity, *attribute* is used to denote both component and attribute in this paper. Thus, we have the two concepts, *object* and *attribute*.

3.1 Objective

Task objective: To carry out coreference resolution on objects and attributes in opinion text.

As we discussed in the introduction section, coreference resolution on objects and attributes is important because they are the core entities on which people express opinions. Due to our objective, we do not evaluate other types of coreferences. We assume that objects and entities have been discovered by an existing system (e.g., Hu and Liu 2004, Popescu and Etzioni 2005). Recall that a coreference relation holds between two noun phrases if they refer to the same entity. For example, we have the following three consecutive sentences:

s_1: I love *the nokia n95* but not sure how good *the flash* would be?

s_2: and also *it* is quite expensive so anyone got any ideas?

s_3: I will be going on contract so as long as i can get a good deal of *it*.

"it" in s_2 refers to the entity "the nokia n95" in s_1. In this case, we call "the nokia n95" the *antecedent* and pronoun "it" in s_2 the *anaphor*. The referent of "it" in s_3 is also "the nokia n95", so the "it" in s_3 is coreferent with the "it" in s_2.

Our task is thus to decide which mentions of objects and attributes refer to the same entities.

3.2 Overview of Our Approach

Like traditional conference resolution, we employ the supervised learning approach by including additional new features. The main steps of our approach are as follows:

Preprocessing: We first preprocess the corpus by running a POS tagger[1], and a Noun Phrase finder[2]. We then produce the set O-NP which includes both possible objects, attributes and other noun phrases. The noun phrases are

[1] http://nlp.stanford.edu/software/tagger.shtml

[2] http://crfchunker.sourceforge.net/

found using the Noun Phrase finder and the object names are consecutive NNPs. O-NP thus contains everything that needs to be resolved.

Feature vector construction: To perform machine learning, we need a set of features. Similar to previous supervised learning approaches (Soon et al., 2001), a feature vector is formed for every pair of phrases in O-NP extracted in the preprocessing step. We use some of the features introduced by Soon et al. (2001) together with some novel new features that we propose in this work. Since our focus is on products and attributes in opinionated documents, we do not use personal pronouns, the gender agreement feature, and the appositive feature, as they are not essential in blogs and forum posts discussing products.

Classifier construction: Using the feature vectors obtained from the previous step, we construct the training data, which includes all pairs of manually tagged phrases that are either object names or attributes. More precisely, each pair contains at least one object or one attribute. Using the training data, a decision tree is constructed using WEKA[3].

Testing: The testing phase employs the same preprocessing and feature vector construction steps as described above, followed by the application of the learnt classifier on all candidate coreference pairs (which are represented as feature vectors). Since we are only interested in coreference information for objects and attribute noun phrases, we discard non-object and non-attribute noun phrases.

4 The Proposed New Features

On surface, object and attribute coreference resolution seems to be the same as the traditional noun phrase coreference resolution. We can apply an existing coreference resolution technique. However, as we mentioned earlier, in the opinion mining context, we can have a better solution by integrating opinion information into the traditional lexical and grammatical features. Below are several novel features that we have proposed. We use α_i to denote an antecedent candidate and α_j an anaphor candidate. Note that we will not repeat the features used in previous systems, but only focus on the new features.

4.1 Sentiment Consistency

Intuitively, in a post, if the author starts expressing opinions on an object, he/she will continue to have the same opinion on that object or its attributes unless there are contrary words such as “but” and “however”. For example, we have the following blog (an id is added before each sentence to facilitate later discussion):

> *“(1) I bought Camera-A yesterday. (2) I took a few pictures in the evening in my living room. (3) The images were very clear. (4) They were definitely better than those from my old Camera-B. (5a) It is cheap too. (5b) The pictures of that camera were blurring for night shots, but for day shots it was ok”*

The comparative sentence (4) says that Camera-A is superior to Camera-B. If the next sentence is (5a) ((5a) and (5b) are alternative sentences), “it” should refer to the superior product/object (Camera-A) because sentence (5a) expresses a positive opinion. Similarly, if the next sentence is sentence (5b) which expresses a negative opinion in its first clause, “that camera” should refer to the inferior product (Camera-B). We call this phenomenon *sentiment consistency* (*SC*), which says that consecutive sentiment expressions should be consistent with each other unless there are contrary words such as “but” and “however”. It would be ambiguous if such consistency is not observed.

Following the above observation, we further observe that if the author wants to introduce a new object o, he/she has to state the name of the object explicitly in a sentence $s_{i\text{-}1}$. The question is what happens to the next sentence s_i if we need to resolve the pronouns in s_i.

We consider several cases:

1. $s_{i\text{-}1}$ is a normal sentence (not a comparative sentence). If s_i expresses a consistent sentiment with $s_{i\text{-}1}$, it should refer to the same object as $s_{\text{i-1}}$. For example, we have
   $s_{i\text{-}1}$: *The N73* is my favorite.
   s_i: *It* can produce great pictures.
   Here “It” in s_i clearly refers to “The N73” in the first sentence $s_{i\text{-}1}$.
2. $s_{i\text{-}1}$ is a normal sentence and s_i does not express a consistent sentiment, then α_i and α_j introduced in these two sentences may not be coreferenced. For example, we have
   $s_{i\text{-}1}$: *The K800* is awesome.
   s_i: *That phone* has short battery life.

[3] http://www.cs.waikato.ac.nz/ml/weka/

Here “The K800” and “That phone” may not be a coreference pair according to sentiment consistency. “That phone” should refer to an object appeared in an earlier sentence.

3. s_{i-1} is a comparative sentence. If s_i expresses a positive (respectively negative) sentiment, the pronoun in s_i should refer to the superior (or inferior) entity in s_{i-1} to satisfy sentiment consistency. This situation is depicted in the earlier example blog. For completeness, we give another example.

   s_{i-1}: *The XBR4* is brighter than *the 5080.*
   s_i: Overall, *it* is a great choice.

   Here “it” in s_i should refer to “The XBR4” in s_{i-1} since they both have positive sentiments expressed on them.

Opinion Mining of Comparative Sentences: To deal with case (3), we need to identify superior entities from comparative sentences. In fact, we first need to find such comparative sentences. There is a prior work on identifying comparative sentences (Jindal and Liu. 2006). Since our focus is not to identify such sentences, we used several heuristic rules based on some comparative keywords, e.g. *than*, *win*, *superior*, etc. They achieve the F-score of 0.9. We then followed the opinion mining method introduced in (Ding et al. 2009) to find superior entities. Since a comparative sentence typically has entities on the two sides of a comparative keyword, i.e., “*Camera-X is better than Camera-Y*”, based on opinion mining, if the sentence is positive, then the entities before the comparative keyword is superior and otherwise they are inferior (with the negation considered).

SC Feature: The possible value for this feature is 0, 1, or 2. If α_i and α_j have the same opinion, return 1; different opinions, return 0; and if the opinions cannot be identified for one or both of them, return 2. Here is an example explaining how the feature is used in our system:

> “*My wife has currently got a Nokia 7390, which is terrible. My 6233 would always get great reception, hers would get no signal.*”

Using our algorithm for opinion mining, “hers” gets a negative opinion in the second sentence. So the value for this feature for the pair, “hers” and “a Nokia 7390”, is 1. The feature value for the pair “hers” and “My 6233” is 0. The idea is that because the first sentence expresses a negative sentiment on “a Nokia 7390”, and there is no discourse connective (such as “but” and “however”) between these two sentences. “Hers” should be talking about “a Nokia 7390” so as to satisfy sentiment consistency.

4.2 Entity and Opinion Word Association

One of the most important factors determining the orientation of opinions is the opinion words that opinion holders use to express their opinions. Different entities may be modified by different opinion words. We can use their association information with entities (both objects and attributes) to identify their coreferences.

Opinion Words: In most cases, opinions in sentences are expressed using *opinion words*. For example, the sentence, “*The picture quality is amazing*”, expresses a positive opinion on the “picture quality” attribute because of the positive opinion word “amazing”.

Researchers have compiled sets of such words for adjectives, adverbs, verbs, and nouns respectively. Such lists are collectively called the *opinion lexicon*. We obtained an opinion lexicon from the authors of (Ding et al. 2009).

It is useful to note that opinion words used to express opinions on different entities are usually different apart from some general opinion words such as good, great, bad, etc, which can express opinions on almost anything. For example, we have the following passage:

> “*i love the nokia n95 but not sure how strong the flash would be? And also it is quite expensive, so anyone got any ideas*?”

Here “strong” is an opinion word that expresses a positive opinion on “the flash”, but is seldom used to describe “the nokia n95”. “expensive”, on the other hand, should not be associated with “the flash”, but is an opinion word that indicates a negative opinion on “the nokia n95”. So “the nokia n95” is more likely to be the antecedent of “it” in the second sentence.

The question is how to find such associations of entities and opinion words. We use their co-occurrence information to measure, i.e., the *pointwise mutual information* of the two terms. First, we estimate the probability of $P(NP)$, $P(OW)$ and $P(NP\&OW)$. Here *NP* means a noun phrase, e.g., an object (attribute) after removing determiners, and *OW* means an opinion word. To compute the probability, we first count the occurrences of the words. Then the probability is computed as follow:

$$P(NP, OW) = \frac{NumofS(NP \,\&\, OW)}{TotalNumofSentence}$$

where *NumofS* is a function that gives the number of sentences that contain the particular word string. *P*(*NP, OW*) is computed in the same way. Let us use the previous example again. We compute *P*("nokia n95","expensive") as the number of sentences containing both "nokia n95" and "expensive" divided by the total number of sentences in the whole corpus.

Then we use the pointwise mutual information between a noun phrase and an opinion word to measure the association.

$$PMI(NP, OW) = \log \frac{P(NP, OW)}{P(NP)P(OW)}$$

However, this PMI value cannot be encoded directly as a feature as it only captures the local information between antecedent candidates and opinion words. That is, it cannot be used as a global feature in the classifier. We thus rank all possible antecedents of anaphor α_j based on their *PMI* values and use the ranking as the feature value. The highest ranked antecedent α_i has value 1; the second one has value 2 and so on. The candidates ranked below the fourth place all have the value 5. In the example above, if *PMI*("nokia n95", "expensive") is greater than *PMI*("flash", "expensive"), the feature for "nokia n95" and "it" pair will have a smaller value than the feature for the "flash" and "it" pair.

One may ask if we can use all adjectives and adverbs to associate with objects and attributes rather than just opinion words since most opinion words are adjectives and adverbs. We tested that, but the results were poor. We believe the reason is that there are many adjectives and adverbs which are used for all kinds of purposes and may not be meaningful for our task.

4.3 String Similarity Feature

Soon et al. (2001) has a string match feature (SOON STR), which tests whether the two noun phrases are the same string after removing determiners from each. Ng and Cardie (2002) split this feature into several primitive features, depending on the type of noun phrases. They replace the SOON STR feature with three features — PRO STR, PN STR, and WORDS STR — which restrict the application of string matching to pronouns, proper names, and non-pronominal noun phrases, respectively.

In the user generated opinion data, these may not be sufficient. For a certain product, people can have a large number of ways to express it. For example, we have

"*Panasonic TH50PZ700U VS TH50PZ77U, Which Plasma tv should I go for. The TH77U is about $500.00 more than the 700U.*"

Here "TH77U" is the same entity as "Panasonic TH50PZ77U", and "TH50PZ700U" is the same as "700U". But they cannot be easily identified by "same string" features mentioned above. Although "700U" can be solved using substring features, "TH77U" is difficult to deal with.

We employ a modified edit distance to computing a similarity score between different mentions and use that as a feature in our system. When one candidate is a substring of another, return 1; otherwise, 1 plus the edit distance.

4.4 Other Useful Features

In the machine learning approach introduced by Soon et al. (2001), they had several general features that can deal with various kinds of entities, e.g., semantic class agreement features dealing with different semantic classes like date, location, etc., and the gender agreement feature related to personal entities. However, these features are not so useful for our task because the semantic class of a product in one domain is usually consistent, and dates and locations are unlikely to be of any products that people will express their opinions. Moreover, we do not study opinion holders (as they are known in the Web environment), so personal entities are not the aspect that we concentrate on. Thus we did not use the following features: semantic class agreement features, the gender agreement feature, and appositive feature.

However, we added some specific features, which are based on two extracted entities, α_i and α_j, where α_i is the potential antecedent and α_j is the potential anaphor:

Is-between feature: Its possible values are true and false. If the words between α_i and α_j have an is-like verb (i.e., is, are, was, were, and be) between them and there is no comparative indicators, this feature has the value of true, e.g., "*The nokia e65* is *a good handset*."

In sentences similar to this example, the entities before and after "is" usually refer to the same object or attribute by a definition relation.

And the value of this feature will be true.

If "is" appears together with a comparative word, it is probably an indication that the two entities are different, and the value for this feature will be false, e.g., "*Overall the K800 is far superior to the W810*."

Has-between feature: Its possible values are also true and false. If the words between α_i and α_j have a has-like verb (i.e., has, have, and had), the value is true, and otherwise false, e.g., "*The k800* has *a 3.2 megapixel camera*."

This feature usually indicates a "part-of" relation if "has" appears between two entities. They do not refer to the same entity. Table 1 gives a summary of all the features used in our system.

5 Experiments and Discussions

5.1 Datasets

For evaluation, we used forum discussions from three domains, mobile phones, plasma and LCD TVs, and cars. Table 2 shows the characteristics of the three data sets. Altogether, we downloaded 64 discussion threads, which contain 453 individual posts with a total of 3939 sentences. All the sentences and product names were annotated strictly following the MUC-7 coreference task annotation standard[4]. Here is an example:

"*Phil had <COREF ID = "6" TYPE = "OBJ">a z610</COREF> which has <COREF ID = "7" TYPE = "ATTR">a 2MP camera</COREF>, and he never had a problem with <COREF ID = "8" TYPE = "OBJ" REF = "6">it</COREF>.*"

ID and REF features are used to indicate that there is a coreference link between two strings. ID is arbitrary but uniquely assigned to each noun phrase. REF uses the ID to indicate a coreference link. "TYPE" can be "OBJ" (an object or a product), or "ATTR" (an attribute of an object). The annotation was done by the first author and another student before the algorithm construction, and the annotated data sets will be made public for other researchers to use.

For our experiments, we used the J48-decision tree builder in WEKA, a popular of machine learning suite developed at the University of Waikato. We conducted 10-fold cross validation on each dataset.

The performances are measured using the standard evaluation measures of precision (*p*), recall (*r*) and F-score (*F*), $F = 2pr/(p+r)$. As we stated in Section 3, we are only interested in object and attributes noun phrases. So in the testing phrases, we only compute the precision and recall based on those pairs of candidates that contain at least one object or attribute noun phrase in each pair. If both of the candidates are not an object or an attribute, we ignore them.

5.2 Baseline

As the baseline systems, we duplicated two representative systems. Baseline1 is the decision tree system in Soon et al. (2001). We do not use the semantic class agreement feature, gender agreement feature and appositive feature in the original 12 features for the reason discussed in Section 4.4. Thus, the total number of features in Baseline1 is 9. The second baseline (baseline2) is based on the centering theory from the semantic perspective introduced by Fang et al. (2009). Centering theory is a theory about the local discourse structure that models the interaction of referential continuity and the salience of discourse entities in the internal organization of a text. Fang et al. (2009) extended the centering theory from the grammar level to the semantic level in tracking the local discourse focus.

5.3 Results Analysis

Table 3 gives the experimental results of the two baseline systems and our system with different features included. From Table 3, we can make several observations.

(1) Comparing the results of Baseline1 and our system with all features (Our System (All)), the new features introduced in this paper improves Baseline1 on average by more than 9% in F-score.

(2) Comparing the results of Baseline2 and our system with all features (Our System (All)), our system performs better than Baseline2 by about 3 - 5%. We also observe that centering theory (Baseline2) is indeed better than the traditional decision tree.

(3) Our system with sentiment consistency (SC) makes a major difference. It improves Baseline1 (our method is based on Baseline1) by 5-6% in F-score.

(4) With the additional feature of entity and opinion association (EOA), the results are

[4] http://www-nlpir.nist.gov/related_projects/muc/proceedings/co_task.html

Feature category	Feature	Remark
Opinion mining based features	Opinion consistency	1, if the opinion orientation of α_i is the same as α_j, 0 if the opinions are different, else 2
	Entity and opinion words association	1, 2, 3, 4, 5 which indicate the rank positive based on the *PMI* value introduced in Section 4.2
grammatical	i-Pronoun feature	1, if α_i is a pronoun, else 0
	j-Pronoun feature	1, if α_j is a pronoun, else 0
	Number agreement feature	1, if both of the noun phrases agree in numbers, else 0
	Definite feature	1, if α_j starts with the word "the", else 0
	Demonstrative feature	1, if α_j starts with the word "this", "that", "those", or "these", else 0
	Both proper-name feature	1, if α_i and α_j are both proper names, else 0
lexical	String similarity	The string similarity score between α_i and α_j
	Alias feature	1, If α_i is an alias of α_j or vice versa, else 0
Others	Distance feature	The sentence distance between the pair of noun phrases, 0 if they are in the same sentence
	Keywords between features	1, if some keywords exist between α_i and α_j, else 0. Details are discussed in Section 4.5

Table 1: Feature list: α_i denotes the antecedent candidate and α_j the anaphor candidate

	Posts	Sentences
Phone	168	1498
TVs	173	1376
Cars	112	1065
Total	453	3939

Table 2: Characteristics of the datasets

		Cellphone			TVs			Cars		
		p	r	**F**	p	r	**F**	p	r	**F**
1	Baseline1	0.66	0.57	0.**61**	0.67	0.61	0.**64**	0.70	0.63	0.**66**
2	Baseline2	0.70	0.64	0.**67**	0.72	0.65	0.**68**	0.76	0.70	0.**73**
3	Our System (SC)	0.71	0.64	0.**67**	0.73	0.66	0.**69**	0.74	0.69	0.**72**
4	Our System (SC+EOA)	0.74	0.68	0.**71**	0.74	0.68	0.**71**	0.77	0.71	0.**74**
5	Our System (All)	0.75	0.70	0.**72**	0.76	0.70	0.**73**	0.78	0.73	0.**75**

Table 3: Results of object and attribute coreference resolution

improved further by another 2-4%.

(5) Our system with all features (row 5) performs the best.

Paired *t*-tests were performed on the three systems, i.e., baseline1, baseline2, and our system (row 5). The tests show that the improvements of our method over both Baseline1 and Baseline2 are significant at the confidence level of 95% for the first two datasets. For the third dataset, the improvement over Baseline1 is also significant at the confidence level of 95%, while the improvement over Baseline2 is significant at the confidence level of 90%.

In summary, we can conclude that the new technique is effective and is markedly better than the existing methods. It is clear that the new features made a major difference.

6 Conclusion

This paper investigated the coreference resolution problem in the opinion mining context. In particular, it studied object and attribute resolutions which are crucial for improving opinion mining results. Although we still took the supervised learning approach, we proposed several novel features in the opinion mining context, e.g., sentiment consistency, and object/attribute and opinion word associations. Experimental results using forum posts demonstrated the effectiveness of the proposed technique. In our future work, we plan to further improve the method and discover some other opinion related features that can be exploited to produce more accurate results.

References

A. Andreevskaia and S. Bergler. 2006. *Mining WordNet for Fuzzy Sentiment: Sentiment Tag Extraction from WordNet Glosses*. EACL'06.

J. Carbonell and R. Brown. 1988. *Anaphora resolution: a multi-strategy approach. COLING'1988.*

G. Carenini, R. Ng, and A. Pauls. 2006. *Interactive multimedia summaries of evaluative text. IUI'06.*

X. Ding, B. Liu and L. Zhang. 2009. *Entity Discovery and Assignment for Opinion Mining Application.* KDD'09.

A. Esuli and F. Sebastiani. 2006. *Determining Term Subjectivity and Term Orientation for Opinion Mining*, EACL'06.

J. Fan, K. Barker and B. Porter. 2005. *Indirect anaphora resolution as semantic path search*. K-CAP'05.

C. Gasperin and T. Briscoe. 2008. *Statistical anaphora resolution in biomedical texts*. COLING'08

B. J. Grosz, A. K. Joshi and S. Weinstein. 1995. *Centering: a framework for modeling the local coherence of discourse*. Computational Linguistics, 21(2).

V. Hatzivassiloglou and K. McKeown. 1997. *Predicting the Semantic Orientation of Adjectives*. ACL-EACL'97.

M. Hu and B. Liu. 2004. *Mining and summarizing customer reviews*. KDD'04.

N. Jindal, and B. Liu. 2006. *Mining Comparative Sentences and Relations*. AAAI'06.

H. Kanayama and T. Nasukawa. 2006. *Fully Automatic Lexicon Expansion for Domain-Oriented Sentiment Analysis*. EMNLP'06.

S. Kim and E. Hovy. 2004. *Determining the Sentiment of Opinions*. COLING'04.

N. Kobayashi, K. Inui and Y. Matsumoto. 2007. *Extracting Aspect-Evaluation and Aspect-of Relations in Opinion Mining*. EMNLP-CoNLL'07.

F. Kong, G. Zhou, Q. Zhu and P. Qian. 2009. *Employing the Centering Theory in Pronoun Resolution from the Semantic Perspective.* EMNLP'09.

L.-W. Ku, Y.-T. Liang and H.-H. Chen. 2006. *Opinion Extraction, Summarization and Tracking in News and Blog Corpora*. CAAW'06.

B. Liu. 2006. *Web Data Mining*, Springer.

R. McDonald, K. Hannan, T Neylon, M. Wells, and J.Reynar. 2007. *Structured Models for Fine-to-Coarse Sentiment Analysis*. ACL-07.

Q. Mei, X. Ling, M. Wondra, H. Su, and C. Zhai. 2007. *Topic Sentiment Mixture: Modeling Facets and Opinions in Weblogs*. WWW'07.

T. S. Morton. 2000. *Coreference for NLP applications*. ACL'00.

V. Ng and C. Cardie. 2002. *Improving machine learning approaches to coreference resolution*. ACL'02.

V. Ng. 2007. *Semantic Class Induction and Coreference Resolution*. ACL'07.

V. Ng. 2008. *Unsupervised Models for Coreference Resolution*. EMNLP'08.

N. Nicolov, F. Salvetti and S. *Ivanova, Sentiment analysis: Does coreference matter?* AISB'2008.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. *Thumbs up? Sentiment Classification Using Machine Learning Techniques*. EMNLP'02.

H. Poon and P. Domingos. 2008. *Joint Unsupervised Coreference Resolution with Markov Logic*. EMNLP'08, 650–659.

A-M. Popescu and O. Etzioni. 2005. *Extracting product features and opinions from review*s. EMNLP'05.

Qiu, Guang, B. Liu, J. Bu and C. Chen. 2009 *Expanding Domain Sentiment Lexicon through Double Propagation*. IJCAI 2009.

W. M. Soon, H. T. Ng and D. Lim. 2001. *A machine learning approach to coreference resolution of noun phrase*. Computational Linguistics, 27(4).

V. Stoyanov, C. Cardie. 2006. *Partially supervised coreference resolution for opinion summarization through structured rule learning.* EMNLP'06.

I. Titov and R. McDonald. 2008, *A joint model of text and aspect ratings for sentiment summarization*, ACL'08.

J. Tetreault. 2001. *A corpus-based evaluation of centering and pronoun resolution*. Computational Linguistics. 27(4):507-520.

P. Turney. 2002. *Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews.* ACL'02.

Y. Versley, A. Moschitti, M. Poesio and X. Yang. 2008. *Coreference systems based on kernels methods.* COLING'08.

T. Wilson, J. Wiebe, and R. Hwa. 2004. *Just how mad are you? Finding strong and weak opinion clauses.* AAAI'04.

X. F. Yang, J. Su and C. L. Tan. 2005. *Improving Pronoun Resolution Using Statistics - Based Semantic Compatibility Information*. ACL'05.

X. F. Yang and J. Su. 2007. *Coreference Resolution Using Semantic Relatedness Information from Automatically Discovered Patterns.* ACL'07.

G. D. Zhou and J. Su. 2004. *A high-performance coreference resolution system using a multi-agent strategy*. COLING'04.

Entity Disambiguation for Knowledge Base Population

†**Mark Dredze** and †**Paul McNamee** and †**Delip Rao** and †**Adam Gerber** and ⋆**Tim Finin**
†Human Language Technology Center of Excellence, Center for Language and Speech Processing
Johns Hopkins University
⋆University of Maryland – Baltimore County
mdredze,mcnamee,delip,adam.gerber@jhu.edu, finin@umbc.edu

Abstract

The integration of facts derived from information extraction systems into existing knowledge bases requires a system to disambiguate entity mentions in the text. This is challenging due to issues such as non-uniform variations in entity names, mention ambiguity, and entities absent from a knowledge base. We present a state of the art system for entity disambiguation that not only addresses these challenges but also scales to knowledge bases with several million entries using very little resources. Further, our approach achieves performance of up to 95% on entities mentioned from newswire and 80% on a public test set that was designed to include challenging queries.

1 Introduction

The ability to identify entities like people, organizations and geographic locations (Tjong Kim Sang and De Meulder, 2003), extract their attributes (Pasca, 2008), and identify entity relations (Banko and Etzioni, 2008) is useful for several applications in natural language processing and knowledge acquisition tasks like populating structured knowledge bases (KB).

However, inserting extracted knowledge into a KB is fraught with challenges arising from natural language ambiguity, textual inconsistencies, and lack of world knowledge. To the discerning human eye, the “Bush” in “Mr. Bush left for the Zurich environment summit in Air Force One.” is clearly the US president. Further context may reveal it to be the 43rd president, George W. Bush, and not the 41st president, George H. W. Bush. The ability to disambiguate a polysemous entity mention or infer that two orthographically different mentions are the same entity is crucial in updating an entity’s KB record. This task has been variously called entity disambiguation, record linkage, or entity linking. When performed without a KB, entity disambiguation is called coreference resolution: entity mentions either within the same document or across multiple documents are clustered together, where each cluster corresponds to a single real world entity.

The emergence of large scale publicly available KBs like Wikipedia and DBPedia has spurred an interest in linking textual entity references to their entries in these public KBs. Bunescu and Pasca (2006) and Cucerzan (2007) presented important pioneering work in this area, but suffer from several limitations including Wikipedia specific dependencies, scale, and the assumption of a KB entry for each entity. In this work we introduce an entity disambiguation system for linking entities to corresponding Wikipedia pages designed for open domains, where a large percentage of entities will not be linkable. Further, our method and some of our features readily generalize to other curated KB. We adopt a supervised approach, where each of the possible entities contained within Wikipedia are scored for a match to the query entity. We also describe techniques to deal with large knowledge bases, like Wikipedia, which contain millions of entries. Furthermore, our system learns when to withhold a link when an entity has no matching KB entry, a task that has largely been neglected in prior research in cross-document entity coreference. Our system produces high quality predictions compared with recent work on this task.

2 Related Work

The information extraction oeuvre has a gamut of relation extraction methods for entities like persons, organizations, and locations, which can be classified as open- or closed-domain depending on the restrictions on extractable relations (Banko and Etzioni, 2008). Closed domain systems extract a fixed set of relations while in open-domain systems, the number and type of relations are unbounded. Extracted relations still require processing before they can populate a KB with facts: namely, entity linking and disambiguation.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 277–285,
Beijing, August 2010

Motivated by ambiguity in personal name search, Mann and Yarowsky (2003) disambiguate person names using biographic facts, like birth year, occupation and affiliation. When present in text, biographic facts extracted using regular expressions help disambiguation. More recently, the Web People Search Task (Artiles et al., 2008) clustered web pages for entity disambiguation.

The related task of cross document coreference resolution has been addressed by several researchers starting from Bagga and Baldwin (1998). Poesio et al. (2008) built a cross document coreference system using features from encyclopedic sources like Wikipedia. However, successful coreference resolution is insufficient for correct entity linking, as the coreference chain must still be correctly mapped to the proper KB entry.

Previous work by Bunescu and Pasca (2006) and Cucerzan (2007) aims to link entity mentions to their corresponding topic pages in Wikipedia but the authors differ in their approaches. Cucerzan uses heuristic rules and Wikipedia disambiguation markup to derive mappings from surface forms of entities to their Wikipedia entries. For each entity in Wikipedia, a context vector is derived as a prototype for the entity and these vectors are compared (via dot-product) with the context vectors of unknown entity mentions. His work assumes that all entities have a corresponding Wikipedia entry, but this assumption fails for a significant number of entities in news articles and even more for other genres, like blogs. Bunescu and Pasca on the other hand suggest a simple method to handle entities not in Wikipedia by learning a threshold to decide if the entity is not in Wikipedia. Both works mentioned rely on Wikipedia-specific annotations, such as category hierarchies and disambiguation links.

We just recently became aware of a system fielded by Li *et al.* at the TAC-KBP 2009 evaluation (2009). Their approach bears a number of similarities to ours; both systems create candidate sets and then rank possibilities using differing learning methods, but the principal difference is in our approach to NIL prediction. Where we simply consider absence (*i.e.,* the NIL candidate) as another entry to rank, and select the top-ranked option, they use a separate binary classifier to decide whether their top prediction is correct, or whether NIL should be output. We believe relying on features that are designed to inform whether absence is correct is the better alternative.

3 Entity Linking

We define *entity linking* as matching a textual entity mention, possibly identified by a named entity recognizer, to a KB entry, such as a Wikipedia page that is a canonical entry for that entity. An entity linking *query* is a request to link a textual entity mention in a given document to an entry in a KB. The system can either return a matching entry or `NIL` to indicate there is no matching entry. In this work we focus on linking organizations, geo-political entities and persons to a Wikipedia derived KB.

3.1 Key Issues

There are 3 challenges to entity linking:

Name Variations. An entity often has multiple mention forms, including abbreviations (Boston Symphony Orchestra vs. BSO), shortened forms (Osama Bin Laden vs. Bin Laden), alternate spellings (Osama vs. Ussamah vs. Oussama), and aliases (Osama Bin Laden vs. Sheikh Al-Mujahid). Entity linking must find an entry despite changes in the mention string.

Entity Ambiguity. A single mention, like Springfield, can match multiple KB entries, as many entity names, like people and organizations, tend to be polysemous.

Absence. Processing large text collections virtually guarantees that many entities will not appear in the KB (`NIL`), even for large KBs.

The combination of these challenges makes entity linking especially challenging. Consider an example of "William Clinton." Most readers will immediately think of the 42nd US president. However, the only two William Clintons in Wikipedia are "William de Clinton" the 1st Earl of Huntingdon, and "William Henry Clinton" the British general. The page for the 42nd US president is actually "Bill Clinton". An entity linking system must decide if either of the William Clintons are correct, even though neither are exact matches. If the system determines neither matches, should it return NIL or the variant "Bill Clinton"? If variants are acceptable, then perhaps "Clinton, Iowa" or "DeWitt Clinton" should be acceptable answers?

3.2 Contributions

We address these entity linking challenges.
Robust Candidate Selection. Our system is flexible enough to find name variants but sufficiently restrictive to produce a manageable candidate list despite a large-scale KB.
Features for Entity Disambiguation. We developed a rich and extensible set of features based on the entity mention, the source document, and the KB entry. We use a machine learning ranker to score each candidate.
Learning NILs. We modify the ranker to learn NIL predictions, which obviates hand tuning and importantly, admits use of additional features that are indicative of NIL.

Our contributions differ from previous efforts (Bunescu and Pasca, 2006; Cucerzan, 2007) in several important ways. First, previous efforts depend on Wikipedia markup for significant performance gains. We make no such assumptions, although we show that optional Wikipedia features lead to a slight improvement. Second, Cucerzan does not handle NILs while Bunescu and Pasca address them by learning a threshold. Our approach *learns* to predict NIL in a more general and direct way. Third, we develop a rich feature set for entity linking that can work with any KB. Finally, we apply a novel finite state machine method for *learning* name variations. [1]

The remaining sections describe the candidate selection system, features and ranking, and our novel approach learning NILs, followed by an empirical evaluation.

4 Candidate Selection for Name Variants

The first system component addresses the challenge of name variants. As the KB contains a large number of entries (818,000 entities, of which 35% are PER, ORG or GPE), we require an efficient selection of the relevant candidates for a query.

Previous approaches used Wikipedia markup for filtering – only using the top-k page categories (Bunescu and Pasca, 2006) – which is limited to Wikipedia and does not work for general KBs. We consider a KB independent approach to selection that also allows for tuning candidate set size. This involves a linear pass over KB entry names (Wikipedia page titles): a naive implementation took two minutes per query. The following section reduces this to under two *seconds* per query.

For a given query, the system selects KB entries using the following approach:

- Titles that are exact matches for the mention.
- Titles that are wholly contained in or contain the mention (e.g., *Nationwide* and *Nationwide Insurance*).
- The first letters of the entity mention match the KB entry title (e.g., *OA* and *Olympic Airlines*).
- The title matches a known alias for the entity (aliases described in Section 5.2).
- The title has a strong string similarity score with the entity mention. We include several measures of string similarity, including: character Dice score > 0.9, skip bigram Dice score > 0.6, and Hamming distance $<= 2$.

We did not optimize the thresholds for string similarity, but these could obviously be tuned to minimize the candidate sets and maximize recall.

All of the above features are general for any KB. However, since our evaluation used a KB derived from Wikipedia, we included a few Wikipedia specific features. We added an entry if its Wikipedia page appeared in the top 20 Google results for a query.

On the training dataset (Section 7) the selection system attained a recall of 98.8% and produced candidate lists that were three to four orders of magnitude smaller than the KB. Some recall errors were due to inexact acronyms: ABC (Arab Banking; 'Corporation' is missing), ASG (Abu Sayyaf; 'Group' is missing), and PCF (French Communist Party; French reverses the order of the pre-nominal adjectives). We also missed International Police (Interpol) and Becks (David Beckham; Mr. Beckham and his wife are collectively referred to as 'Posh and Becks').

[1] http://www.clsp.jhu.edu/ markus/fstrain

4.1 Scaling Candidate Selection

Our previously described candidate selection relied on a linear pass over the KB, but we seek more efficient methods. We observed that the above non-string similarity filters can be precomputed and stored in an index, and that the skip bigram Dice score can be computed by indexing the skip bigrams for each KB title. We omitted the other string similarity scores, and collectively these changes enable us to avoid a linear pass over the KB. Finally we obtained speedups by serving the KB concurrently[2]. Recall was nearly identical to the full system described above: only two more queries failed. Additionally, more than 95% of the processing time was consumed by Dice score computation, which was only required to correctly retrieve less than 4% of the training queries. Omitting the Dice computation yielded results in a few milliseconds. A related approach is that of canopies for scaling clustering for large amounts of bibliographic citations (McCallum et al., 2000). In contrast, our setting focuses on alignment vs. clustering mentions, for which overlapping partitioning approaches like canopies are applicable.

5 Entity Linking as Ranking

We select a single correct candidate for a query using a supervised machine learning ranker. We represent each query by a D dimensional vector $\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^D$, and we aim to select a single KB entry y, where $y \in \mathcal{Y}$, a set of possible KB entries for this query produced by the selection system above, which ensures that $\mathcal{Y}$ is small. The ith query is given by the pair $\{\mathbf{x}_i, y_i\}$, where we assume at most one correct KB entry.

To evaluate each candidate KB entry in $\mathcal{Y}$ we create feature functions of the form $f(\mathbf{x}, y)$, dependent on both the example $\mathbf{x}$ (document and entity mention) and the KB entry y. The features address name variants and entity disambiguation.

We take a maximum margin approach to learning: the correct KB entry y should receive a higher score than all other possible KB entries $\hat{y} \in \mathcal{Y}, \hat{y} \neq y$ plus some margin γ. This learning constraint is equivalent to the ranking SVM algorithm of Joachims (2002), where we define an ordered pair constraint for each of the incorrect KB entries $\hat{y}$ and the correct entry y. Training sets parameters such that score$(y) \geq$ score$(\hat{y}) + \gamma$. We used the library SVM$^{\text{rank}}$ to solve this optimization problem.[3] We used a linear kernel, set the slack parameter C as 0.01 times the number of training examples, and take the loss function as the total number of swapped pairs summed over all training examples. While previous work used a custom kernel, we found a linear kernel just as effective with our features. This has the advantage of efficiency in both training and prediction [4] – important considerations in a system meant to scale to millions of KB entries.

5.1 Features for Entity Disambiguation

200 atomic features represent $\mathbf{x}$ based on each candidate query/KB pair. Since we used a linear kernel, we explicitly combined certain features (e.g., acroynym-match AND known-alias) to model correlations. This included combining each feature with the predicted type of the entity, allowing the algorithm to learn prediction functions specific to each entity type. With feature combinations, the total number of features grew to 26,569. The next sections provide an overview; for a detailed list see McNamee et al. (2009).

5.2 Features for Name Variants

Variation in entity name has long been recognized as a bane for information extraction systems. Poor handling of entity name variants results in low recall. We describe several features ranging from simple string match to finite state transducer matching.

String Equality. If the query name and KB entry name are identical, this is a strong indication of a match, and in our KB entry names are distinct. However, similar or identical entry names that refer to distinct entities are often qualified with parenthetical expressions or short clauses. As an example, “London, Kentucky” is distinguished

[2] Our Python implementation with indexing features and four threads achieved up to 80× speedup compared to naive implementation.

[3] `www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`

[4] Bunescu and Pasca (2006) report learning tens of thousands of support vectors with their “taxonomy” kernel while a linear kernel represents all support vectors with a single weight vector, enabling faster training and prediction.

from "London, Ontario", "London, Arkansas", "London (novel)", and "London". Therefore, other string equality features were used, such as whether names are equivalent after some transformation. For example, "Baltimore" and "Baltimore City" are exact matches after removing a common GPE word like city; "University of Vermont" and "University of VT" match if VT is expanded.

Approximate String Matching. Many entity mentions will not match full names exactly. We added features for character Dice, skip bigram Dice, and left and right Hamming distance scores. Features were set based on quantized scores. These were useful for detecting minor spelling variations or mistakes. Features were also added if the query was wholly contained in the entry name, or vice-versa, which was useful for handling ellipsis (e.g., "United States Department of Agriculture" vs. "Department of Agriculture"). We also included the ratio of the recursive longest common subsequence (Christen, 2006) to the shorter of the mention or entry name, which is effective at handling some deletions or word reorderings (e.g., "Li Gong" and "Gong Li"). Finally, we checked whether all of the letters of the query are found in the same order in the entry name (e.g., "Univ Wisconsin" would match "University of Wisconsin").

Acronyms. Features for acronyms, using dictionaries and partial character matches, enable matches between "MIT" and "Madras Institute of Technology" or "Ministry of Industry and Trade."

Aliases. Many aliases or nicknames are nontrivial to guess. For example JAVA is the stock symbol for Sun Microsystems, and "Ginger Spice" is a stage name of Geri Halliwell. A reasonable way to do this is to employ a dictionary and alias lists that are commonly available for many domains[5].

FST Name Matching. Another measure of surface similarity between a query and a candidate was computed by training finite-state transducers similar to those described in Dreyer et al. (2008). These transducers assign a score to any string pair by summing over all alignments and scoring all contained character n-grams; we used n-grams of length 3 and less. The scores are combined using a global log-linear model. Since different spellings of a name may vary considerably in length (e.g., *J Miller* vs. *Jennifer Miller*) we eliminated the limit on consecutive insertions used in previous applications.[6]

5.3 Wikipedia Features

Most of our features do not depend on Wikipedia markup, but it is reasonable to include features from KB properties. Our feature ablation study shows that dropping these features causes a small but statistically significant performance drop.

WikiGraph statistics. We added features derived from the Wikipedia graph structure for an entry, like indegree of a node, outdegree of a node, and Wikipedia page length in bytes. These statistics favor common entity mentions over rare ones.

Wikitology. KB entries can be indexed with human or machine generated metadata consisting of keywords or categories in a domain-appropriate taxonomy. Using a system called *Wikitology*, Syed et al. (2008) investigated use of ontology terms obtained from the explicit category system in Wikipedia as well as relationships induced from the hyperlink graph between related Wikipedia pages. Following this approach we computed top-ranked categories for the query documents and used this information as features. If none of the candidate KB entries had corresponding highly-ranked Wikitology pages, we used this as a `NIL` feature (Section 6.1).

5.4 Popularity

Although it may be an unsafe bias to give preference to common entities, we find it helpful to provide estimates of entity popularity to our ranker as others have done (Fader et al., 2009). Apart from the graph-theoretic features derived from the Wikipedia graph, we used Google's PageRank to by adding features indicating the rank of the KB entry's corresponding Wikipedia page in a Google query for the target entity mention.

[5] We used multiple lists, including class-specific lists (i.e., for PER, ORG, and GPE) lists extracted from Freebase (Bollacker et al., 2008) and Wikipedia redirects. PER, ORG, and GPE are the commonly used terms for entity types for people, organizations and geo-political regions respectively.

[6] Without such a limit, the objective function may diverge for certain parameters of the model; we detect such cases and learn to avoid them during training.

5.5 Document Features

The mention document and text associated with a KB entry contain context for resolving ambiguity.

Entity Mentions. Some features were based on presence of names in the text: whether the query appeared in the KB text and the entry name in the document. Additionally, we used a named-entity tagger and relation finder, SERIF (Boschee et al., 2005), identified name and nominal mentions that were deemed co-referent with the entity mention in the document, and tested whether these nouns were present in the KB text. Without the NE analysis, accuracy on non-NIL entities dropped 4.5%.

KB Facts. KB nodes contain infobox attributes (or facts); we tested whether the fact text was present in the query document, both locally to a mention, or anywhere in the text. Although these facts were derived from Wikipedia infoboxes, they could be obtained from other sources as well.

Document Similarity We measured similarity between the query document and the KB text in two ways: cosine similarity with TF/IDF weighting (Salton and McGill, 1983); and using the Dice coefficient over bags of words. IDF values were approximated using counts from the Google 5-gram dataset as by Klein and Nelson (2008).

Entity Types. Since the KB contained types for entries, we used these as features as well as the predicted NE type for the entity mention in the document text. Additionally, since only a small number of KB entries had PER, ORG, or GPE types, we also inferred types from Infobox class information to attain 87% coverage in the KB. This was helpful for discouraging selection of eponymous entries named after famous entities (e.g., the former U.S. president vs. “John F. Kennedy International Airport”).

5.6 Feature Combinations

To take into account feature dependencies we created combination features by taking the cross-product of a small set of diverse features. The attributes used as combination features included entity type; a popularity based on Google’s rankings; document comparison using TF/IDF; coverage of co-referential nouns in the KB node text; and name similarity. The combinations were cascaded to allow arbitrary feature conjunctions. Thus it is possible to end up with a feature *kbtype-is-ORG AND high-TFIDF-score AND low-name-similarity*. The combined features increased the number of features from roughly 200 to 26,000.

6 Predicting NIL Mentions

So far we have assumed that each example has a correct KB entry; however, when run over a large corpus, such as news articles, we expect a significant number of entities will not appear in the KB. Hence it will be useful to predict `NIL`s.

We *learn* when to predict `NIL` using the SVM ranker by augmenting $\mathcal{Y}$ to include `NIL`, which then has a single feature unique to `NIL` answers. It can be shown that (modulo slack variables) this is equivalent to learning a single threshold τ for `NIL` predictions as in Bunescu and Pasca (2006).

Incorporating `NIL` into the ranker has several advantages. First, the ranker can set the threshold optimally without hand tuning. Second, since the SVM scores are relative within a single example and cannot be compared across examples, setting a single threshold is difficult. Third, a threshold sets a uniform standard across all examples, whereas in practice we may have reasons to favor a `NIL` prediction in a given example. We design features for `NIL` prediction that cannot be captured in a single parameter.

6.1 NIL Features

Integrating `NIL` prediction into learning means we can define arbitrary features indicative of `NIL` predictions in the feature vector corresponding to `NIL`. For example, if many candidates have good name matches, it is likely that one of them is correct. Conversely, if no candidate has high entry-text/article similarity, or overlap between facts and the article text, it is likely that the entity is absent from the KB. We included several features, such as a) the max, mean, and difference between max and mean for 7 atomic features for all KB candidates considered, b) whether any of the candidate entries have matching names (exact and fuzzy string matching), c) whether any KB entry was a top Wikitology match, and d) if the top Google match was not a candidate.

	Micro-Averaged				Macro-Averaged			
	Best	*Median*	*All Features*	*Best Features*	*Best*	*Median*	*All Features*	*Best Features*
All	0.8217	0.7108	0.7984	0.7941	0.7704	0.6861	0.7695	**0.7704**
non-NIL	0.7725	0.6352	0.7063	0.6639	0.6696	0.5335	0.6097	0.5593
NIL	0.8919	0.7891	0.8677	**0.8919**	0.8789	0.7446	0.8464	0.8721

Table 1: Micro and macro-averaged accuracy for TAC-KBP data compared to best and median reported performance. Results are shown for all features as well as removing a small number of features using feature selection on development data.

7 Evaluation

We evaluated our system on two datasets: the Text Analysis Conference (TAC) track on Knowledge Base Population (TAC-KBP) (McNamee and Dang, 2009) and the newswire data used by Cucerzan (2007) (Microsoft News Data).

Since our approach relies on supervised learning, we begin by constructing our own training corpus.[7] We highlighted 1496 named entity mentions in news documents (from the TAC-KBP document collection) and linked these to entries in a KB derived from Wikipedia infoboxes. [8] We added to this collection 119 sample queries from the TAC-KBP data. The total of 1615 training examples included 539 (33.4%) PER, 618 (38.3%) ORG, and 458 (28.4%) GPE entity mentions. Of the training examples, 80.5% were found in the KB, matching 300 unique entities. This set has a higher number of NIL entities than did Bunescu and Pasca (2006) (10%) but lower than the TAC-KBP test set (43%).

All system development was done using a train (908 examples) and development (707 examples) split. The TAC-KBP and Microsoft News data sets were held out for final tests. A model trained on all 1615 examples was used for experiments.

7.1 TAC-KBP 2009 Experiments

The KB is derived from English Wikipedia pages that contained an infobox. Entries contain basic descriptions (article text) and attributes. The TAC-KBP query set contains 3904 entity mentions for 560 distinct entities; entity type was only provided for evaluation. The majority of queries were for organizations (69%). Most queries were missing from the KB (57%). 77% of the distinct GPEs in the queries were present in the KB, but for PERs and ORGs these percentages were significantly lower, 19% and 30% respectively.

Table 1 shows results on TAC-KBP data using all of our features as well a subset of features based on feature selection experiments on development data. We include scores for both micro-averaged accuracy – averaged over all queries – and macro-averaged accuracy – averaged over each unique entity – as well as the best and median reported results for these data (McNamee and Dang, 2009). We obtained the best reported results for macro-averaged accuracy, as well as the best results for NIL detection with micro-averaged accuracy, which shows the advantage of our approach to learning NIL. See McNamee et al. (2009) for additional experiments.

The candidate selection phase obtained a recall of 98.6%, similar to that of development data. Missed candidates included *Iron Lady*, which refers metaphorically to Yulia Tymoshenko, *PCC*, the Spanish-origin acronym for the Cuban Communist Party, and *Queen City*, a former nickname for the city of Seattle, Washington. The system returned a mean of 76 candidates per query, but the median was 15 and the maximum 2772 (*Texas*). In about 10% of cases there were four or fewer candidates and in 10% of cases there were more than 100 candidate KB nodes. We observed that ORGs were more difficult, due to the greater variation and complexity in their naming, and that they can be named after persons or locations.

7.2 Feature Effectiveness

We performed two feature analyses on the TAC-KBP data: an additive study – starting from a small baseline feature set used in candidate selection we add feature groups and measure performance changes (omitting feature combinations), and an ablative study – starting from all features, remove a feature group and measure performance.

[7] Data available from www.dredze.com

[8] http://en.wikipedia.org/wiki/Help:Infobox

Class	*All*	*non-NIL*	*NIL*
Baseline	0.7264	0.4621	0.9251
Acronyms	0.7316	0.4860	0.9161
NE Analysis	0.7661	0.7181	0.8022
Google	0.7597	0.7421	0.7730
Doc/KB Text Similarity	0.7313	0.6699	0.7775
Wikitology	0.7318	0.4549	0.9399
All	0.7984	0.7063	0.8677

Table 2: Additive analysis: micro-averaged accuracy.

	Num. Queries		Accuracy		
	Total	*Nil*	*All*	*non-NIL*	*NIL*
NIL	452	187	0.4137	0.0	1.0
GPE	132	20	0.9696	1.00	0.8000
ORG	115	45	0.8348	0.7286	1.00
PER	205	122	0.9951	0.9880	1.00
All	452	187	**0.9469**	0.9245	0.9786
Cucerzan (2007)			0.914	-	-

Table 3: Micro-average results for Microsoft data.

Table 2 shows the most significant features in the feature addition experiments. The baseline includes only features based on string similarity or aliases and is not effective at finding correct entries and strongly favors NIL predictions. Inclusion of features based on analysis of named-entities, popularity measures (e.g., Google rankings), and text comparisons provided the largest gains. The overall changes are fairly small, roughly ±1%; however changes in non-NIL precision are larger.

The ablation study showed considerable redundancy across feature groupings. In several cases, performance could have been slightly improved by removing features. Removing all feature combinations would have improved overall performance to 81.05% by gaining on non-NIL for a small decline on NIL detection.

7.3 Experiments on Microsoft News Data

We downloaded the evaluation data used in Cucerzan (2007)[9]: 20 news stories from MSNBC with 642 entity mentions manually linked to Wikipedia and another 113 mentions not having any corresponding link to Wikipedia.[10] A significant percentage of queries were not of type PER, ORG, or GPE (e.g., "Christmas"). SERIF assigned entity types and we removed 297 queries not recognized as entities (counts in Table 3).

We learned a new model on the training data above using a reduced feature set to increase speed.[11] Using our fast candidate selection system, we resolved each query in 1.98 seconds (median). Query processing time was proportional to the number of candidates considered. We selected a median of 13 candidates for PER, 12 for ORG and 102 for GPE. Accuracy results are in Table 3. The high results reported for this dataset over TAC-KBP is primarily because we perform very well in predicting popular and rare entries – both of which are common in newswire text.

One issue with our KB was that it was derived from infoboxes in Wikipedia's Oct 2008 version which has both new entities, [12] and is missing entities.[13] Therefore, we manually confirmed NIL answers and new answers for queries marked as NIL in the data. While an exact comparison is not possible (as described above), our results (94.7%) appear to be at least on par with Cucerzan's system (91.4% overall accuracy).With the strong results on TAC-KBP, we believe that this is strong confirmation of the effectiveness of our approach.

8 Conclusion

We presented a state of the art system to disambiguate entity mentions in text and link them to a knowledge base. Unlike previous approaches, our approach readily ports to KBs other than Wikipedia. We described several important challenges in the entity linking task including handling variations in entity names, ambiguity in entity mentions, and missing entities in the KB, and we showed how to each of these can be addressed. We described a comprehensive feature set to accomplish this task in a supervised setting. Importantly, our method discriminately learns when not to link with high accuracy. To spur further research in these areas we are releasing our entity linking system.

[9] http://research.microsoft.com/en-us/um/people/silviu/WebAssistant/TestData/

[10] One of the MSNBC news articles is no longer available so we used 759 total entities.

[11] We removed Google, FST and conjunction features which reduced system accuracy but increased performance.

[12] 2008 vs. 2006 version used in Cucerzan (2007) We could not get the 2006 version from the author or the Internet.

[13] Since our KB was derived from infoboxes, entities not having an infobox were left out.

References

Javier Artiles, Satoshi Sekine, and Julio Gonzalo. 2008. Web people search: results of the first evaluation and the plan for the second. In *WWW*.

Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Conference on Computational Linguistics (COLING)*.

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Association for Computational Linguistics*.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Management of Data*.

E. Boschee, R. Weischedel, and A. Zamanian. 2005. Automatic information extraction. In *Conference on Intelligence Analysis*.

Razvan C. Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *European Chapter of the Assocation for Computational Linguistics (EACL)*.

Peter Christen. 2006. A comparison of personal name matching: Techniques and practical issues. Technical Report TR-CS-06-02, Australian National University.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Markus Dreyer, Jason Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2009. Scaling Wikipedia-based named entity disambiguation to arbitrary web text. In *WikiAI09 Workshop at IJCAI 2009*.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Knowledge Discovery and Data Mining (KDD)*.

Martin Klein and Michael L. Nelson. 2008. A comparison of techniques for estimating IDF values to generate lexical signatures for the web. In *Workshop on Web Information and Data Management (WIDM)*.

Fangtao Li, Zhicheng Zhang, Fan Bu, Yang Tang, Xiaoyan Zhu, and Minlie Huang. 2009. THU QUANTA at TAC 2009 KBP and RTE track. In *Text Analysis Conference (TAC)*.

Gideon S. Mann and David Yarowsky. 2003. Unsupervised personal name disambiguation. In *Conference on Natural Language Learning (CONLL)*.

Andrew McCallum, Kamal Nigam, and Lyle Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Knowledge Discovery and Data Mining (KDD)*.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the TAC 2009 knowledge base population track. In *Text Analysis Conference (TAC)*.

Paul McNamee, Mark Dredze, Adam Gerber, Nikesh Garera, Tim Finin, James Mayfield, Christine Piatko, Delip Rao, David Yarowsky, and Markus Dreyer. 2009. HLTCOE approaches to knowledge base population at TAC 2009. In *Text Analysis Conference (TAC)*.

Marius Pasca. 2008. Turning web text and search queries into factual knowledge: hierarchical class attribute extraction. In *National Conference on Artificial Intelligence (AAAI)*.

Massimo Poesio, David Day, Ron Artstein, Jason Duncan, Vladimir Eidelman, Claudio Giuliano, Rob Hall, Janet Hitzeman, Alan Jern, Mijail Kabadjov, Stanley Yong, Wai Keong, Gideon Mann, Alessandro Moschitti, Simone Ponzetto, Jason Smith, Josef Steinberger, Michael Strube, Jian Su, Yannick Versley, Xiaofeng Yang, and Michael Wick. 2008. Exploiting lexical and encyclopedic resources for entity disambiguation: Final report. Technical report, JHU CLSP 2007 Summer Workshop.

Gerard Salton and Michael McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Conference on Natural Language Learning (CONLL)*.

Zareen Syed, Tim Finin, and Anupam Joshi. 2008. Wikipedia as an ontology for describing documents. In *Proceedings of the Second International Conference on Weblogs and Social Media*. AAAI Press.

A Discriminative Latent Variable-Based “DE” Classifier for Chinese–English SMT

Jinhua Du and **Andy Way**
CNGL, School of Computing
Dublin City University
{jdu, away}@computing.dcu.ie

Abstract

Syntactic reordering on the source-side is an effective way of handling word order differences. The 的 (DE) construction is a flexible and ubiquitous syntactic structure in Chinese which is a major source of error in translation quality. In this paper, we propose a new classifier model — discriminative latent variable model (DPLVM) — to classify the DE construction to improve the accuracy of the classification and hence the translation quality. We also propose a new feature which can automatically learn the reordering rules to a certain extent. The experimental results show that the MT systems using the data reordered by our proposed model outperform the baseline systems by 6.42% and 3.08% relative points in terms of the BLEU score on PB-SMT and hierarchical phrase-based MT respectively. In addition, we analyse the impact of DE annotation on word alignment and on the SMT phrase table.

1 Introduction

Syntactic structure-based reordering has been shown to be significantly helpful for handling word order issues in phrase-based machine translation (PB-SMT) (Xia and McCord, 2004; Collins et al., 2005; Wang et al., 2007; Li et al., 2007; Elming, 2008; Chang et al., 2009). It is well-known that in MT, it is difficult to translate between Chinese–English because of the different word orders (cf. the different orderings of head nouns and relative clauses). Wang et al. (2007) pointed out that Chinese differs from English in several important respects, such as relative clauses appearing before the noun being modified, prepositional phrases often appearing before the head they modify, etc. Chang et al. (2009) argued that many of the structural differences are related to the ubiquitous Chinese structural particle phrase 的 (DE) construction, used for a wide range of noun modification constructions (both single word and clausal) and other uses. They pointed out that DE is a major source of word order error when a Chinese sentence is translated into English due to the different ways that the DE construction can be translated.

In this paper, we focus on improving the classification accuracy of DE constructions in Chinese as well as investigating its impact on translation quality. From the grammatical perspective, the 的(DE) in Chinese represents the meaning of “noun modification” which generally is shown in the form of a Noun phrase (NP) [A DE B]. A includes all the words in the NP before DE and B contains all the words in the NP after DE. Wang et al. (2007) first introduced a reordering of the DE construction based on a set of rules which were generated manually and achieved significant improvements in translation quality. Chang et al. (2009) extended this work by classifying DE into 5 finer-grained categories using a log-linear classifier with rich features in order to achieve higher accuracy both in reordering and in lexical choice. Their experiments showed that a higher accuracy of the DE classification improved the accuracy of reordering component, and further indirectly improved the translation quality in terms of BLEU (Papineni et al., 2002) scores.

We regard the DE classification as a labeling task, and hence propose a new model to label the DE construction using a discriminative latent variable algorithm (DPLVM) (Morency et al., 2007; Sun and Tsujii, 2009), which uses latent variables to carry additional information that may not be expressed by those original labels and capture more complicated dependencies between DE and its corresponding features. We also propose a new feature defined as “tree-pattern” which can automatically learn the reordering rules rather than using manually generated ones.

The remainder of this paper is organised as follows. In section 2, we introduce the types of word order errors caused by the DE construction. Section 3 describes the closely related work on DE construction. In section 4, we detail our proposed DPLVM algorithm and its adaptation to our task. We also describe the feature templates as well as the proposed new feature used in our model. In section 5, the classification experiments are conducted to compare the proposed classification model with a log-linear model. Section 6 reports comparative experiments conducted on the NIST 2008 data set using two sets of reordered and non-reordered data. Meanwhile, in section 7, an analysis on how the syntactic DE reordering affects word alignment and phrase table is given. Section 8 concludes and gives avenues for future work.

2 The Problem of Chinese DE Construction Translation

Although syntactic reordering is an effective way of significantly improving translation quality, word order is still a major error source between Chinese and English translation. Take examples in Figure 1 as an illustration. The errors of three translation results in Figure 1 are from different MT systems, and many errors relate to incorrect reordering for the 的 (DE) structure.

These three translations are from different Hiero systems. Although Hiero has an inherent reordering capability, none of them correctly reordered “bad reputation” and “middle school” around the DE. Chang et al. (2009) suggested that this is because it is not sufficient to have a formalism which supports phrasal reordering. They claimed it is necessary to have sufficient linguistic modeling, so that the system knows when and how much to rearrange.

Source: 当地(local) 一所(a) 名声不佳(bad reputation)
的(with) 中学(middle school)
Reference: ’a local middle school with a bad reputation’
Team 1: ’a bad reputation of the local secondary school’
Team 2: ’the local a bad reputation secondary school’
Team 3: ’a local stigma secondary schools’

Figure 1: Examples of DE construction translation errors from (Chang et al., 2009)

Figure 2 gives an example illustrating how the reordering of DE construction influences the translation of a Chinese sentence. We can see that if we can properly recognise the DE construction [A DE B] and correctly perform the reordering, we can achieve a closer word order with English and hence a good English translation even it is literal.

Although the Hiero system has a strong reordering capability in its generalised phrases, it still cannot process some complicated and flexible cases of DE construction like those in Figure 1. Therefore, a lot of work has gone into word reordering before decoding so that the Chinese sentences have a closer word order with corresponding English sentences.

3 Related Work on DE Construction

To address the word order problems of the DE construction, Wang et al. (2007) proposed a syntactic reordering approach to deal with structural differences and to reorder source language sentences to be much closer to the order of target language sentences. They presented a set of manually generated syntactic rules to determine whether a 的(DE) construction should be reordered or not before translation, such as “For DNPs consisting of ‘XP+DEG’, reorder if XP is PP or LCP” etc. (cf. (Wang et al., 2007)). The deficiency of their algorithm is that they did not fully consider the flexibility of the DE construction, as it can be translated in many different ways.

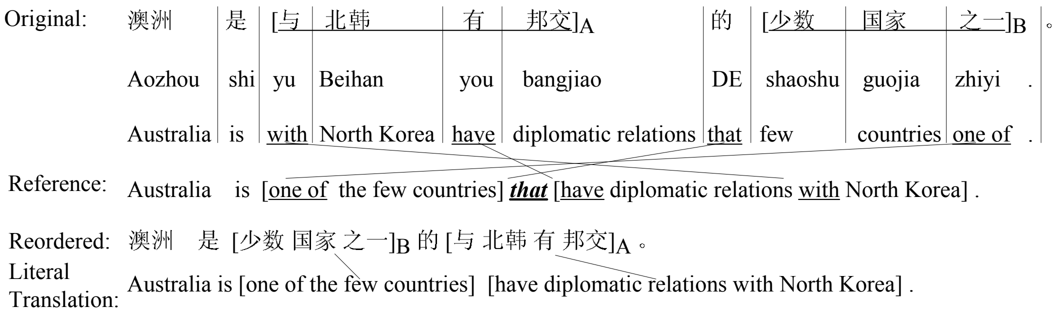

Original: 澳洲 是 [与 北韩 有 邦交]A 的 [少数 国家 之一]B 。
Aozhou shi yu Beihan you bangjiao DE shaoshu guojia zhiyi .
Australia is with North Korea have diplomatic relations that few countries one of .
Reference: Australia is [one of the few countries] that [have diplomatic relations with North Korea] .
Reordered: 澳洲 是 [少数 国家 之一]B 的 [与 北韩 有 邦交]A 。
Literal Translation: Australia is [one of the few countries] [have diplomatic relations with North Korea] .

Figure 2: An example of DE construction reordering (extended from the original figure in (Chiang, 2005))

Chang et al. (2009) extended the work of (Wang et al., 2007) and characterised the DE structures into 5 finer-grained classes based on their syntactic behaviour. They argued that one possible reason why the 的(DE) construction remains problematic is that previous work has paid insufficient attention to the many ways that the 的 (DE) construction can be translated, as well as the rich structural cues which exist for these translations.

For a Chinese noun phrase [A 的 B], it can be categorized into one of the following five classes (cf. (Chang et al., 2009) for some real examples of each class):

- A B (label: DE$_{AB}$)

  In this category, A on the Chinese side is translated as a pre-modifier of B. In most cases A is an adjectival form.

- B preposition A (label: DE$_{BprepA}$)

  There are several cases that are translated into the form B preposition A.

- A's B (label: DE$_{AsB}$)

  In this class, the English translation is an explicit s-genitive case. This class occurs much less often but is still interesting because of the difference from the of-genitive.

- relative clause (label: DE$_{relc}$)

  In this class, the relative clause would be introduced by a relative pronoun or be a reduced relative clause.

- A preposition B (label: DE$_{AprepB}$)

  This class is another small one. The English translations that fall into this class usually have some number, percentage or level word in the Chinese A.

Chang et al. (2009) used 6 kinds of features for DE classification, namely part-of-speech tag of DE (DEPOS), Chinese syntactic patterns appearing before DE (A-pattern), unigrams and bigrams of POS tags(POS-ngram), suffix unigram and bigram of word (Lexical), Semantic class of words (SemClass) and Re-occurrence of nouns (Topicality). A conditional log-linear classifier (Chang et al., 2009) is trained to classify each DE based on features extracted from the parsed data.

4 Discriminative Probabilistic Latent Variable Model

4.1 Motivation

Based on the discussion so far, we can see that:

- syntactic reordering of the DE construction in Chinese is an effective way to improve the translation quality;
- classifying the DE construction into finer-grained categories could achieve better reordering and translation performance;
- classification accuracy of the DE construction in Chinese has a significant impact on SMT performance.

Driven by these three points, especially the third one, we propose a DPLVM-based classifier to improve classification accuracy. In natural language processing (NLP) such as sequential labeling (Sun and Tsujii, 2009), DPLVM demonstrated excellent capability of learning latent dependencies of the specific problems, and have outperformed several commonly-used conventional models, such as support vector machines, conditional random fields and hidden Markov models.

4.2 DPLVM Algorithm

In this section, we theoretically introduce the definition and mathematical description of the DPLVM algorithm used in NLP tasks (Sun and Tsujii, 2009).

Given a sequence of observations $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$ and a sequence of labels $\mathbf{y} = \{y_1, y_2, \ldots, y_m\}$, the task is to learn a mapping between $\mathbf{x}$ and $\mathbf{y}$. y_i is a class label and is a member of a set $\mathbf{Y}$ of possible class labels. DPLVM also assumes a sequence of latent variables $\mathbf{h} = \{h_1, h_2, \ldots, h_m\}$, which is hidden in the training examples.

The DPLVM is defined as in (1) (Morency et al., 2007; Sun and Tsujii, 2009):

$$P(\mathbf{y}|\mathbf{x}, \Theta) = \sum_{\mathbf{h}} P(\mathbf{y}|\mathbf{h}, \mathbf{x}, \Theta) P(\mathbf{h}|\mathbf{x}, \Theta) \quad (1)$$

where Θ are the parameters of the model. It can be seen that the DPLVM equates to a CRF model if it has only one latent variable for each label.

For the sake of efficiency, the model is restricted to have disjoint sets of latent variables associated with each class label. Each h_j is a member in a set $\mathbf{H}_{y_j}$ of possible latent variables for the class label y_j. We define $\mathbf{H}$ as the union of all $\mathbf{H}_{y_j}$ sets, so sequences which have any $h_j \notin \mathbf{H}_{y_j}$ will by definition have $P(\mathbf{y}|\mathbf{x}, \Theta) = 0$, so that the model can be rewritten as in (2):

$$P(\mathbf{y}|\mathbf{x}, \Theta) = \sum_{\mathbf{h} \in \mathbf{H}_{y_1} \times \ldots \mathbf{H}_{ym}} P(\mathbf{h}|\mathbf{x}, \Theta) \quad (2)$$

where $P(\mathbf{h}|\mathbf{x}, \Theta)$ is defined by the usual conditional random field formulation, as in (3):

$$P(\mathbf{h}|\mathbf{x}, \Theta) = \frac{\exp \Theta \cdot \mathbf{f}(\mathbf{h}, \mathbf{x})}{\sum_{\forall \mathbf{h}} \exp \Theta \cdot \mathbf{f}(\mathbf{h}, \mathbf{x})} \quad (3)$$

in which $\mathbf{f}(\mathbf{h}, \mathbf{x})$ is a feature vector. Given a training set consisting of n labeled sequences (x_i, y_i), for $i = 1 \ldots n$, parameter estimation is performed by optimizing the objective function in (4):

$$L(\Theta) = \sum_{i=1}^{n} \log P(y_i|x_i, \Theta) - R(\Theta) \quad (4)$$

The first term of this equation is the conditional log-likelihood of the training data. The second term is a regularizer that is used for reducing overfitting in parameter estimation.

For decoding in the test stage, given a test sequence $\mathbf{x}$, we want to find the most probable label sequence y^*, as in (5):

$$\mathbf{y}^* = \arg\max_{y} P(\mathbf{y}|\mathbf{x}, \Theta^*) \quad (5)$$

Sun and Tsujii (2009) argued that for latent conditional models like DPLVMs, the best label path $\mathbf{y}^*$ cannot directly be generated by the Viterbi algorithm because of the incorporation of hidden states. They proposed a latent-dynamic inference (LDI) method based on A^* search and dynamic programming to efficiently decode the optimal label sequence $\mathbf{y}^*$. For more details of the LDI algorithm, refer to (Sun and Tsujii, 2009).

In our experiments, we use the open source toolkit of DPLVM[1] and adapt it to our special requirements based on the different features and scenarios.

4.3 Data and DE Annotation

We use the 5 classes of DE of (Chang et al., 2009) shown in Section 3 to label DE using our DPLVM model. In order to fairly compare the classification performance between that of Chang et al. (2009) and our proposed classifiers, we use the same data sets and conditions to train and test the classifier. The data sets are the Chinese Treebank 6.0 (LDC2007T36) and the English–Chinese Translation Treebank 1.0 (LDC2007T02). For more details about the data sets, refer to (Chang et al., 2009). There are 3523 DEs in the data set, with 543 of them in the "other" category which do not belong to any of the 5 pre-defined classes. In the classification experiments, the "other" class is excluded[2] and 2980 DEs remain, each of which

[1] http://www.ibis.t.u-tokyo.ac.jp/XuSun

[2] In the classification experiments of Chang et al. (2009), the "other" class was excluded, so in order to carry out a

is manually annotated with DE labels for the purpose of classifier training and evaluation.

In order to match the training and testing conditions, we used a parser trained on CTB6 excluding files 1-325 to parse the data sets with DE annotation and extract parse-related features rather than using gold-standard parses (same conditions as in (Chang et al., 2009)). It is worth noting that in the Chinese Treebank, there are two types of POS tag for DE in NPs, namely DEC and DEG. However, as a result of using a trained parser, the POS tags of DE might have other values than DEC and DEG. In our data set, there are four other POS tags, namely {AS, DER, DEV,SP}.

4.4 Labels and Features in DPLVM Model

In our task, we use the 5 class labels of DE constructions in NPs, namely DE$_{AB}$, DE$_{AprepB}$, DE$_{AsB}$, DE$_{BprepA}$, DE$_{relc}$.

Note that in the case of the DE construction in Chinese, it is different from traditional sequence labeling tasks such as POS tagging, parsing etc. We only need to label one word in the NP structure, i.e. the 的(DE) in a Chinese NP [A DE B]. Therefore the sequence labeling task becomes efficient and speedy using the DPLVM algorithm.

Based on our task, the mathematical conditions for DE classification in a sequence of [A DE B] are denoted as follows:

- **Sequence of Observations:**

  $\mathbf{x} = x_1, \ldots, x_l, x_{DE}, x_k, \ldots, x_m$, where A=$\{x_1, \ldots, x_l\}$, x_{DE} is the Chinese character 的 (DE), and B=$\{x_k, \ldots, x_m\}$;

- **Set of Labels:**

  $\mathbf{Y} = \{y_i | 1 \leq i \leq 5\}$, in which the five labels are DE$_{AB}$, DE$_{AprepB}$, DE$_{AsB}$, DE$_{BprepA}$, DE$_{relc}$.

- **Latent Variables:**

  $\mathbf{h} = h_1, h_2, \ldots, h_m$, where $m = 3$ in our task.

We employ five features as well in the DPLVM model, namely DEPOS, POS-gram, lexical features, SemClass as well as a new feature: tree-pattern, which is discussed below.

We did not add the sixth feature used in (Chang et al., 2009) – topicality – in our classifier because we do not consider it to be a very useful in a data set in which the sentences which are randomly stored. In such a corpus, the content between any adjacent sentences are irrelevant in many cases.

The new feature and the templates of all features used in our task are defined as:

DEPOS:
As mentioned in section 4.3, there are 6 kinds of POS tags of DE. Thus, the feature template is defined as in (5):

$$\mathbf{T}_{depos} = \{d_{DE} | d_{DE} \in \mathbf{DP}\}, \text{ where } \mathbf{DP} = \{\text{AS, DEC, DEG,DER,DEV,SP}\}. \quad (5)$$

Tree-pattern:
Chang (2009) used an A-pattern feature which is an indicator function that fires when some syntactic rules are satisfied, such as “A is ADJP if A+DE is a DNP with the form of ‘ADJP+DEG’”, etc. These rules are induced manually based on the grammatical phenomena at hand. Here we propose a more generalised feature defined as “tree-pattern” to automatically learn the reordering from the training data.

We consider all the sub-tree structures around DE without any word POS tags. For example, consider the parse structure (an example in (Chang et al., 2009)) in (6):

(NP (NP (NR 韩国)) (CP (IP (VP (ADVP (AD 最)) (VP (VA 大)))) (DEC 的)) (NP (NN 投资) (NN 对象国)))))) (6)

where the tree-pattern is “NP NP CP IP VP ADVP VP DEC NP”. We do not use the word POS tag (except DE) in this feature, such as NR, AD, VA, etc. The intention of this feature is to enable the classifier to automatically learn the structural rules around DE. Given that the position of DE in the parsing of [A DE B] is i, then the feature template is defined as in (7):

$$\mathbf{T}_{tree\_u} = \{t_{i-l}, \ldots, t_{i-1}, t_i, t_{i+1}, \ldots, t_{i+m}\} \quad (7)$$
$$\mathbf{T}_{tree\_b} = \{t_{i-l}t_{i-l+1}, \ldots, t_{i-1}t_i, t_it_{i+1}, \ldots, t_{i+m-1}t_{i+m}\}$$

where $\mathbf{T}_{tree\_u}$ is the sequence of unigrams in connection with DE and $\mathbf{T}_{tree\_b}$ is the sequence of bigrams related to DE; l and m are the window

fair comparison, we did so too. For the SMT experiments, however, we kept it.

sizes of A and B respectively. Generally, we use all the unigrams and bigrams in the parsing of A and B in our experiments. We argue that the important advantage of this feature is that it does not depend on manually generated rules, but instead of learns and generalises the reordering rules from the training data directly.

POS-gram:
The POS-ngram feature adds all unigrams and bigrams in A and B. Given that the position of DE is i in [A DE B], the feature template is defined as in (8):

$$\mathbf{T}_{pos\_u} = \{p_{i-l}, \ldots, p_{i-1}, p_{i+1}, \ldots, p_{i+m}\}$$
$$\mathbf{T}_{pos\_b} = \{p_{i-l}p_{i-l+1}, \ldots, \mathbf{p}_{i-1}\mathbf{p}_{i+1}, \ldots, p_{i+m-1}p_{i+m}\} \quad (8)$$

where $\mathbf{T}_{pos\_u}$ and $\mathbf{T}_{pos\_b}$ are uigrams and bigrams in A and B. In the unigrams, we exclude the POS of DE; in the bigrams, we include a bigram pair across DE.

Some other features such as lexical features, SemClass (cf. (Chang et al., 2009) for details) can be defined using similar feature template.

5 Experiments on DPLVM DE Classifier

In this section, we compare the performance of DE classifiers between the DPLVM and log-linear methods.

The accuracy of classification is defined as in (9):

$$\frac{\text{number of correctly labeled DEs}}{\text{number of all DEs}} \times 100 \quad (9)$$

Phrase Type	Log-linear		DPLVM	
	5-A	2-A	5-A	2-A
DEPOS	54.8	71.0	**56.2**	**72.3**
+A-pattern	67.9	83.7	-	-
+Tree-pattern	-	-	**69.6**	**85.2**
+POS-gram	72.1	84.9	**73.6**	**86.5**
+Lexical	74.9	86.5	**76.4**	**87.9**
+SemClass	75.1	86.7	**76.8**	**88.3**
+Topicality	75.4	86.9	-	-

Table 1: Comparison between the two classifiers on 5-class and 2-class accuracy

Table 1 shows the comparison of accuracy, where "5-A" and "2-A" represent the accuracy of the 5-class and 2-class respectively. The 2-class is the categorised classes of DE in (Wang et al., 2007) which are defined as "reordered" and "non-reordered" categories. It can be seen that our DPLVM classifier outperforms the log-linear classifier by 1.4 absolute (1.86% and 1.61% relative respectively) points both on 5-class and 2-class classifications. Furthermore, we see that the DPLVM achieves significantly better performance than the log-linear model only with the simple feature of "DEPOS". As to the new feature "tree-pattern", we can see that it achieves the improvement of 1.5% compared to the "A-pattern" in terms of the accuracy of "2-A". This improvement attributes to the good learning ability of DPLVM as well as the strong generalisation capability of the tree-pattern feature.

In terms of speed, in our task we only need to label the Chinese character DE in the NP structure [A DE B] rather than label the whole sentence, so that we have a feature matrix of $n \times 1$ for each DE. Accordingly, the DPLVM classifier can run efficiently with low memory usage.

6 Experiments on SMT

6.1 Experimental Setting

For our SMT experiments, we used two systems, namely Moses (Koehn et al., 2007) and Moses-chart. The former is the state-of-the-art PB-SMT system while the latter is a new extended system of the Moses toolkit re-implementing the hierarchical PB-SMT (HPB) model (Chiang, 2005). The alignment is carried out by GIZA++ (Och and Ney, 2003) and then we symmetrized the word alignment using the grow-diag-final heuristic. Parameter tuning is performed using Minimum Error Rate Training (Och, 2003).

The training data contains 2,159,232 sentence pairs.The 5-gram language model is trained on the English part of the parallel training data. The development set (devset) is the NIST MT2006 test set and the test set is the NIST MT2008 "current" test set. All the results are reported in terms of BLEU (Papineni et al., 2002) and METEOR (MTR) (Banerjee and Lavie, 2005) scores.

To run the DE classifiers, we use the Stanford Chinese parser (Levy and Manning, 2003) to parse the Chinese side of the MT training data, the devset and test set.

6.2 Statistics of 5-class DE Annotation

For the DE-annotated MT experiments, after we parse the training data, the devset and the test set, we separately use the two DE classifiers to annotate the DE constructions in NPs in all of the parsed data. Once the DE data are labeled, we pre-process the Chinese data by reordering the sentences only with 的$_{BprepA}$ and 的$_{relc}$ annotations. Table 2 lists the statistics of the DE classes in the MT training data, devset and test set using our DPLVM classifier. "的$_{non}$" denotes the unlabeled 的(DE) which does not belong to any of the 5 classes.

6.3 Experimental Results

The experimental results from the PB-SMT and HPB systems separately using the DPLVM and log-linear classifiers are shown in Table 3.

	PB-SMT			Moses-chart		
	BL	LL	LV	BL	LL	LV
BLEU	22.42	23.47	**23.86**	24.36	24.75	**25.11**
MTR	52.03	53.25	**53.78**	53.37	53.75	**54.21**

Table 3: Experimental results on PB-SMT and Moses-chart. "BL" are the baselines; "LL" indicates the log-linear model-based system; "LV" is our DPLVM method.

The baseline systems indicate that the data is neither categorised into DE classes nor reordered on the Chinese side. We can see that (1) the "LV" method outperformed the "BL" and "LL" by 1.44 absolute (6.42% relative), 0.39 absolute (1.66% relative) BLEU points for PB-SMT, and by 0.75 absolute (3.08% relative), 0.36 absolute (1.45% relative) BLEU points for Moses-chart; (2) the "LV" method achieved the improvements for PB-SMT and Moses-chart in terms of MTR scores compared to the "BL" and "LL" systems. Therefore, using DE classification and reordering on the source-side is helpful in improving translation quality; (3) the results using DPLVM achieve better translation quality than that of the "LL" processed data in terms of BLEU and METEOR (Banerjee and Lavie, 2005) scores, which indirectly shows that DPLVM outperforms the log-linear classification model; and (4) the improvements on both PB-SMT and Moses-chart show that the effectiveness of DE reordering is consistent for different types of MT systems. The results are verified by significance test on 95% confidence interval (Zhang and Vogel, 2004).[3]

7 Analysis

In this section, we plan to evaluate how DE reordering contributes to the improvement of translation quality in two respects, namely word alignment and phrase table.

7.1 Evaluating the Word Alignment

We create a word alignment test set which includes 500 sentences with human alignment annotation, and then add this test set into the MT training corpus. Accordingly, the DE-reordered test set is added into the reordered training corpus as well. Thus, we run GIZA++ using the same configurations for these two sets of data and symmetrize the bidirectional word alignment using grow-diag heuristic. The word alignment of the test set is evaluated with the human annotation using Precision, Recall, F1 and AER measures. The results are reported in Table 4.

	P	R	F1	AER
non-reordered	71.67	62.02	66.49	33.44
reordered	74.02	62.79	67.95	31.98
Gain	**2.35**	**0.77**	**1.46**	**-1.46**

Table 4: Comparison of Precision, Recall, F1 and AER scores of evaluating word alignment on original and reordered data

We can see that in terms of the four measures, the word alignment produced by the reordered data is slightly better than that of the original data. In some sense, we might say that the DE reordering is helpful in improving the word alignment of the training data.

7.2 Evaluating the Phrase Table

Wang et al. (2007) proposed one way to indirectly evaluate the phrase table by giving the same type of input to the baseline and reordered systems,

[3] http://projectile.sv.cmu.edu/research/public/tools/bootStrap/tutorial.htm.

	training		devset		testset	
DE-class	count	percent (%)	count	percent (%)	count	percent (%)
的$_{AB}$	312,679	23.08	523	25.80	453	28.78
的$_{AprepB}$	6,975	0.51	9	0.44	7	0.44
的$_{AsB}$	13,205	0.97	23	1.13	14	0.89
的$_{BprepA}$	658,589	47.31	956	48.05	688	43.71
的$_{relc}$	316,772	23.38	419	20.67	341	21.66
的$_{non}$	46,547	3.44	97	4.79	71	4.51
Total 的	1,354,767	100	2027	100	1574	100

Table 2: The number of different DE classes labeled for training data, devset and testset using the DPLVM classifier

with the consideration that if the reordered system learned a better phrase table, then it may outperform the baseline on non-reordered inputs despite the mismatch and vice versa. However, they did not settle the question as to whether the reordered system can learn better phrase tables.

We also try to use the idea of Wang et al (2007) to carry out the phrase table evaluation on PB-SMT,[4] i.e. we tune the baseline on a reordered devset and then evaluate on a reordered test set; tune the reordered system on a non-reordered devset and then evaluate on a non-reordered test set. The results are shown in Table 5.

Testset	baseline	reordered	
		LL	DPLVM
non-reordered set	22.42	22.76	22.85
reordered set	23.36	23.47	23.86

Table 5: Comparison of BLEU scores in matched and mismatched conditions on PB-SMT.

We find that (1) given the non-reordered test set, the DE reordered system performs better than the baseline system, which is consistent when different DE classifiers are applied; (2) given the reordered test set system, the reordered set produces a better result than the baseline, which is also consistent when different DE classifiers are applied; and (3) the results from the DPLVM-based reordered data are better than those from the LL-based reordered data. From the comparison, one might say that the reordered system was learned a better phrase table and the reordered test set addresses the problem of word order.

To sum up, from the SMT results and the evaluation results on the word alignment and the phrase table, we can conclude that the DE reordering methods contribute significantly to the improvements in translation quality, and it also implies that using DE reordered data can achieve better word alignment and phrase tables.

8 Conclusions and Future Work

In this paper, we presented a new classifier: a DPLVM model to classify the Chinese 的(DE) constructions in NPs into 5 classes. We also proposed a new and effective feature – tree-pattern – to automatically learn the reordering rules using the DPLVM algorithm. The experimental results showed that our DPLVM classifier outperformed the log-linear model in terms of both the classification accuracy and MT translation quality. In addition, the evaluation of the experimental results in section 7 indicates that the DE-reordering approach is helpful in improving the accuracy of the word alignment, and can also produce better phrase pairs and thus generate better translations.

As for future work, firstly we plan to examine and classify the DE constructions in other syntactic structures such as VP, LCP etc. Secondly, we plan to apply the DE-annotated approach in a syntax-based MT system (Zollmann and Venugopal, 2006) and examine the effects. We also intend to improve the classification accuracy of the DE classifier with richer features to further improve translation quality.

[4] The phrases in HPB systems are different from those in PB-SMT because they are variable-based, so we evaluate the hierarchical phrases in (Du and Way, 2010)

Acknowledgment

Many thanks to Dr. Pi-Chuan Chang for providing the source code of her DE classifier and manually DE-annotated training data as well as valuable instruction in their use. Thanks also to Dr. Xu Sun for the source code of his Latent Variable classifier together with help in their use. This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Dublin City University.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, ACL-2005*, pages 65–72.

Pi-Chuan Chang, Dan Jurafsky and Christopher D. Manning. 2009 Disambiguating “DE” for Chinese-English machine translation. In *Proceedings of the Fourth Workshop on SMT*, pages 215–223.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL’05*, pages 263–270.

Michael Collins, Philipp Koehn, and Ivona Kucerova. newblock 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL05*, pages 531–540.

Jinhua Du and Andy Way. 2010. The impact of source-side syntactic reordering on hierarchical phrase-based SMT. In *Proceedings of the 14th Annual conference of the European Association for Machine Translation*, Saint-Raphael, France.

Jakob Elming. 2008. Syntactic reordering integrated with phrase-based SMT. In *Proceedings of ACL-08 SSST-2*, pages 46–54.

Philipp Koehn, Hieu Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, Wade Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *demonstration session of ACL’07*, pages 177–180.

Roger Levy and Christopher Manning. 2003. Is it harder to parse Chinese, or the Chinese treebank? In *Proceedings of ACL’03*, pages 439–446.

Chi-Ho Li, Dongdong Zhang, Mu Li, Ming Zhou, Minghui Li and Yi Guan. 2007. A probabilistic approach to syntax-based reordering for statistical machine translation. In *proceedings of the ACL’07*, pages 720–727.

Louis-Philippe Morency, Ariadna Quattoni and Trevor Darrell. 2007. Latent-dynamic Discriminative Models for Continuous Gesture Recognition. In *proceedings of CVPR’07*, pages 1–8.

Franz Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of ACL’03*, pages 160–167.

Franz Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the ACL-02*, pages 311–318.

Xu Sun and Jun’ichi Tsujii. 2009. Sequential Labeling with Latent Variables: An Exact Inference Algorithm and An Efficient Approximation. In *Proceedings of The European Chapter of the Association for Computational Linguistics (EACL’09)*, pages 772-780.

Chao Wang, Michael Collins, and Philipp Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proceedings of EMNLP-CoNLL*, pages 737–745.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of Coling 2004*, pages 508–514.

Ying Zhang and Stephan Vogel. 2004. Measuring Confidence Intervals for the Machine Translation Evaluation Metrics. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, pages 85–94.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of HLT-NAACL 2006: Proceedings of the Workshop on Statistical Machine Translation*, New York, pages 138–141.

An Empirical Study on Learning to Rank of Tweets

[1]Yajuan Duan* [2]Long Jiang [2]Tao Qin [2]Ming Zhou [2]Heung-Yeung Shum
[1]Department of Computer Science and Technology
University of Science and Technology of China
[2]Microsoft Research Asia
{v-yaduan,longj,taoqin,mingzhou,hshum}@microsoft.com

Abstract

Twitter, as one of the most popular micro-blogging services, provides large quantities of fresh information including real-time news, comments, conversation, pointless babble and advertisements. Twitter presents tweets in chronological order. Recently, Twitter introduced a new ranking strategy that considers popularity of tweets in terms of number of retweets. This ranking method, however, has not taken into account content relevance or the twitter account. Therefore a large amount of pointless tweets inevitably flood the relevant tweets. This paper proposes a new ranking strategy which uses not only the content relevance of a tweet, but also the account authority and tweet-specific features such as whether a URL link is included in the tweet. We employ *learning to rank* algorithms to determine the best set of features with a series of experiments. It is demonstrated that whether a tweet contains URL or not, length of tweet and account authority are the best conjunction.

1 Introduction

Twitter provides a platform to allow users to post text messages known as tweets to update their followers with their findings, thinking and comments on some topics (Java et al., 2007). The searched tweets are presented by Twitter in chronological order except the first three, which are ranked by considering popularity of tweets in terms of the number of retweets.

This ranking method, however, has not taken into account the content relevance and twitter account; inevitably, a large amount of pointless tweets (Pear Analytics, 2009) may flood the relevant tweets. Although this ranking method can provide fresh information to tweet users, users frequently expect to search relevant tweets to the search queries. For example, consider someone researching consumer responses toward the iPad. He or she would like to find tweets with appropriate comments such as *iPad is great* or *you can find many useful features of iPad,* rather than tweets with irrelevant comment, even if they are most recent or popular.

Moreover, neither Twitter's current chronological order based ranking nor the recently introduced popularity based ranking can avoid spam. A developer can accumulate hundreds of thousands of followers in a day or so. At the same time, it is not difficult for spammers to create large quantities of retweets. By contrast, content relevance ranking can effectively prevent spammers from cheating. Different from ranking tweets through chronological order and popularity, a content relevance strategy considers many characteristics of a tweet to determine its ranking level. Thus it is difficult for spammers to break the ranking system by simple methods such as increasing retweet count or number of followers.

In this paper, we propose a method to rank the tweets which outputs the matched tweets based on their content relevance to the query. We

* The work was done when the first author was intern at Microsoft Research Asia

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 295–303,
Beijing, August 2010

investigate the effects of content features and non-content features and produce a ranking system by a *learning to rank* approach.

With a series of experiments, we determined the best set of features and analyzed the effects of each of individual feature. We provide empirical evidence supporting the following claims,

- Account authority, length of tweet and whether a tweet contains a URL are the top three effective features for tweet ranking, where containing a URL is the most effective feature.
- We find an effective representation of account authority: the number of times the author was listed by other users. We find through experiments that this representation is better than the widely adopted number of followers.

2 Related Work

2.1 Real-time Search

At present, a number of web sites offer the so-called real-time search service which mainly returns real-time posts or shared links, videos and images obtained from micro-blogging systems or other medium according to the user's query. We investigate the ranking method used by these web sites. From their self-introduction page, we find four main criteria for ranking real-time posts. They are posting time, account authority, topic popularity and content relevance.

Specifically, Twitter maintains a specialized search engine which ranks tweets according to posting time and topic popularity. In addition, Google, Twazzup[2] and Chirrps[3] rank real-time tweets by posting time. While the last one also ranks tweets by popularity, which is measured by retweet count.

Tweefind[4] ranks search result according to authority of authors which depends on how popular, relevant, and active the author is. Additionally, Twitority[5] rank tweets by author authority as well.

Bing and CrowdEye[6] rank tweets by posting time or content relevance. Bing takes authors authority, retweet count and freshness into consideration while measuring the relevance. To determine the relevance of a tweet, CrowdEye considers a number of factors including content relevance and author influence which appears to rely heavily on the number of followers an author has. It turns out that the number of followers is not a very reasonable measure of the influence of an account according to our experimental results.

2.2 Twitter Recommendation

Besides tweet search, recently some researchers have focused on twitter recommendation system.

Chen et al. (2010) presented an approach to recommend URLs on Twitter as a means to better direct user attention in information streams. They designed the recommender taking three separate dimensions into consideration: content source, topic interest and social voting.

Sun et al. (2009) proposed a diffusion-based micro-blogging recommendation framework aiming to recommend micro-blogs during critical events via optimizing story coverage, reading effort and delay time of a story. The key point of this method is to construct an exact diffusion graph for micro-blogging, which is difficult due to the presence of extensive irrelevant personal messages and spam.

2.3 Blog Search and Forum Search

Another related topic is blog search and forum search. Recently, many approaches for blog search and forum search have been developed, which include *learning to rank* methods and link-based method.

Learning to rank approach

Xi et al. (2004) used features from the thread trees of forums, authors, and lexical distribution within a message thread and then applied Linear Regression and Support Vector Machine (SVM) to train the ranking function. Fujimura et al. (2005) exploited provisioning link and evaluation link between bloggers and blog entries, and scored each blog entry by weighting the hub and authority scores of the bloggers.

Link-Based approach

[2] Twazzup: http://www.twazzup.com/

[3] Chirrps: http://chirrps.com/

[4] Tweefind: http://www.tweefind.com/

[5] Twitority: http://www.twitority.com/

[6] CrowdEye: http://www.crowdeye.com/

Kritikopoulos et al. (2006) introduced similarities among bloggers and blogs into blog ranking. This method enabled the assignment of a higher score to the blog entry published by a blogger who has already accepted a lot of attention. Xu and Ma (2006) built a topic hierarchy structure through content similarity. Liu et al. (2007) presented a newsgroup structure-based approach PostRank which built posting trees according to response relationship between postings.

Chen et al. (2008) proposed a posting rank algorithm which built link graphs according to co-replier relationships. This kind of method exploits different types of structures among postings and improved the performance of traditional link-based ranking algorithm for forum search. However, it is difficult to rank postings which only have a few words simply based on content by using FGRank algorithm. And PostingRank approach relies too much on reply relations which are more likely to suffer from topic excursion.

Although approaches proposed above perform effectively in forum search and blog search, they are not appropriate for twitter search because tweets are usually shorter and more informal than blogs. Furthermore, it does not have the explicit hierarchy structure of newsgroup messages on forums. In addition, tweets possess many particular characteristics that blog and forum do not have.

3 Overview of Our Approach

To generate a good ranking function which provides relevant search results and prevents spammers' cheating activities, we analyze both content features and authority features of tweets and determine effective features. We adopt *learning to rank* algorithms which have demonstrated excellent power in addressing various ranking problems of search engines.

3.1 Learning to Rank Framework

Learning to Rank is a data-driven approach which integrates a bag of features in the model effectively. Figure 1 shows the paradigm of learning for tweet ranking.

At the first step, we prepare the training and test corpus as described in Section 5. Then we extract features from the training corpus. RankSVM algorithm (Joachims Thorsten, 1999) is used to train a ranking model from the training corpus. Finally, the model is evaluated by the test corpus.

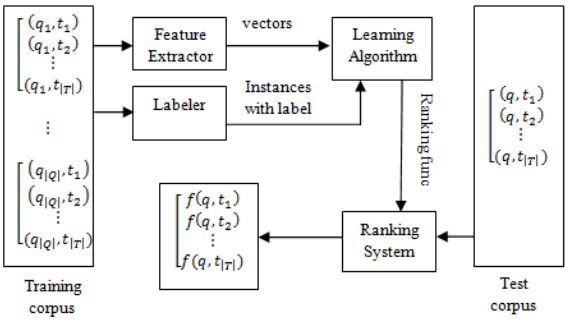


Figure 1. General Paradigm of Learning for Tweets Ranking

3.2 Features for Tweets Ranking

One of the most important tasks of a *learning to rank* system is the selection of a feature set. We exploit three types of features for tweet ranking.

1) *Content relevance features* refer to those features which describe the content relevance between queries and tweets.
2) *Twitter specific features* refer to those features which represent the particular characteristics of tweets, such as retweet count and URLs shared in tweet.
3) *Account authority features* refer to those features which represent the influence of authors of the tweets in Twitter (Leavitt et al., 2009).

In the next section, we will describe these three types of features in detail.

4 Feature Description

4.1 Content Relevance Features

We used three content relevance features, Okapi BM25 (Robertson et al., 1998), similarity of contents and length of tweet.

Okapi BM25 score measures the content relevance between query Q and tweet T. The standard BM25 weighting function is:

$$\mathrm{BM25}(\mathrm{T},\mathrm{Q}) = \sum_{\mathrm{q_i} \in \mathrm{Q}} \frac{\mathrm{IDF}(\mathrm{q_i}) \cdot \mathrm{tf}(\mathrm{q_i},\mathrm{T}) \cdot (\mathrm{k_1}+1)}{\mathrm{tf}(\mathrm{q_i},\mathrm{T}) + \mathrm{k_1}\left(1 - \mathrm{b} + \mathrm{b}\frac{\mathrm{Length(T)}}{\mathrm{avg_{length}}}\right)} \quad (1)$$

where Length(T) denotes the length of T and avg_{length} represents average length of tweet in corpus. IDF(q_i) is Inverse Document Frequency.

Similarity of contents estimates the popularity of documents in the corpus (Song et al., 2008). In our case, it measures how many tweets of the query are similar in content with the current tweet. We calculate a cosine similarity score for every pair of tweets, and the final similarity score for tweet T_i in T_{Q_k} is computed by the following formula:

$$\mathrm{Similarity}(T_i) = \frac{1}{|T_{Q_k}|-1} \sum_{T_j \epsilon T_{Q_k}, j \neq i} \frac{TV_i \cdot TV_j}{|TV_i| \cdot |TV_j|} \quad (2)$$

Where TV_i represents the TFIDF vector of T_i and T_{Q_k} refers to tweets collection of query Q_k.

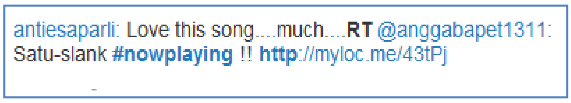
Length is measured by the number of words that a tweet contains. Intuitively, a long sentence is apt to contain more information than a short one. We use length of tweet as a measure of the information richness of a tweet.

4.2 Twitter's Specific Features

Tweets have many special characteristics. We exploit these characteristics and extract six twitter specific features as listed in Table 1.

Feature	Description
URL	Whether the tweet contains a URL
URL Count	Frequency of URLs in corpus
Retweet Count	How many times has this tweet been retweeted
Hash tag Score	Sum of frequencies of the top-n hash tags appeared in the tweet
Reply	Is the current tweet a reply tweet
OOV	Ratio of words out of vocabulary

Table 1. Twitter Specific Features



antiesaparli: Love this song....much....**RT** @anggabapet1311: Satu-slank **#nowplaying** !! **http**://myloc.me/43tPj

Figure 2. A Tweet Example

URL & URL Count: Twitter allows users to include URL as a supplement in their tweets. The tweet in Figure 2 contains URL http://myloc.me/43tPj which leads to a map indicating where the publisher located.

URL is a binary feature. It is assigned 1 when a tweet contains at least one URL, otherwise 0.

URL Count estimates the number of times that the URL appears in the tweet corpus.

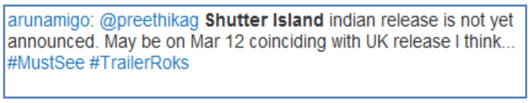
Retweet Count: Twitter users can forward a tweet to his or her followers with or without modification on the forwarded tweets, which is called retweet on Twitter. A retweeted tweet usually includes an *RT* tag. Generally, sentences before *RT* are comments of the retweeter and sentences after *RT* are the original content, perhaps with some modifications. Here we only consider tweets including *RT* with the original content unmodified. Retweet count is defined as the number of times a tweet is retweeted. In Figure 2, original tweet *Satu-slank #nowplaying !! http://myloc.me/43tPj* is retweeted once.

Hash Tag Score: Publishers are allowed to insert hash tags into their tweets to indicate the topic. In Figure 2, *#nowplaying* is a hash tag. We collect hash tags appearing in the tweets of every query and sort them in descending order according to frequency. Tag frequency for tweet T_i of query Q_k is computed from normalized frequency of top-n tags.

$$\mathrm{TagScore}(T_i) = \frac{1}{z_k} \sum_{\substack{j=1, tag_j \epsilon Tag_{Q_k} \\ tag_j \in T_i}}^{n} \mathrm{freq}(tag_j) \quad (3)$$

Where z_k is the normalization factor. freq(tag_j) represents the frequent of tag_j in corpus. And Tag_{Q_k} denotes the tag collection extracted from T_{Q_k}.

Reply: This is a binary feature. It is 1 when the tweet is a reply and 0 otherwise. A tweet starting with a twitter account is regarded as a reply tweet in our experiment. Figure 3 shows an example.



arunamigo: @preethikag **Shutter Island** indian release is not yet announced. May be on Mar 12 coinciding with UK release I think... #MustSee #TrailerRoks

Figure 3. Reply Tweet

OOV: This feature is used to roughly approximate the language quality of tweets. Words out of vocabulary in Twitter include spelling errors and named entities. According to a small-scale investigation, spelling errors account for more than 90% of OOVs excluding capitalized words, tags, mentions of users and

URLs. We use a dictionary with 0.5 million entries to compute the ratio of OOVs in a tweet.

$$\text{Quality(T)} = \frac{\text{\# of OOVs in T}}{\text{Length(T)}} \quad (4)$$

4.3 Account Authority Features

There are three important relations between users in Twitter: follow, retweet, and mention. Additionally, users are allowed to classify their followings into several lists based on topics. We measured the influence of users' authorities on tweets based on the following assumptions:

- Users who have more followers and have been mentioned in more tweets, listed in more lists and retweeted by more important users are thought to be more authoritative.
- A tweet is more likely to be an informative tweet rather than pointless babble if it is posted or retweeted by authoritative users.

```
PageRank algorithm for calculating popularity score for users.
Input: Directed Graph G of retweet relationship
       Damping factor e.
Output: popularity score for each user
Procedure:
Step 1: popularity score of all users are initialized as $1 - e$.
Step 2: update the popularity score for users.
    $PScore_{t+1}(v_i) = 1 - e + e \cdot \sum_{v_j \in R_{v_i}} \frac{PScore_t(v_j) RN_{ij}}{N_j}$
    $R_{v_i}$ denotes the collection of users who retweeted $v_i$'s tweet.
    $RN_{ij}$ is the number of times $v_i$ has been retweeted by $v_j$.
    $N_j$ is the number of users whose tweets $v_j$ has retweeted.
Step 3: Repeat the second step until all popularity scores will never change.
```

Figure 4. PageRank Algorithm for Calculating Popularity Score for Users

In order to distinguish the effect of the three relations, we computed four scores for each user representing the authority independently.

- Follower Score: number of followers a user has.
- Mention Score: number of times a user is referred to in tweets.
- List Score: number of lists a user appears in.
- Popularity Score: computed by PageRank algorithm (Page et al., 1999) based on retweet relations.

Following the retweet relationship among users, we construct a directed graph G (V, E). In our experiments, G is built from a tweet collection including about 1.1 million tweets. V denotes twitter users that appear in training examples. E is a set of directed edges. If author v_i published the tweet t_k, and author v_j retweeted t_k after v_i, there exists an edge from v_j to v_i. We call v_i original author and v_j retweeter. Figure 4 shows the PageRank algorithm for calculating popularity scores for twitter users. In our experiment, damping factor e was set to 0.8. Like Dong et al. (2010) did, we define three subtypes for each account authority score. Table 2 presents features of account authority we use.

Feature	Description
Sum_follower	Sum of follower scores of users who published or retweeted the tweet
Sum_popularity	Sum of popularity scores of users who published or retweeted the tweet
Sum_mention	Sum of mention scores of users who published or retweeted the tweet
Sum_list	Sum of list scores of users who published or retweeted the tweet
First_follower	Follower score of the user who published the tweet
First_popularity	Popularity score of the user who published the tweet
First_mention	Mention score of the user who published the tweet
First_list	List score of the user who published the tweet
Important_follower	The highest follower score of the user who published or retweeted the tweet
Important_popularity	The highest popularity score of the user who published or retweeted the tweet
Important_mention	The highest mention score of the user who published or retweeted the tweet
Important_list	The highest list score of the user who published or retweeted the tweet

Table 2. Account Authority Features for tweet

5 Experiment Data and Evaluation

We introduce the data we used in experiment and the evaluation metrics in this section.

5.1 Data

We analyze 140 hot searches on CrowdEye within a week. They consist of big events, famous persons, new products, festivals, movies and so on. The most frequent types of hot searches, which account for more than 81% of all hot searches, are as follows:

- News: news about public figures and news related to some places.
- Products: character description, promotion information and comments about products.
- Entertainment: mainly about movies, including film reviews and introductions about plots.

We select 20 query terms as shown in Table 3, including 5 persons, 5 locations, 5 products and 5 movie names. Specifically, Locations are sampled from a list of American cities. Person names come from the hot search and hot trends provided by Twitter and CrowdEye. Products are sampled from the popular searches of 35 product categories on eBay. And movies are selected from a collection of recommended movies from 2005 to 2010. We crawl 162,626 English tweets for the selected queries between March 25, 2010 and April 2, 2010 from Twitter Search. After removing the repeated ones, 159,298 tweets remained.

Query type	Query terms
Locations	New York, Nashville, Denver, Raleigh, Lufkin
Person Names	Obama, Bill Clinton, James Cameron, Sandra Bullock, LeBron James
products	Corvette, iPad, Barbie, Harry Potter, Windows 7
Movies	The Dark Knight, up in the air, the hurt locker, Batman Begins, Wall E

Table 3. 20 Query Terms

Retweets are forwardings of corresponding original tweets, sometimes with comments of retweeters. They are supposed to contain no more information than the original tweets, therefore they drops out of ranking in this paper.

We sample 500 tweets for each query from its original tweets collection and ask a human editor to label them with a relevance grade. In order to ensure the annotation is reasonable, we set multiple search intentions for each query referring to the topics arising in the tweets about the query in the corpus. Specifically, for Locations, tweets describing news related to the location are relevant. For people, what they have done and the comments about them are regarded as relevant information. For products, tweets including feature description, promotion and comments are considered relevant. And for movies, tweets about comment on the movies, show time and tickets information are preferred. We apply four judgment grades on query-tweet pairs: excellent, good, fair and bad. According to the statistics, about half of the tweets in the experiment data are labeled as bad. Table 4 presents the distribution for all grades.

Grade	Excellent	Good	Fair	Bad
Percentage	20.9%	10.9%	16.9%	51.3%
Min	2.4%	1.8%	4.0%	8.0%
Max	69.8%	23.2%	54.4%	81.0%

Table 4. Tweet Distribution of Each Grade

5.2 Evaluation Metrics

There are several metrics that are often used to measure the quality of rankings. In this paper, we use Normalized Discount Cumulative Gain (NDCG) which can handle multiple levels of relevance as the evaluation metrics (Jarvelin and Kekalainen, 2002).

6 Results

Five-fold cross-validation was used in our experiments. We choose tweets of sixteen queries (four from each query type) as the training data. The remaining tweets are divided into evaluation data and validation data equally.

6.1 Learning to Rank for Tweet Ranking

We learn a ranking model by using a RankSVM algorithm based on all features we extracted, which is denoted as RankSVM_Full. In the experiment, a toolkit named svm^{struct} [7] implemented by Thorsten Joachims is used. Figure 5 shows the comparison between our method which integrates three types of features and ranking through chronological order, account authority, and content relevance individually.

In this experiment, Content Relevance is measured by BM25 score. And Account

[7] SVM^{struct}: http://svmlight.joachims.org/svm_struct.html

Authority is approximated by the number of followers of the user. Figure 5 illustrates that ranking through content relevance is not as effective as other methods. This is because our work is essentially re-ranking on the result of Twitter Search. Hence almost all tweets include the query term which makes it difficult to distinguish them by BM25 score. Figure 5 also reveals that account authority is useful for ranking tweet relevance; it outperforms ranking through chronological order and is competitive to our model trained from all features. This agrees with the assumption we made about the influence of user authorities on tweets.

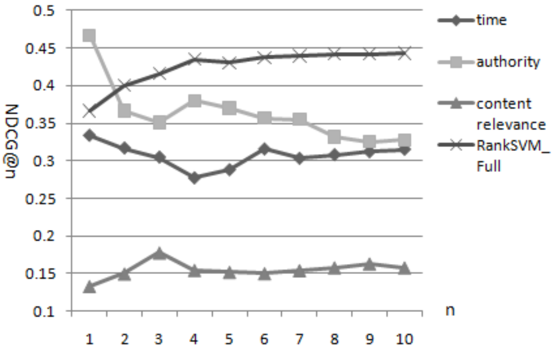


Figure 5. Performance of Four Ranking Methods

6.2 Feature Selection

As the RankSVM_Full underperforms against some models trained from subsets of features, we use an advanced greedy feature selection method and find the best feature conjunction to improve the performance of RankSVM_full. Figure 6 shows the feature selection approach.

Although greedy feature selection approach is commonly used in many problems, it does not work efficiently in addressing this problem partly for data sparseness. It is always blocked by a local optimum feature set. In order to resolve this problem, we first generate several feature sets randomly and run the greedy selection algorithm based the best one among them. Finally, we find the best feature conjunction composed by *URL, Sum_mention, First_List, Length,* and *Important_follower*, from which a model is learnt denoted as RankSVM_Best. Figure 7 illustrates that this model outperforms RankSVM_Full by about 15.3% on NDCG@10.

```
An advanced greedy feature selection algorithm.
Input: All features we extracted.
Output: the best feature conjunction BFC
Procedure:
Step1: Randomly generate 80 feature set F.
Step 2: Evaluate every feature set in F and select the best one denoted by RBF.
Features excluded those in RBF are denoted as EX_RBF
Step 3: t = 0,BFC(t)=RBF;
  Repeat
    Foreach feature in EX_RBF
      If  Evaluation(BFC)
          < Evaluation(BFC, feature)
          BFC(t+1) = {BFC(t), feature}
          EX_RBF(t+1) = EX_RBF(t) – {feature}
  While BFC(t+1) ≠ BFC(t)
Note: Evaluation(BFC) refers to the performance of ranking function trained from features in BFC on validation data.
```

Figure 6. Advanced Greedy Feature Selection Algorithm

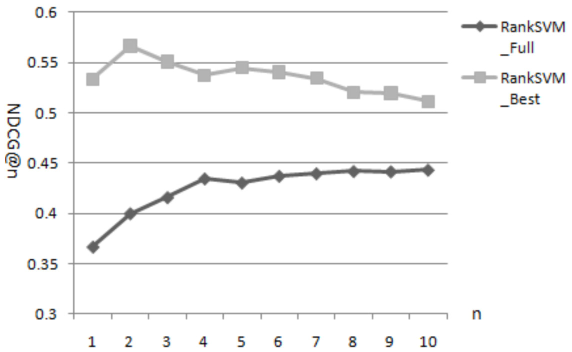


Figure 7. Comparison between RankSVM_Full and RankSVM_Best

We conduct a paired t-test between RankSVM_Best and each of other four ranking methods on NDCG@10 of ten test queries. The results demonstrate that RankSVM_Best outperforms ranking through time, account authority and content relevance respectively with a significance level of 0.01, and RankSVM_Full with a level of 0.05.

6.3 Feature Analysis

We are interested in which features in particular are highly valued by our model for tweet ranking. We evaluate the importance of each feature by the decrement of performance when removing the feature measured from RankSVM_Best. Figure 8 reveals the importance of each feature in our model.

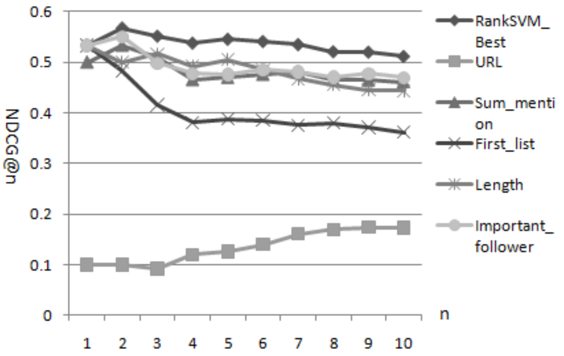


Figure 8. Importance of Each Feature

We observe from Figure 8 that URL is very important for our model; without it the performance declines seriously (with a significance level of 0.001). The reason may be that URLs shared in tweets, which provide more detailed information beyond the tweet's 140 characters, may be relevant to the query at a high probability.

Another useful feature is the number of lists that the author of the tweet has been listed in. The performance of ranking decreases with a significance level of 0.05 when removing it from the best feature combination. However, other features do not show significant contribution.

7 Discussion

Our experiment in section 6.2 demonstrates that features such as Hash tag Score and Retweet Count are not as effective as expected. This may be due to the small size of training data. We present an approach to learn an effective tweets ranker in a small dataset through feature selection. However, 20 queries are not sufficient to train a powerful ranker for Twitter.

In this study, to minimize the annotation effort, for each test query, we only annotate the tweets containing the query (returned by Twitter Search) and then used them for evaluation. With this kind of evaluation, it is hard to completely evaluate the significance of some features, such as content relevance features. In the future, we will select more queries including both hot searches and long tail searches, and select tweets for annotation directly from the twitter firehose.

There is also an opportunity for more accurate retweet relation detection in our work. At present, we just identify the retweet whose original tweet has not been modified, which leaves out a fair amount of retweet information. We would need to develop a more precise retweet relation detection method.

8 Conclusion

In this paper, we study three types of tweet features and propose a tweet ranking strategy by applying learning to rank algorithm. We find a set of most effective features for tweet ranking. The results of experiments demonstrate that the system using *Sum_mention*, *First_list*, *Important_follower*, *length* and *URL* performs best. In particular, whether a tweet contains a URL is the most effective feature. Additionally, we find in the experiments that the number of times the account is listed by other users is an effective representation of account authority and performs better than the number of followers that is widely used in previous work.

There are many aspects we would like to explore in the future. First, this research is based on the search results returned from Twitter which contains the input query. The tweets not containing the queries are not returned. We will explore query expansion approaches to improve the recall of the search results. We did not consider spam issues in the ranking process. However, spam filtering is important to all types of search engines. We will explore the impacts of spam and work out a spam filtering approach.

References

Chen Jilin, Rowan Nairn, Les Nelson, Michael Bernstein, and Ed H. Chi. 2010. Short and Tweet: Experiments on Recommending Content from Information Streams. *In the Proceedings of the 28th International conference on Human Factors in Computing Systems*, Pages: 1185-1194.

Chen Zhi, Li Zhang, Weihua Wang. 2008. PostingRank: Bringing Order to Web Forum Postings. *In the proceedings of the 4th Asia Information Retrieval Symposium*, Pages: 377-384.

Dong Anlei, Ruiqiang Zhang, Pranam Kolari, Jing Bai, Fernando Diaz, Yi Chang, Zhaohui Zheng, and Hongyuan Zha. 2010. Time of the essence: improving recency ranking using Twitter data. *In the proceedings of the 19th International Conference on World Wide Web*, Pages: 331-340.

Fujimura Ko, Takafumi Inoue, and Masayuki Sugisaki. 2005. The EigenRumor Algorithm for Ranking Blogs. *In the proceedings of the 2nd Annual Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics, World Wide Web*.

Jarvelin Kalervo, and Jaana Kekalainen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, Volume 20, Pages: 422-446.

Java Akshay, Xiaodan Song, Tim Finin, and Belle Tseng. 2007. Why we twitter: Understanding Microblogging Usage and Communities. *In the proceedings of the 9th International Workshop on Knowledge Discovery on the Web and the 1st International Workshop on Social Networks Analysis*. Pages: 118-138.

Joachims Thorsten. 1999. Making Large-Scale SVM Learning Practical. *Advances in Kernel Methods: Support Vector Learning*, Pages: 169-184.

Pear Analytics. 2009. Twitter Study-August 2009.

Kritikopoulos Apostolos, Martha Sideri, and Iraklis Varlamis. 2006. BlogRank: Ranking Weblogs Based on Connectivity and Similarity Features. *In the proceedings of the 2nd International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications*.

Leavitt Alex, Evan Burchard, David Fisher, and Sam Gilbert. 2009. The Influentials: New Approaches for Analyzing Influence on Twitter. *A publication of the Web Ecology Project*.

Liu Hongbo, Jiahai Yang, Jiaxin Wang, Yu Zhang. 2007. A Link-Based Rank of Postings in Newsgroup. *In the proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition*, Pages: 392-403.

Page Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank Citation Ranking: Bring Order to the Web. *Technical report*, Stanford University.

Robertson Stephen E., Steve Walker, and Micheline Hancock-Beaulieu. 1998. Okapi at TREC-7: Automatic Ad Hoc, Filtering, VLC and Interactive. *In the Proceedings of the 7th Text Retrieval Conference*. Pages: 199-210

Song Young-In, Chin-Yew Lin, Yunbo Cao, and Hae-Chang Rim. 2008. Question Utility: A Novel Static Ranking of Question Search. *In the Proceedings of the 23rd AAAI Conference on Artificial Intelligence*. Pages: 1231-1236

Sun Aaron R., Jiesi Cheng, and Daniel D. Zeng. 2009. A Novel Recommendation Framework for Micro-blogging based on Information Diffusion. *In the proceedings of the 19th Workshop on Information Technologies and Systems*.

Xi Wensi, Jesper Lind, and Eric Brill. 2004. Learning effective ranking functions for newsgroup search. *In the proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pages: 394-401

Xu Gu, and Ma Wei-Ying. 2006. Building Implicit Links from Content for Forum Search. *In the proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Pages: 300-307.

Translation Model Generalization using Probability Averaging for Machine Translation

Nan Duan[1]**, Hong Sun**
School of Computer Science and Technology
Tianjin University
v-naduan@microsoft.com
v-hongsun@microsoft.com

Ming Zhou
Microsoft Research Asia
mingzhou@microsoft.com

Abstract

Previous methods on improving translation quality by employing multiple SMT models usually carry out as a second-pass decision procedure on hypotheses from multiple systems using extra features instead of using features in existing models in more depth. In this paper, we propose *translation model generalization* (TMG), an approach that updates probability feature values for the translation model being used based on the model itself and a set of auxiliary models, aiming to enhance translation quality in the first-pass decoding. We validate our approach on translation models based on auxiliary models built by two different ways. We also introduce novel probability variance features into the log-linear models for further improvements. We conclude that our approach can be developed independently and integrated into current SMT pipeline directly. We demonstrate BLEU improvements on the NIST Chinese-to-English MT tasks for single-system decodings, a system combination approach and a model combination approach.

1 Introduction

Current research on Statistical Machine Translation (SMT) has made rapid progress in recent decades. Although differed on paradigms, such as phrase-based (Koehn, 2004; Och and Ney, 2004), hierarchical phrase-based (Chiang, 2007) and syntax-based (Galley *et al.*, 2006; Shen *et al.*, 2008; Huang, 2008), most SMT systems follow the similar pipeline and share common translation probability features which constitute the principal components of translation models. However, due to different model structures or data distributions, these features are usually assigned with different values in different translation models and result in translation outputs with individual advantages and shortcomings.

In order to obtain further improvements, many approaches have been explored over multiple systems: system combination based on confusion network (Matusov *et al.*, 2006; Rosti *et al.*, 2007; Li *et al.*, 2009a) develop on multiple *N*-best outputs and outperform primary SMT systems; consensus-based methods (Li *et al.*, 2009b; DeNero *et al.*, 2010), on the other hand, avoid the alignment problem between translations candidates and utilize *n*-gram consensus, aiming to optimize special decoding objectives for hypothesis selection. All these approaches act as the second-pass decision procedure on hypotheses from multiple systems by using extra features. They begin to work only after the generation of translation hypotheses has been finished.

In this paper, we propose *translation model generalization* (TMG), an approach that takes effect during the first-pass decoding procedure by updating translation probability features for the translation model being used based on the model itself and a set of auxiliary models. Bayesian Model Averaging is used to integrate values of identical features between models. Our contributions mainly include the following 3 aspects:

- *Alleviate the model bias problem based on translation models with different paradigms.* Because of various model constraints, translation models based on different paradigms could have individual biases. For instance, phrase-based models prefer translation pairs with high frequencies and assign them high

[1] This work has been done while the author was visiting Microsoft Research Asia.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 304–312,
Beijing, August 2010

probability values; yet such pairs could be disliked or even be absent in syntax-based models because of their violation on syntactic restrictions. We alleviate such model bias problem by using the generalized probability features in first-pass decoding, which computed based on feature values from all translation models instead of any single one.

- *Alleviate the over-estimation problem based on translation models with an identical paradigm but different training corpora.*
  In order to obtain further improvements by using an existing training module built for a specified model paradigm, we present a random data sampling method inspired by bagging (Breiman, 1996) to construct translation model ensembles from a unique data set for usage in TMG. Compared to results of TMG based on models with different paradigms, TMG based on models built in such a way can achieve larger improvements.
- *Novel translation probability variance features introduced.*
  We present how to compute the variance for each probability feature based on its values in different involved translation models with prior model probabilities. We add them into the log-linear model as new features to make current SMT models to be more flexible.

The remainder of this paper is organized as follows: we review various translation models in Section 2. In Section 3, we first introduce Bayesian Model Averaging method for SMT tasks and present a generic TMG algorithm based on it. We then discuss two solutions for constructing TM ensembles for usage in TMG. We next introduce probability variance features into current SMT models as new features. We evaluate our method on four state-of-the-art SMT systems, a system combination approach and a model combination approach. Evaluation results are shown in Section 4. In Section 5, we discuss some related work. We conclude the paper in Section 6.

2 Summary of Translation Models

Translation Model (TM) is the most important component in current SMT framework. It provides basic translation units for decoders with a series of probability features for model scoring. Many literatures have paid attentions to TMs from different aspects: DeNeefe *et al.* (2007) compared strengths and weaknesses of a phrase-based TM and a syntax-based TM from the *statistic* aspect; Zollmann *et al.* (2008) made a systematic comparison of three TMs, including phrasal, hierarchical and syntax-based, from the *performance* aspect; and Auli *et al.* (2009) made a systematic analysis of a phrase-based TM and a hierarchical TM from the *search space* aspect.

Given a word-aligned training corpus, we separate a TM training procedure into two phases: *extraction phase* and *parameterization phase*.

Extraction phase aims to pick out all valid translation pairs that are consistent with predefined model constraints. We summarize current TMs based on their corresponding model constraints into two categories below:

- *String-based* TM (string-to-string): reserves all translation pairs that are consistent with word alignment and satisfy length limitation. SMT systems using such TMs can benefit from a large convergence of translation pairs.
- *Tree-based* TM (string-to-tree, tree-to-string or tree-to-tree): needs to obey syntactic restrictions in one side or even both sides of translation candidates. The advantage of using such TMs is that translation outputs trend to be more syntactically well-formed.

Parameterization phase aims to assign a series of probability features to each translation pair. These features play the most important roles in the decision process and are shared by most current SMT decoders. In this paper, we mainly focus on the following four commonly used dominant probability features including:

- translation probability features in two directions: $p(\bar{e}|\bar{f})$ and $p(\bar{f}|\bar{e})$
- lexical weight features in two directions: $p_{lex}(\bar{e}|\bar{f})$ and $p_{lex}(\bar{f}|\bar{e})$

Both string-based and tree-based TMs are state-of-the-art models, and each extraction approach has its own strengths and weaknesses comparing to others. Due to different predefined model constraints, translation pairs extracted by different models usually have different distributions, which could directly affect the resulting probability feature values computed in parameterization phase. In order to utilize translation pairs more fairly in decoding, it is desirable to use more information to measure the quality of translation pairs based on different TMs rather than totally believing any single one.

3 Translation Model Generalization

We first introduce Bayesian Model Averaging method for SMT task. Based on it, we then formally present the generic TMG algorithm. We also provide two solutions for constructing TM ensembles as auxiliary models. We last introduce probability variance features based on multiple TMs for further improvements.

3.1 Bayesian Model Averaging for SMT

Bayesian Model Averaging (BMA) (Hoeting *et al.*, 1999) is a technique designed to solve uncertainty inherent in model selection.

Specifically, for SMT tasks, f is a source sentence, $\mathcal{D}$ is the training data, $\mathcal{M}_k$ is the k^{th} SMT model trained on $\mathcal{D}_k \subset \mathcal{D}$, $p_k(\cdot\,|f,e)$ represents the probability score predicted by $\mathcal{M}_k$ that f can be translated into a target sentence e. BMA provides a way to combine decisions of all $K+1$ SMT models by computing the final translation probability score $\bar{p}_E(\cdot\,|f,e,\mathcal{D})$ as follows:

$$\bar{p}_E(\cdot\,|f,e,\mathcal{D}) = \sum_{k=0}^{K} p(\mathcal{M}_k|\mathcal{D}_k)p_k(\cdot\,|f,e), \quad (1)$$

where $p(\mathcal{M}_k|\mathcal{D}_k)$ is the prior probability that $\mathcal{M}_k$ is a true model. For convenience, we will omit all symbols $\mathcal{D}_k$ in following descriptions.

Ideally, if all involved models $\{\mathcal{M}_0, \dots, \mathcal{M}_K\}$ share the same search space, then translation hypotheses could only be differentiated in probability scores assigned by different SMT models. In such case, BMA can be straightly developed on the whole SMT models in either span level or sentence level to re-compute translation scores of hypotheses for better rankings. However, because of various reasons, e.g. different pruning methods, different training data used, different generative capabilities of SMT models, search spaces between different models are always not identical. Thus, it is intractable to develop BMA on the whole SMT model level directly.

As a tradeoff, we notice that translation pairs between different TMs share a relatively large convergence because of word length limitation. So we instead utilize BMA method to multiple TMs by re-computing values of probability features between them, and we name this process as translation model generalization.

3.2 A Generic BMA-based TMG Algorithm

For a translation model $\mathcal{M}_0$, TMG aims to recompute its values of probability features based on itself and K collaborative TMs $\{\mathcal{M}_1, \dots, \mathcal{M}_K\}$. We describe the re-computation process for an arbitrary feature $p(\cdot\,|\bar{f},\bar{e}) \in \mathcal{M}_0$ as follows:

$$\bar{p}_E(\cdot\,|\bar{f},\bar{e}) = \sum_{k=0}^{K} p(\mathcal{M}_k)p_k(\cdot\,|\bar{f},\bar{e}), \quad (2)$$

where $p_k(\cdot\,|\bar{f},\bar{e})$ is the feature value assigned by $\mathcal{M}_k$. We denote $\mathcal{M}_0$ as the ***main model***, and other collaborative TMs as ***auxiliary models***. Figure 1 describes an example of TMG on two TMs, where the main model is a phrasal TM.

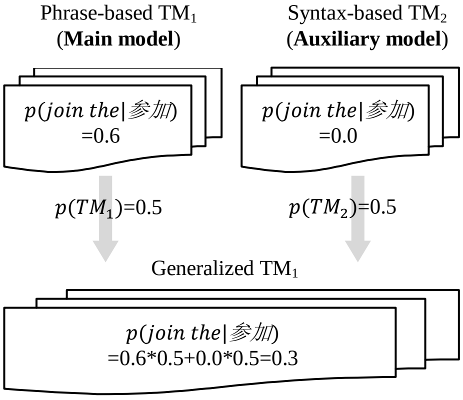


Figure 1. TMG applied to a phrasal TM (main model) and a syntax-based TM (auxiliary model). The value of a translation probability feature $p(join\ the|$参加$)$ in TM$_1$ is de-valued (from 0.6 to 0.3), in which ‘join the’ is absent in TM$_2$ because of its bad syntactic structure.

Equation 2 is a general framework that can be applied to all TMs. The only limitation is that the segmentation (or tokenization) standards for source (or target) training sentences should be identical for all models. We describe the generic TMG procedure in Algorithm 1[2].

[2] In this paper, since all data sets used have relative large sizes and all SMT models have similar performances, we heuristically set all $p(\mathcal{M}_k)$ equally to $1/(K+1)$.

Algorithm 1: TMG for a main model $\mathcal{M}_0$

```
for the $k^{\text{th}}$ auxiliary TM do
    run training procedure on $\mathcal{D}_k$ with specified model constraints and generate $\mathcal{M}_k$
end for
for each translation pair $< \bar{f}, \bar{e} >$ in $\mathcal{M}_0$ do
    for each probability feature $p(\cdot\,|\bar{f}, \bar{e})$ do
        for each translation model $\mathcal{M}_k$ do
            if $< \bar{f}, \bar{e} >$ is contained in $\mathcal{M}_k$ then
                $\bar{p}_E(\cdot\,|\bar{f}, \bar{e}) \mathrel{+}= p(\mathcal{M}_k) p_k(\cdot\,|\bar{f}, \bar{e})$
            end if
        end for
    end for
end for
return the generalized $\mathcal{M}_0$ for SMT decoding
```

3.3 Auxiliary Model Construction

In order to utilize TMG, more than one TM as auxiliary models is needed. Building TMs with different paradigms is one solution. For example, we can build a syntax-based TM as an auxiliary model for a phrase-based TM. However, it has to re-implement more complicated TM training modules besides the existing one.

In this sub-section, we present an alternative solution to construct auxiliary model ensembles by using the existing training module with different training data extracted from a unique data set. We describe the general procedure for constructing K auxiliary models as follows:

1) Given a unique training corpus $\mathcal{D}$, we randomly sample $N\%$ bilingual sentence pairs without replacement and denote them as $\mathcal{D}_i$. N is a number determined empirically;
2) Based on $\mathcal{D}_i$, we *re-do* word alignment and train an auxiliary model $\mathcal{M}_i$ using the existing training module;
3) We execute Step 1 and Step 2 iteratively for K times, and finally obtain K auxiliary models. The optimal setting of K for TMG is also determined empirically.

With all above steps finished, we can perform TMG as we described in Algorithm 1 based on the K auxiliary models generated already.

The random data sampling process described above is very similar to bagging except for it not allowing replacement during sampling. By making use of this process, translation pairs with low frequencies have relatively high probabilities to be totally discarded, and in resulting TMs, their probabilities could be zero; meanwhile, translation pairs with high frequencies still have high probabilities to be reserved, and hold similar probability feature values in resulting TMs comparing to the main model. Thus, after TMG procedure, feature values could be smoothed for translation pairs with low frequencies, and be stable for translation pairs with high frequencies. From this point of view, TMG can also be seen as a TM smoothing technique based on multiple TMs instead of single one such as Foster *et al.* (2006). We will see in Section 4 that TMG based on TMs generated by both of these two solutions can improve translation quality for all baseline decoders on a series of evaluation sets.

3.4 Probability Variance Feature

The re-computed values of probability features in Equation 2 are actually the feature expectations based on their values from all involved TMs. In order to give more statistical meanings to translation pairs, we also compute their corresponding feature variances based on feature expectations and TM-specified feature values with prior probabilities. We introduce such variances as new features into the log-linear model for further improvements. Our motivation is to quantify the differences of model preferences between TMs for arbitrary probability features.

The variance for an arbitrary probability feature $p(\cdot) \in \mathcal{M}_0$ can be computed as follows:

$$p_V(\cdot) = \sum_{k=0}^{K} \{p_k(\cdot) - \bar{p}_E(\cdot)\}^2 p_k(\mathcal{M}_k), \qquad (3)$$

where $\bar{p}_E(\cdot)$ is the feature expectation computed by Equation 2, $p_k(\cdot)$ is the feature value predicted by $\mathcal{M}_k$, and $p_k(\mathcal{M}_k)$ is the prior probability for $\mathcal{M}_k$. Each probability feature now corresponds to a variance score. We extend the original feature set of $\mathcal{M}_0$ with variance features added in and list the updated set below:

- translation probability expectation features in two directions: $\bar{p}_E(\bar{e}|\bar{f})$ and $\bar{p}_E(\bar{f}|\bar{e})$
- translation probability variance features in two directions: $p_V(\bar{e}|\bar{f})$ and $p_V(\bar{f}|\bar{e})$
- lexical weight expectation features in two directions: $\bar{p}_{E_{lex}}(\bar{e}|\bar{f})$ and $\bar{p}_{E_{lex}}(\bar{f}|\bar{e})$
- lexical weight variance features in two directions: $p_{V_{lex}}(\bar{e}|\bar{f})$ and $p_{V_{lex}}(\bar{f}|\bar{e})$

4 Experiments

4.1 Data Condition

We conduct experiments on the NIST Chinese-to-English MT tasks. We tune model parameters on the NIST 2003 (*MT03*) evaluation set by MERT (Och, 2003), and report results on NIST evaluation sets including the NIST 2004 (*MT04*), the NIST 2005 (*MT05*), the newswire portion of the NIST 2006 (*MT06*) and 2008 (*MT08*). Performances are measured in terms of the case-insensitive BLEU scores in percentage numbers. Table 1 gives statistics over these evaluation sets.

	MT03	MT04	MT05	MT06	MT08
Sent	919	1,788	1,082	616	691
Word	23,788	48,215	29,263	17,316	17,424

Table 1. Statistics on dev/test evaluation sets

We use the *selected data* that picked out from the whole data available for the NIST 2008 constrained track of Chinese-to-English machine translation task as the training corpora, including LDC2003E07, LDC2003E14, LDC2005T06, LDC2005T10, LDC2005E83, LDC2006E26, LDC2006E34, LDC2006E85 and LDC2006E92, which contain about 498,000 sentence pairs after pre-processing. Word alignments are performed by GIZA++ (Och and Ney, 2000) in both directions with an *intersect-diag-grow* refinement.

A traditional 5-gram language model (LM) for all involved systems is trained on the English side of all bilingual data plus the Xinhua portion of LDC English Gigaword Version 3.0. A lexicalized reordering model (Xiong *et al.*, 2006) is trained on the selected data in maximum entropy principle for the phrase-based system. A tri-gram target dependency LM (DLM) is trained on the English side of the selected data for the dependency-based hierarchical system.

4.2 MT System Description

We include four baseline systems. The first one (***Phr***) is a phrasal system (Xiong *et al.*, 2006) based on Bracketing Transduction Grammar (Wu, 1997) with a lexicalized reordering component based on maximum entropy model. The second one (***Hier***) is a hierarchical phrase-based system (Chiang, 2007) based on Synchronous Context Free Grammar (SCFG). The third one (***Dep***) is a string-to-dependency hierarchical phrase-based system (Shen *et al.*, 2008) with a dependency language model, which translates source strings to target dependency trees. The fourth one (***Synx***) is a syntax-based system (Galley *et al.*, 2006) that translates source strings to target syntactic trees.

4.3 TMG based on Multiple Paradigms

We develop TMG for each baseline system's TM based on the other three TMs as auxiliary models. *All prior probabilities of TMs are set equally to 0.25 heuristically as their similar performances.* Evaluation results are shown in Table 2, where gains more than 0.2 BLEU points are highlighted as improved cases. Compared to baseline systems, systems based on generalized TMs improve in most cases (18 times out of 20). We also notice that the improvements achieved on tree-based systems (Dep and Synx) are relatively smaller than those on string-based systems (Phr and Hier). A potential explanation can be that with considering more syntactic restrictions, tree-based systems suffer less than string-based systems on the over-estimation problem. We do not present further results with variance features added because of their consistent un-promising numbers. *We think this may be due to the considerable portion of non-overlapping translation pairs between main model and auxiliary models, which cause the variances not so accurate*.

		MT03(dev)	MT04	MT05	MT06	MT08	Average
Phr	Baseline	40.45	39.21	38.03	34.24	30.21	36.43
	TMG	41.19**(+0.74)**	39.74**(+0.53)**	38.39**(+0.36)**	34.71**(+0.47)**	30.69**(+0.48)**	36.94**(+0.51)**
Hier	Baseline	41.30	39.63	38.83	34.63	30.46	36.97
	TMG	41.67**(+0.37)**	40.25**(+0.62)**	39.11**(+0.28)**	35.78**(+1.15)**	31.17**(+0.71)**	37.60**(+0.63)**
Dep	Baseline	41.10	39.81	39.47	35.72	30.50	37.32
	TMG	41.37**(+0.27)**	39.92(+0.11)	39.91**(+0.44)**	35.99**(+0.27)**	31.07**(+0.57)**	37.65**(+0.33)**
Synx	Baseline	41.02	39.88	39.47	36.41	32.15	37.79
	TMG	41.26**(+0.24)**	40.09**(+0.21)**	39.90**(+0.43)**	36.77**(+0.36)**	32.15(+0.00)	38.03**(+0.24)**

Table 2. Results of TMG based on TMs with different paradigms

4.4 TMG based on Single Paradigm

We then evaluate TMG based on auxiliary models generated by the random sampling method.

We first decide the percentage of training data to be sampled. We empirically vary this number by 20%, 40%, 60%, 80% and 90% and use each sampled data to train an auxiliary model. We then run TMG on the baseline TM with different auxiliary model used each time. For time saving, we only evaluate on MT03 for Phr in Figure 2.

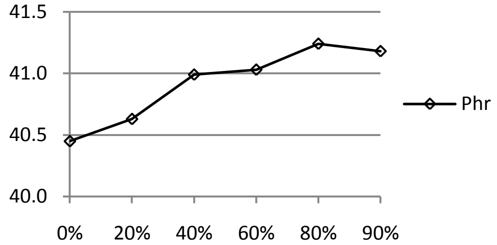


Figure 2. Affects of different percentages of data

The optimal result is achieved when the percentage is 80%, and we fix it as the default value in following experiments.

We then decide the number of auxiliary models used for TMG by varying it from 1 to 5. We list different results on MT03 for Phr in Figure 3.

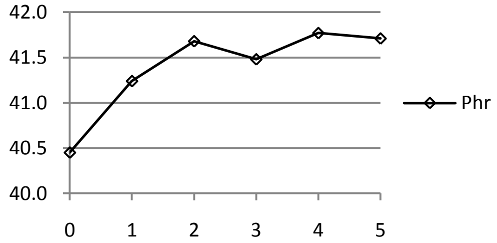


Figure 3. Affects of different numbers of auxiliary models

The optimal result is achieved when the number of auxiliary models is 4, and we fix it as the default value in following experiments.

We now develop TMG for each baseline system's TM based on auxiliary models constructed under default settings determined above. Evaluation results are shown in Table 3. We also investigate the affect of variance features for performance, whose results are denoted as ***TMG+Var***.

From Table 3 we can see that, compared to the results on baseline systems, systems using generalized TMs obtain improvements on almost all evaluation sets (19 times out of 20). With probability variance features added further, the improvements become even more stable than the ones using TMG only (20 times out of 20). Similar to the trend in Table 2, we also notice that TMG method is more preferred by string-based systems (Phr and Hier) rather than tree-based systems (Dep and Synx). This makes our conclusion more solidly that syntactic restrictions can help to alleviate the over-estimation problem.

4.5 Analysis on Phrase Coverage

We next empirically investigate on the translation pair coverage between TM ensembles built by different ways, and use them to analyze results got from previous experiments. Here, we only focus on *full lexicalized* translation entries between models. Those entries with variables are out of consideration in comparisons because of their model dependent properties.

Phrase pairs in the first three TMs have a length limitation in source side up to 3 words, and each source phrase can be translated to at most 20 target phrases.

		MT03(dev)	MT04	MT05	MT06	MT08	Average
	Baseline	40.45	39.21	38.03	34.24	30.21	36.43
Phr	TMG	41.77**(+1.32)**	40.28**(+1.07)**	39.13**(+1.10)**	35.38**(+1.14)**	31.12**(+0.91)**	37.54**(+1.11)**
	TMG+Var	41.77**(+1.32)**	40.31**(+1.10)**	39.43**(+1.30)**	35.61**(+1.37)**	31.62**(+1.41)**	37.74**(+1.31)**
	Baseline	41.30	39.63	38.83	34.63	30.46	36.97
Hier	TMG	42.28**(+0.98)**	40.45**(+0.82)**	39.61**(+0.78)**	35.67**(+1.04)**	31.54**(+1.08)**	37.91**(+0.94)**
	TMG+Var	42.42**(+1.12)**	40.55**(+0.92)**	39.69**(+0.86)**	35.55**(+0.92)**	31.41**(+0.95)**	37.92**(+0.95)**
	Baseline	41.10	39.81	39.47	35.72	30.50	37.32
Dep	TMG	41.49**(+0.39)**	40.20**(+0.39)**	40.00**(+0.53)**	36.13**(+0.41)**	31.24**(+0.74)**	37.81**(+0.49)**
	TMG+Var	41.72**(+0.62)**	40.57**(+0.76)**	40.44**(+0.97)**	36.15**(+0.43)**	31.31**(+0.81)**	38.04**(+0.72)**
	Baseline	41.02	39.88	39.47	36.41	32.15	37.79
Synx	TMG	41.18(+0.16)	40.30**(+0.42)**	39.90**(+0.43)**	36.99**(+0.58)**	32.45**(+0.30)**	38.16**(+0.37)**
	TMG+Var	41.42**(+0.40)**	40.55**(+0.67)**	40.17**(+0.70)**	36.89**(+0.48)**	32.51**(+0.36)**	38.31**(+0.52)**

Table 3. Results of TMG based on TMs constructed by random data sampling

For the fourth TM, these two limitations are released to 4 words and 30 target phrases. We treat phrase pairs identical on both sides but with different syntactic labels in the fourth TM as a unique pair for conveniences in statistics.

We first make statistics on TMs with different paradigms in Table 4. We can see from Table 4 that only slightly over half of the phrase pairs contained by the four involved TMs are common, which is also similar to the conclusion drawn in DeNeefe *et al.* (2006).

Models	#Translation Pair	#Percentage
Phr	1,222,909	**50.6%**
Hier	1,222,909	**50.6%**
Dep	1,087,198	**56.9%**
Synx	1,188,408	**52.0%**
Overlaps	618,371	-

Table 4. Rule statistics on TMs constructed by different paradigms

We then make statistics on TMs with identical paradigm in Table 5. For each baseline TM and its corresponding four auxiliary models constructed by random data sampling, we count the number of phrase pairs that are common between them and compute the percentage numbers based on it for each TM individually.

Models	TM_0	TM_1	TM_2	TM_3	TM_4
Phr	**61.8%**	74.0%	74.1%	73.9%	74.1%
Hier	**61.8%**	74.0%	74.1%	73.9%	74.1%
Dep	**60.8%**	73.6%	73.6%	73.5%	73.7%
Synx	**57.2%**	68.4%	68.5%	68.5%	68.6%

Table 5. Rule statistics on TMs constructed by random sampling (TM_0 is the main model)

Compared to the numbers in Table 4, we find that the coverage between baseline TM and sampled auxiliary models with identical paradigm is larger than that between baseline TM and auxiliary models with different paradigms (about 10 percents). It is a potential reason can explain why results of TMG based on sampled auxiliary models are more effective than those based on auxiliary models built with different paradigms, as we infer that *they share more common phrase pairs each other and make the computation of feature expectations and variances to be more reliable and accurate*.

4.6 Improvements on System Combination

Besides working for single-system decoding, we also perform a system combination method on *N*-best outputs from systems using generalized TMs. We re-implement a state-of-the-art word-level System Combination (**SC**) approach based on incremental HMM alignment proposed by Li *et al.* (2009a). The default number of *N*-best candidates used is set to 20.

We evaluate SC on *N*-best outputs generated from 4 baseline decoders by using different TM settings and list results in Table 6, where ***Base*** stands for combination results on systems using default TMs; ***Paras*** stands for combination results on systems using TMs generalized based on auxiliary models with different paradigms; and ***Samp*** stands for combination results on systems using TMs generalized based on auxiliary models constructed by the random data sampling method. For the Samp setting, we also include probability variance features computed based on Equation 3 in the log-linear model.

SC	MT03	MT04	MT05	MT06	MT08
Base	44.20	42.30	41.22	37.77	33.07
Paras	**44.40**	**42.69**	**41.53**	**38.05**	**33.31**
Samp	**44.80**	**42.95**	**42.10**	**38.39**	**33.67**

Table 6. Results on system combination

From Table 6 we can see that system combination can benefit from TMG method.

4.7 Improvements on Model Combination

As an alternative, model combination is another effective way to improve translation performance by utilizing multiple systems. We re-implement the Model Combination (**MC**) approach (DeNero *et al.*, 2010) using *N*-best lists as its inputs and develop it on *N*-best outputs used in Table 6. Evaluation results are presented in Table 7.

MC	MT03	MT04	MT05	MT06	MT08
Base	42.31	40.57	40.31	38.65	33.88
Paras	**42.87**	**40.96**	**40.77**	**38.81**	**34.47**
Samp	**43.29**	**41.29**	**41.11**	**39.28**	**34.77**

Table 7. Results on model combination

From Table 7 we can see that model combination can also benefit from TMG method.

5 Related Work

Foster and Kuhn (2007) presented an approach that resembles more to our work, in which they divided the training corpus into different components and integrated models trained on each component using the mixture modeling. However, their motivation was to address the *domain adaption problem*, and additional genre information should be provided for the corpus partition to create multiple models for mixture. We instead present two ways for the model ensemble construction without extra information needed: building models by different paradigms or by a random data sampling technique inspired by a machine learning technique. Compared to the prior work, our approach is more general, which can also be used for model adaptation. We can also treat TMG as a smoothing way to address the over-estimation problem existing in almost all TMs. Some literatures have paid attention to this issue as well, such as Foster *et al.* (2006) and Mylonakis and Sima 'an (2008). However, they did not leverage information between multiple models as we did, and developed on single models only. Furthermore, we also make current translation probability features to contain more statistical meanings by introducing the probability variance features into the log-linear model, which are completely novel to prior work and provide further improvements.

6 Conclusion and Future Work

In this paper, we have investigated a simple but effective translation model generalization method that benefits by integrating values of probability features between multiple TMs and using them in decoding phase directly. We also introduce novel probability variance features into the current feature sets of translation models and make the SMT models to be more flexible. We evaluate our method on four state-of-the-art SMT systems, and get promising results not only on single-system decodings, but also on a system combination approach and a model combination approach.

Making use of different distributions of translation probability features is the essential of this work. In the future, we will extend TMG method to other statistical models in SMT framework, (e.g. LM), which could be also suffered from the over-estimation problem. And we will make further research on how to tune prior probabilities of models automatically as well, in order to make our method to be more robust and tunable.

References

Auli Michael, Adam Lopez, Hieu Hoang, and Philipp Koehn. 2009. *A Systematic Analysis of Translation Model Search Spaces*. In *4th Workshop on Statistical Machine Translation*, pages 224-232.

Breiman Leo. 1996. *Bagging Predictors*. *Machine Learning*.

Chiang David. 2007. *Hierarchical Phrase Based Translation*. *Computational Linguistics*, 33(2): 201-228.

DeNero John, Shankar Kumar, Ciprian Chelba, and Franz Och. 2010. *Model Combination for Machine Translation.* To appear in *Proc. of the North American Chapter of the Association for Computational Linguistic.*

DeNeefe Steve, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. *What Can Syntax-based MT Learn from Phrase-based MT?* In *Proc. of Empirical Methods on Natural Language Processing*, pages 755-763.

Foster George, Roland Kuhn, and Howard Johnson. 2006. *Phrasetable Smoothing for Statistical Machine Translation.* In *Proc. of Empirical Methods on Natural Language Processing*, pages 53-61.

Foster George and Roland Kuhn. 2007. *Mixture-Model Adaptation for SMT.* In *2th Workshop on Statistical Machine Translation*, pages 128-135.

Galley Michel, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. *Scalable Inference and Training of Context-Rich Syntactic Translation Models*. In *Proc. of 44th Meeting of the Association for Computational Linguistics*, pages: 961-968.

Huang Liang. 2008. *Forest Reranking: Discriminative Parsing with Non-Local Features*. In *Proc. of 46th Meeting of the Association for Computational Linguistics*, pages 586-594.

Hoeting Jennifer, David Madigan, Adrian Raftery, and Chris Volinsky. 1999. *Bayesian Model Averaging: A tutorial. Statistical Science*, Vol. 14, pages 382-417.

He Xiaodong, Mei Yang, Jianfeng Gao, Patrick Nguyen, and Robert Moore. 2008. *Indirect-HMM-based Hypothesis Alignment for Combining Outputs from Machine Translation Systems*. In *Proc. of Empirical Methods on Natural Language Processing*, pages 98-107.

Koehn Philipp. 2004. *Phrase-based Model for SMT. Computational Linguistics*, 28(1): 114-133.

Li Chi-Ho, Xiaodong He, Yupeng Liu, and Ning Xi. 2009a. *Incremental HMM Alignment for MT system Combination*. In *Proc. of 47th Meeting of the Association for Computational Linguistics*, pages 949-957.

Li Mu, Nan Duan, Dongdong Zhang, Chi-Ho Li, and Ming Zhou. 2009b. *Collaborative Decoding: Partial Hypothesis Re-Ranking Using Translation Consensus between Decoders*. In *Proc. of 47th Meeting of the Association for Computational Linguistics*, pages 585-592.

Liu Yang, Haitao Mi, Yang Feng, and Qun Liu. 2009. *Joint Decoding with Multiple Translation Models*. In *Proc. of 47th Meeting of the Association for Computational Linguistics*, pages 576-584.

Mylonakis Markos and Khalil Sima 'an. 2008. *Phrase Translation Probabilities with ITG Priors and Smoothing as Learning Objective*. In *Proc. of Empirical Methods on Natural Language Processing*, pages 630-639.

Matusov Evgeny, Nicola Ueffi ng, and Hermann Ney. 2006. Computing consensus translation from multiple machine translation systems using enhanced hypotheses alignment. In *Proc. of European Charter of the Association for Computational Linguistics*, pages 33-40.

Och Franz and Hermann Ney. 2000. *Improved Statistical Alignment Models*. In *Proc. of 38th Meeting of the Association for Computational Linguistics*, pages 440-447.

Och Franz. 2003. *Minimum Error Rate Training in Statistical Machine Translation*. In *Proc. of 41th Meeting of the Association for Computational Linguistics*, pages 160-167.

Och Franz and Hermann Ney. 2004. *The Alignment template approach to Statistical Machine Translation*. *Computational Linguistics*, 30(4): 417-449.

Shen Libin, Jinxi Xu, and Ralph Weischedel. 2008. *A new string-to-dependency machine translation algorithm with a target dependency language model*. In *Proc. of 46th Meeting of the Association for Computational Linguistics*, pages 577-585.

Wu Dekai. 1997. *Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora*. *Computational Linguistics*, 23(3): 377-404.

Xiong Deyi, Qun Liu, and Shouxun Lin. 2006. *Maximum Entropy based Phrase Reordering Model for Statistical Machine Translation*. In *Proc. of 44th Meeting of the Association for Computational Linguistics*, pages 521-528.

Zollmann Andreas, Ashish Venugopal, Franz Och, and Jay Ponte. 2008. *A Systematic Comparison of Phrase-Based, Hierarchical and Syntax-Augmented Statistical MT.* In *23rd International Conference on Computational Linguistics*, pages 1145-1152.

Mixture Model-based Minimum Bayes Risk Decoding using Multiple Machine Translation Systems

Nan Duan[1]
School of Computer Science and Technology
Tianjin University
v-naduan@microsoft.com

Mu Li, Dongdong Zhang, Ming Zhou
Microsoft Research Asia
muli@microsoft.com
dozhang@microsoft.com
mingzhou@microsoft.com

Abstract

We present Mixture Model-based Minimum Bayes Risk (MMMBR) decoding, an approach that makes use of multiple SMT systems to improve translation accuracy. Unlike existing MBR decoding methods defined on the basis of single SMT systems, an MMMBR decoder reranks translation outputs in the combined search space of multiple systems using the MBR decision rule and a mixture distribution of component SMT models for translation hypotheses. MMMBR decoding is a general method that is independent of specific SMT models and can be applied to various commonly used search spaces. Experimental results on the NIST Chinese-to-English MT evaluation tasks show that our approach brings significant improvements to single system-based MBR decoding and outperforms a state-of-the-art system combination method.

1 Introduction

Minimum Bayes Risk (MBR) decoding is becoming more and more popular in recent Statistical Machine Translation (SMT) research. This approach requires a second-pass decoding procedure to re-rank translation hypotheses by risk scores computed based on model's distribution.

Kumar and Byrne (2004) first introduced MBR decoding to SMT field and developed it on the *N*-best list translations. Their work has shown that MBR decoding performs better than Maximum a Posteriori (MAP) decoding for different evaluation criteria. After that, many dedicated efforts have been made to improve the performances of SMT systems by utilizing MBR-inspired methods. Tromble *et al.* (2008) proposed a linear approximation to BLEU score (log-BLEU) as a new loss function in MBR decoding and extended it from *N*-best lists to lattices, and Kumar *et al.* (2009) presented more efficient algorithms for MBR decoding on both lattices and hypergraphs to alleviate the high computational cost problem in Tromble *et al.*'s work. DeNero *et al.* (2009) proposed a fast consensus decoding algorithm for MBR for both linear and non-linear similarity measures.

All work mentioned above share a common setting: an MBR decoder is built based on one and only one MAP decoder. On the other hand, recent research has shown that substantial improvements can be achieved by utilizing consensus statistics over multiple SMT systems (Rosti *et al.*, 2007; Li *et al.*, 2009a; Li *et al.*, 2009b; Liu *et al.*, 2009). It could be desirable to adapt MBR decoding to multiple SMT systems as well.

In this paper, we present *Mixture Model-based Minimum Bayes Risk (MMMBR) decoding*, an approach that makes use of multiple SMT systems to improve translation performance. In this work, we can take advantage of a larger search space for hypothesis selection, and employ an improved probability distribution over translation hypotheses based on mixture modeling, which linearly combines distributions of multiple component systems for Bayes risk computation. The key contribution of this paper is the usage of mixture modeling in MBR, which allows multiple SMT models to be involved in and makes the computation of *n*-gram consensus statistics to be more accurate. Evaluation results have shown that our approach not only brings significant improvements to single system-based MBR decoding but also outperforms a state-of-the-art word-level system combination method.

[1] This work has been done while the author was visiting Microsoft Research Asia.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 313–321,
Beijing, August 2010

The rest of the paper is organized as follows: In Section 2, we first review traditional MBR decoding method and summarize various search spaces that can be utilized by an MBR decoder. Then, we describe how a mixture model can be used to combine distributions of multiple SMT systems for Bayes risk computation. Lastly, we present detailed MMMBR decoding model on multiple systems and make comparison with single system-based MBR decoding methods. Section 3 describes how to optimize different types of parameters. Experimental results will be shown in Section 4. Section 5 discusses some related work and Section 6 concludes the paper.

2 Mixture Model-based MBR Decoding

2.1 Minimum Bayes Risk Decoding

Given a source sentence F, MBR decoding aims to find the translation with the least expected loss under a probability distribution. The objective of an MBR decoder can be written as:

$$\hat{E} = \underset{E' \in \mathcal{H}_h}{\operatorname{argmin}} \sum_{E \in \mathcal{H}_e} L(E, E') P(E | F, \mathcal{H}_e). \qquad (1)$$

where $\mathcal{H}_h$ denotes a *search space* for hypothesis selection; $\mathcal{H}_e$ denotes an *evidence space* for Bayes risk computation; $L(\cdot)$ denotes a function that measures the loss between E' and E; $P(\cdot)$ is the underlying distribution based on $\mathcal{H}_e$.

Some of existing work on MBR decoding focused on exploring larger spaces for both $\mathcal{H}_h$ and $\mathcal{H}_e$, e.g. from *N*-best lists to lattices or hypergraphs (Tromble *et al.*, 2008; Kumar *et al.*, 2009). Various loss functions have also been investigated by using different evaluation criteria for similarity computation, e.g. Word Error Rate, Position-independent Word Error Rate, BLEU and log-BLEU (Kumar and Byrne, 2004; Tromble *et al.*, 2008). But less attention has been paid to distribution $P(\cdot)$. Currently, many SMT systems based on different paradigms can yield similar performances but are good at modeling different inputs in the translation task (Koehn *et al.*, 2004a; Och *et al.*, 2004; Chiang, 2007; Mi *et al.*, 2008; Huang, 2008). We expect to integrate the advantages of different SMT models into MBR decoding for further improvements. In particular, we make in-depth investigation into MBR decoding concentrating on the translation distribution $P(\cdot)$ by leveraging a mixture model based on multiple SMT systems.

2.2 Summary of Translation Search Spaces

There are three major forms of search spaces that can be obtained from an MAP decoder as a byproduct, depending on the design of the decoder: *N*-best lists, lattices and hypergraphs.

An N-best list contains the N most probable translation hypotheses produced by a decoder. It only presents a very small portion of the entire search space of an SMT model.

A hypergraph is a weighted acyclic graph which compactly encodes an exponential number of translation hypotheses. It allows us to represent both phrase-based and syntax-based systems in a unified framework. Formally, a hypergraph $\mathcal{H}$ is a pair $< \mathcal{V}, \mathcal{E} >$, where $\mathcal{V}$ is a set of hypernodes and $\mathcal{E}$ is a set of hyperedges. Each hypernode $v \in \mathcal{V}$ corresponds to translation hypotheses with identical decoding states, which usually include the span (i, j) of the words being translated, the grammar symbol s for that span and the left and right boundary words of hypotheses for computing language model (LM) scores. Each hyperedge $e \in \mathcal{E}$ corresponds to a translation rule and connects a head node $h(e)$ and a set of tail nodes $T(e)$. The number of tail nodes $|T(e)|$ is called the *arity* of the hyperedge *e* and the arity of a hypergraph is the maximum arity of its hyperedges. If the arity of a hyperedge *e* is zero, $h(e)$ is then called a *source node*. Each hypergraph has a unique *root node* and each path in a hypergraph induces a translation hypothesis. A lattice (Ueffing *et al.*, 2002) can be viewed as a special hypergraph, in which the maximum arity is one.

2.3 Mixture Model for SMT

We first describe how to construct a general distribution for translation hypotheses over multiple SMT systems using mixture modeling for usage in MBR decoding.

Mixture modeling is a technique that has been applied to many statistical tasks successfully. For the SMT task in particular, given K SMT systems with their corresponding model distributions, a mixture model is defined as a probability distribution over the combined search space of all component systems and computed as a weighted sum of component model distributions:

$$P(E|F,\mathcal{H}) = \sum_{k=1}^{K} \lambda_k P_k(E|F,\mathcal{H}_k). \quad (2)$$

In Equation 2, $\lambda_1, \lambda_2, \dots, \lambda_K$ are system weights which hold following constraints: $0 \leq \lambda_k \leq 1$ and $\sum_{k=1}^{K} \lambda_k = 1$, $P_k(E|F,\mathcal{H}_k)$ is the k^{th} distribution estimated on the search space $\mathcal{H}_k$ based on the log-linear formulation:

$$P_k(E|F,\mathcal{H}_k) = \frac{exp(\alpha_k \theta_k(E,F))}{\sum_{E' \in \mathcal{H}_k} exp(\alpha_k \theta_k(E',F))},$$

where $\theta_k(E,F)$ is the score function of the k^{th} system for translation E, $\alpha_k \in [0,\infty)$ is a scaling factor that determines the flatness of the distribution P_k sharp ($\alpha_k > 1$) or smooth ($\alpha_k < 1$).

Due to the inherent differences in SMT models, translation hypotheses have different distributions in different systems. A mixture model can effectively combine multiple distributions with tunable system weights. The distribution of a single model used in traditional MBR can be seen as a special mixture model, where K is one.

2.4 Mixture Model for SMT

Let $\{d_1, d_2, \dots, d_K\}$ denote K machine translation systems, $\mathcal{H}_i$ denotes the search space produced by system d_i in MAP decoding procedure. An MMMBR decoder aims to seek a translation from the combined search space $\mathcal{H} = \bigcup_i \mathcal{H}_i$ that maximizes the expected gain score based on a mixture model $P(E|F,\mathcal{H})$. We write the objective function of MMMBR decoding as:

$$\hat{E} = \operatorname*{argmax}_{E' \in \mathcal{H}} \sum_{E \in \mathcal{H}} G(E,E') P(E|F,\mathcal{H}). \quad (3)$$

For the gain function $G(\cdot)$, we follow Tromble *et al.* (2008) to use log-BLEU, which is scored by the hypothesis length and a linear function of *n*-gram matches as:

$$G(E,E') = \theta_0 |E'| + \sum_{\omega} \theta_{|\omega|} \#_\omega(E') \delta_\omega (E),$$

In this definition, E is a reference translation, $|E'|$ is the length of hypothesis E', ω is an *n*-gram presented in E', $\#_\omega(E')$ is the number of times that ω occurs in E', and $\delta_\omega(E)$ is an indicator function which equals to 1 when ω occurs in E and 0 otherwise. $\theta_0, \theta_1, \dots, \theta_N$ are model parameters, where N is the maximum order of the *n*-grams involved.

For the mixture model $P(\cdot)$, we replace it by Equation 2 and rewrite the total gain score for hypothesis E' in Equation 3:

$$\begin{aligned} &\sum_{E \in \mathcal{H}} G(E,E') P(E|F,\mathcal{H}) \\ &= \sum_{E \in \mathcal{H}} G(E,E') \sum_i \lambda_i P_i(E|F,\mathcal{H}_i) \\ &= \sum_i \lambda_i \sum_{E \in \mathcal{H}} G(E,E') P_i(E|F,\mathcal{H}_i) \\ &= \sum_i \lambda_i \sum_{k=1}^{K} \sum_{E \in \mathcal{H}_k} G(E,E') P_i(E|F,\mathcal{H}_i). \end{aligned} \quad (4)$$

In Equation 4, the total gain score on the combined search space $\mathcal{H}$ can be further decomposed into each local search space $\mathcal{H}_k$ with a specified distribution $P_i(E|F,\mathcal{H}_i)$. This is a nice property and it allows us to compute the total gain score as a weighted sum of local gain scores on different search spaces. We expand the local gain score for E' computed on search space $\mathcal{H}_k$ with $P_i(E|F,\mathcal{H}_i)$ using log-BLEU as:

$$\begin{aligned} &\sum_{E \in \mathcal{H}_k} G(E,E') P_i(E|F,\mathcal{H}_i) \\ &= \sum_{E \in \mathcal{H}_k} \left\{ \theta_0 |E'| + \sum_\omega \theta_{|\omega|} \#_\omega(E') \delta_\omega(E) \right\} P_i(E|F,\mathcal{H}_i) \\ &\approx \theta_0 |E'| + \sum_\omega \theta_{|\omega|} \#_\omega(E') p_i(\omega|\mathcal{H}_i). \end{aligned} \quad (5)$$

We make two approximations for the situations when $i \neq k$: the first is $\sum_{E \in \mathcal{H}_k} P_i(E|F,\mathcal{H}_i) \approx 1$ and the second is $\sum_{E \in \mathcal{H}_k} \delta_\omega(E) P_i(E|F,\mathcal{H}_i) \approx p_i(\omega|\mathcal{H}_i)$. In fact, due to the differences in generative capabilities of SMT models, training data selection and various pruning techniques used, search spaces of different systems are always not identical in practice. For the convenience of formal analysis, we treat all $P_i(E|F,\mathcal{H}_i)$ as ideal distributions with assumptions that all systems work in similar settings, and translation candidates are shared by all systems.

The method for computing *n*-gram posterior probability $p_i(\omega|\mathcal{H}_i)$ in Equation 5 depends on different types of search space $\mathcal{H}_i$:

- When $\mathcal{H}_i$ is an *N*-best list, it can be computed immediately by enumerating all translation candidates in the *N*-best list:

$$p_i(\omega|\mathcal{H}_i) = \sum_{E \in \mathcal{H}_i} \delta_\omega(E) P_i(E|F,\mathcal{H}_i).$$

- When $\mathcal{H}_i$ is a hypergraph (or a lattice) that encodes exponential number of hypotheses, it is often impractical to compute this probability directly. In this paper, we use the algorithm presented in Kumar *et al.* (2009) which is described in Algorithm 1[2]:

$$p_i(\omega|\mathcal{H}_i) = \sum_{E\in\mathcal{H}_i}\sum_{e\in E} f^*(e,\omega,\mathcal{H}_i)\, P_i(E|F,\mathcal{H}_i)$$
$$= \sum_{e\in\mathcal{E}} 1_e(\omega) f^*(e,\omega,\mathcal{H}_i) \sum_{E\in\mathcal{H}_i} 1_E(e) P_i(E|F,\mathcal{H}_i)$$
$$= \sum_{e\in\mathcal{E}} 1_e(\omega) f^*(e,\omega,\mathcal{H}_i) p_i(e|\mathcal{H}_i).$$

$f^*(e,\omega,\mathcal{H}_i)$ counts the edge e with n-gram ω that has the highest edge posterior probability relative to predecessors in the entire graph $\mathcal{H}_i$, and $p_i(e|\mathcal{H}_i)$ is the edge posterior probability that can be efficiently computed with standard inside and outside probabilities $I(v)$ and $O(v)$ as:

$$p_i(e|\mathcal{H}_i) = \frac{1}{Z(f)}\omega(e) O\big(h(e)\big) \prod_{v\in T(e)} I(v),$$

where $\omega(e)$ is the weight of hyperedge e in $\mathcal{H}_i$, $Z(f)$ is the normalization factor that equals to the inside probability of the root node in $\mathcal{H}_i$.

Algorithm 1: Compute n-gram posterior probabilities on hypergraph $\mathcal{H}_i$ (Kumar *et al.*, 2009)

```
sort hypernodes topologically
compute inside/outside probabilities $I(v)$ and $O(v)$ for each hypernode $v \in \mathcal{H}_i$
compute edge posterior probability $p_i(e|\mathcal{H}_i)$ for each hyperedge $e \in \mathcal{H}_i$
for each hyperedge $e \in \mathcal{H}_i$ do
    merge $n$-grams on $T(e)$ and keep the highest probability when $n$-grams are duplicated
    apply the rule of edge $e$ to $n$-grams on $T(e)$ and propagate $n-1$ gram prefixes/suffixes to $h(e)$
    for each $n$-gram $\omega$ introduced by $e$ do
        if $p_i(e|\mathcal{H}_i) > Max(\omega, T(e))$ then
            $p_i(\omega|\mathcal{H}_i) += p_i(e|\mathcal{H}_i) - Max(\omega, T(e))$
            $Max(\omega, h(e)) = p_i(e|\mathcal{H}_i)$
        else
            $Max(\omega, h(e)) = Max(\omega, T(e))$
        end if
    end for
end for
return $n$-gram posterior probability set $\{p_i(\omega|\mathcal{H}_i)\}_\omega$
```

Thus, the total gain score for hypothesis E' on $\mathcal{H} = \bigcup_i \mathcal{H}_i$ can be further expanded as:

$$\sum_i \lambda_i \sum_{k=1}^{K} \sum_{E\in\mathcal{H}_k} G(E,E')P_i(E|F,\mathcal{H}_i)$$
$$\approx \sum_i \lambda_i \sum_{k=1}^{K} \left\{\theta_0|E'| + \sum_\omega \theta_{|\omega|}\#_\omega(E')p_i(\omega|\mathcal{H}_i)\right\}$$
$$= \sum_i \lambda_i K \left\{\theta_0|E'| + \sum_\omega \theta_{|\omega|}\#_\omega(E')p_i(\omega|\mathcal{H}_i)\right\}$$
$$= K\left\{\sum_i \lambda_i \theta_0|E'| + \sum_\omega \theta_{|\omega|}\#_\omega(E') \sum_i \lambda_i p_i(\omega|\mathcal{H}_i)\right\}$$
$$= K\left\{\theta_0|E'| + \sum_\omega \theta_{|\omega|}\#_\omega(E')\mathscr{p}(\omega)\right\} \qquad (6)$$

where $\mathscr{p}(\omega) = \sum_i \lambda_i p_i(\omega|\mathcal{H}_i)$ is a mixture n-gram posterior probability. The most important fact derived from Equation 6 is that, the mixture of different distributions can be simplified to the weighted sum of n-gram posterior probabilities on different search spaces.

We now derive the decision rule of MMMBR decoding based on Equation 6 below:

$$\hat{E} = \operatorname*{argmax}_{E'\in\mathcal{H}} \theta_0|E'| + \sum_\omega \theta_{|\omega|}\#_\omega(E')\mathscr{p}(\omega). \qquad (7)$$

We also notice that MAP decoding and MBR decoding are two different ways of estimating the probability $P(E|F)$ and each of them has advantages and disadvantages. It is desirable to interpolate them together when choosing the final translation outputs. So we include each system's MAP decoding cost as an additional feature further and modify Equation 7 to:

$$\hat{E} = \operatorname*{argmax}_{E'\in\mathcal{H}} \theta_0|E'| + \sum_\omega \theta_{|\omega|}\#_\omega(E')\mathscr{p}(\omega)$$
$$+ \sum_k \theta_k \log C_{MAP}(E'|F,d_k), \qquad (8)$$

where $C_{MAP}(E'|F,d_k)$ is the model cost assigned by the MAP decoder d_k for hypothesis E'. Because the costs of MAP decoding on different SMT models are not directly comparable, we utilize the MERT algorithm to assign an appropriate weight θ_k for each component system.

Compared to single system-based MBR decoding, which obeys the decision rule below:

$$\hat{E} = \operatorname*{argmax}_{E'\in\mathcal{H}_k} \theta_0|E'| + \sum_\omega \theta_{|\omega|}\#_\omega(E')p(\omega|\mathcal{H}_k),$$

[2] We omit the similar algorithm for lattices because of their homogenous structures comparing to hypergraphs as we discussed in Section 2.2.

MMMBR decoding has a similar objective function (Equation 8). The key difference is that, in MMMBR decoding, n-gram posterior probability $\wp(\omega)$ is computed as $\sum_i \lambda_i\, p_i(\omega|\mathcal{H}_i)$ based on an ensemble of search spaces; meanwhile, in single system-based MBR decoding, this quantity is computed locally on single search space $\mathcal{H}_k$. The procedure of MMMBR decoding on multiple SMT systems is described in Algorithm 2.

Algorithm 2: MMMBR decoding on multiple SMT systems

```
for each component system $d_k$ do
    run MAP decoding and generate the corresponding search space $\mathcal{H}_k$
    compute the $n$-gram posterior probability set $\{p_k(\omega|\mathcal{H}_k)\}_\omega$ for $\mathcal{H}_k$ based on Algorithm 1
end for
compute the mixture $n$-gram posterior probability $\wp(\omega) = \sum_i \lambda_i\, p_i(\omega|\mathcal{H}_i)$ for each $\omega$:
for each unique $n$-gram $\omega$ appeared in $\bigcup_k \mathcal{H}_k$ do
    for each search space $\mathcal{H}_i$ do
        $\wp(\omega) += \lambda_i p_i(\omega|\mathcal{H}_i)$
    end for
end for
for each hyperedge $e$ in $\bigcup_k \mathcal{H}_k$ do
    assign $\wp(\omega)$ to the edge $e$ for all $\omega$ contained in $e$
end for
return the best path according to Equation 8
```

3 A Two-Pass Parameter Optimization

In Equation 8, there are two types of parameters: parameters introduced by the gain function $G(\cdot)$ and the model cost $C_{MAP}(\cdot)$, and system weights introduced by the mixture model $P(\cdot)$. Because Equation 8 is not a linear function when all parameters are taken into account, MERT algorithm (Och, 2003) cannot be directly applied to optimize them at the same time. Our solution is to employ a two-pass training strategy, in which we optimize parameters for MBR first and then system weights for the mixture model.

3.1 Parameter Optimization for MBR

The inputs of an MMMBR decoder can be a combination of translation search spaces with arbitrary structures. For the sake of a general and convenience solution for optimization, we utilize the simplest N-best lists with proper sizes as approximations to arbitrary search spaces to optimize MBR parameters using MERT in the first-pass training. System weights can be set empirically based on different performances, or equally without any bias. Note that although we tune MBR parameters on N-best lists, n-gram posterior probabilities used for Bayes risk computation could still be estimated on hypergraphs for non N-best-based search spaces.

3.2 Parameter Optimization for Mixture Model

After MBR parameters optimized, we begin to tune system weights for the mixture model in the second-pass training. We rewrite Equation 8 as:

$$\begin{aligned}\hat{E} = \underset{E' \in \mathcal{H}}{\operatorname{argmax}} \sum_i \lambda_i \{\theta_0 |E'| \\ + \sum_\omega \theta_{|\omega|} \#_\omega(E') p_i(\omega|\mathcal{H}_i) \\ + \sum_k \theta_k \log C_{MAP}(E'|F, d_k)\}. \quad (9)\end{aligned}$$

For each λ_i, the aggregated score surrounded with braces can be seen as its feature value. Equation 9 now turns to be a linear function for all weights and can be optimized by the MERT.

4 Experiments

4.1 Data and Metric

We conduct experiments on the NIST Chinese-to-English machine translation tasks. We use the newswire portion of the NIST 2006 test set (*MT06-nw*) as the development set for parameter optimization, and report results on the NIST 2008 test set (*MT08*). Translation performances are measured in terms of case-insensitive BLEU scores. Statistical significance is computed using the bootstrap re-sampling method proposed by Koehn (2004b). Table 1 gives data statistics.

Data Set	#Sentence	#Word
MT06-nw (dev)	616	17,316
MT08 (test)	1,357	31,600

Table 1. Statistics on dev and test data sets

All bilingual corpora available for the NIST 2008 constrained track of Chinese-to-English machine translation task are used as training data, which contain 5.1M sentence pairs, 128M Chinese words and 147M English words after preprocessing. Word alignments are performed by GIZA++ with an intersect-diag-grow refinement.

A 5-gram language model is trained on the English side of all bilingual data plus the Xinhua portion of LDC English Gigaword Version 3.0.

4.2 System Description

We use two baseline systems. The first one (*SYS1*) is a hierarchical phrase-based system (Chiang, 2007) based on Synchronous Context Free Grammar (SCFG), and the second one (*SYS2*) is a phrasal system (Xiong *et al.*, 2006) based on Bracketing Transduction Grammar (Wu, 1997) with a lexicalized reordering component based on maximum entropy model. Phrasal rules shared by both systems are extracted on all bilingual data, while hierarchical rules for SYS1 only are extracted on a selected data set, including LDC2003E07, LDC2003E14, LDC2005T06, LDC2005T10, LDC2005E83, LDC2006E26, LDC2006E34, LDC2006E85 and LDC2006E92, which contain about 498,000 sentence pairs. Translation hypergraphs are generated by each baseline system during the MAP decoding phase, and 1000-best lists used for MERT algorithm are extracted from hypergraphs by the k-best parsing algorithm (Huang and Chiang, 2005). We tune scaling factor to optimize the performance of HyperGraph-based MBR decoding (HGMBR) on MT06-nw for each system (0.5 for SYS1 and 0.01 for SYS2).

4.3 MMMBR Results on Multiple Systems

We first present the overall results of MMMBR decoding on two baseline systems.

To compare with single system-based MBR methods, we re-implement *N-best MBR*, which performs MBR decoding on 1000-best lists with the fast consensus decoding algorithm (DeNero *et al.*, 2009), and *HGMBR*, which performs MBR decoding on a hypergraph (Kumar *et al.*, 2009). Both methods use log-BLEU as the loss function. We also compare our method with *IHMM Word-Comb*, a state-of-the-art word-level system combination approach based on incremental HMM alignment proposed by Li *et al.* (2009b). We report results of MMMBR decoding on both *N*-best lists (*N-best MMMBR*) and hypergraphs (*Hypergraph MMMBR*) of two baseline systems. As MBR decoding can be used for any SMT system, we also evaluate *MBR-IHMM Word-Comb*, which uses *N*-best lists generated by HGMBR on each baseline systems. The default beam size is set to 50 for MAP decoding and hypergraph generation. The setting of *N*-best candidates used for (MBR-) IHMM Word-Comb is the same as the one used in Li *et al.* (2009b). The maximum order of n-grams involved in MBR model is set to 4. Table 2 shows the evaluation results.

	MT06-nw		MT08	
	SYS1	SYS2	SYS1	SYS2
MAP	38.1	37.1	28.5	28.0
N-best MBR	38.3	37.4	29.0	28.1
HGMBR	38.3	37.5	29.1	28.3
IHMM Word-Comb	39.1		29.3	
MBR-IHMM Word-Comb	39.3		29.7	
N-best MMMBR	39.0*		29.4*	
Hypergraph MMMBR	**39.4***$^+$		**29.9***$^+$	

Table 2. MMMBR decoding on multiple systems (*: significantly better than HGMBR with $p < 0.01$; +: significantly better than IHMM Word-Comb with $p < 0.05$)

From Table 2 we can see that, compared to MAP decoding, *N*-best MBR and HGMBR only improve the performance in a relative small range (+0.1~+0.6 BLEU), while MMMBR decoding on multiple systems can yield significant improvements on both dev set (+0.9 BLEU on *N*-best MMMBR and +1.3 BLEU on Hypergraph MMMBR) and test set (+0.9 BLEU on *N*-best MMMBR and +1.4 BLEU on Hypergraph MMMBR); compared to IHMM Word-Comb, *N*-best MMMBR can achieve comparable results on both dev and test sets, while Hypergraphs MMMBR can achieve even better results (+0.3 BLEU on dev and +0.6 BLEU on test); compared to MBR-IHMM Word-Comb, Hypergraph MMMBR can also obtain comparable results with tiny improvements (+0.1 BLEU on dev and +0.2 BLEU on test). However, MBR-IHMM Word-Comb has ability to generate new hypotheses, while Hypergraph MMMBR only chooses translations from original search spaces.

We next evaluate performances of MMMBR decoding on hypergraphs generated by different beam size settings, and compare them to (MBR-)

IHMM Word-Comb with the same candidate size and HGMBR with the same beam size. We list the results of MAP decoding for comparison. The comparative results on MT08 are shown in Figure 1, ***where X-axis is the size used for all methods each time***, ***Y-axis is the BLEU score***, MAP-i and HGMBR-i stand for MAP decoding and HGMBR decoding for the i^{th} system.

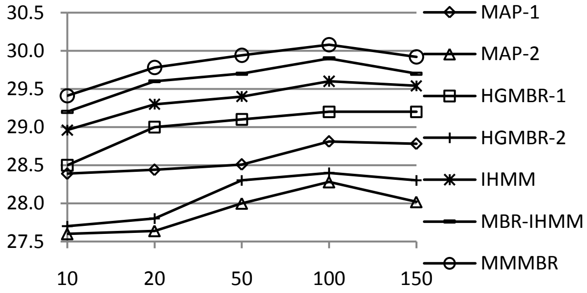


Figure 1. MMMBR vs. (MBR-) IHMM Word-Comb and HGMBR with different sizes

From Figure 1 we can see that, MMMBR decoding performs consistently better than both (MBR-) IHMM Word-Comb and HGMBR on all sizes. The gains achieved are around +0.5 BLEU compared to IHMM Word-Comb, +0.2 BLEU compared to MBR-IHMM Word-Comb, and +0.8 BLEU compared to HGMBR. Compared to MAP decoding, the best result (30.1) is obtained when the size is 100, and the largest improvement (+1.4 BLEU) is obtained when the size is 50. However, we did not observe significant improvement when the size is larger than 50.

We then setup an experiment to verify that the mixture model based on multiple distributions is more effective than any individual distributions for Bayes risk computation in MBR decoding. We use Mix-HGMBR to denote MBR decoding performed on single hypergraph of each system in the meantime using a mixture model upon distributions of two systems for Bayes risk computation. We compare it with HGMBR and Hypergraph MMMBR and list results in Table 3.

	MT08	
	SYS1	SYS2
HGMBR	29.1	28.3
Mix-HGMBR	29.4	28.9
Hypergraph MMMBR	**29.9**	

Table 3. Performance of MBR decoding on different settings of search spaces and distributions

It can be seen that based on the same search space, the performance of Mix-HGMBR is significantly better than that of HGMBR (+0.3/+0.6 BLEU on dev/test). Yet the performance is still not as good as Hypergraph, which indicates the fact that the mixture model and the combination of search spaces are both helpful to MBR decoding, and the best choice is to use them together.

We also empirically investigate the impacts of different system weight settings upon the performances of Hypergraph MMMBR on dev set in Figure 2, ***where X-axis is the weight $\boldsymbol{\lambda_1}$ for SYS1***, ***Y-axis is the BLEU score***. The weight λ_2 for SYS2 equals to $1 - \lambda_1$ as only two systems involved. The best evaluation result on dev set is achieved when the weight pair is set to 0.7/0.3 for SYS1/SYS2, which is also very close to the one trained automatically by the training strategy presented in Section 3.2. Although this training strategy can be processed repeatedly, the performance is stable after the 1st round finished.

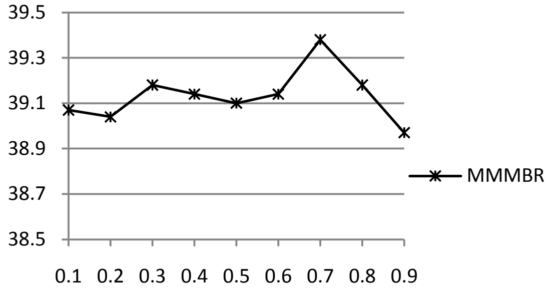


Figure 2. Impacts of different system weights in the mixture model

4.4 MMMBR Results on Identical Systems with Different Translation Models

Inspired by Macherey and Och (2007), we arrange a similar experiment to test MMMBR decoding for each baseline system on an ensemble of sub-systems built by the following two steps.

Firstly, we iteratively apply the following procedure 3 times: at the i^{th} time, we randomly sample 80% sentence pairs from the total bilingual data to train a translation model and use it to build a new system based on the same decoder, which is denoted as *sub-system-i*. Table 4 shows the evaluation results of all sub-systems on MT08, where MAP decoding (the former ones) and corresponding HGMBR (the latter ones) are grouped together by a slash. We set all beam sizes to 20 for a time-saving purpose.

	MT08	
	SYS1	SYS2
Baseline	28.4/29.0	27.6/27.8
sub-system-1	28.1/28.5	26.8/27.3
sub-system-2	28.3/28.4	27.0/27.1
sub-system-3	27.7/28.0	27.3/27.6

Table 4. Performance of sub-systems

Secondly, starting from each baseline system, we gradually add one more sub-system each time and perform Hypergraph MMMBR on hypergraphs generated by current involved systems. Table 5 shows the evaluation results.

	MT08	
	SYS1	SYS2
MAP	28.4	27.6
HGMBR	29.0	27.8
Hypergraph MMMBR		
+ sub-system-1	29.1	27.9
+ sub-system-2	29.1	28.1
+ sub-system-3	**29.3**	**28.3**

Table 5. Performance of Hypergraph MMMBR on multiple sub-systems

We can see from Table 5 that, compared to the results of MAP decoding, MMMBR decoding can achieve significant improvements when more than one sub-system are involved; however, compared to the results of HGMBR on baseline systems, there are few changes of performance when the number of sub-systems increases. One potential reason is that the translation hypotheses between multiple sub-systems under the same SMT model hold high degree of correlation, which is discussed in Macherey and Och (2007).

We also evaluate MBR-IHMM Word-Comb on *N*-best lists generated by each baseline system with its corresponding three sub-systems. Evaluation results are shown in Table 6, where Hypergraph MMMBR still outperforms MBR-IHMM Word-Comb on both baseline systems.

	MT08	
	SYS1	SYS2
MBR-IHMM Word-Comb	29.1	28.0
Hypergraph MMMBR	**29.3**	**28.3**

Table 6. Hypergraph MMMBR vs. MBR-IHMM Word-Comb with multiple sub-systems

5 Related Work

Employing consensus between multiple systems to improve machine translation quality has made rapid progress in recent years. System combination methods based on confusion networks (Rosti *et al.*, 2007; Li *et al.*, 2009b) have shown state-of-the-art performances in MT benchmarks. Different from them, MMMBR decoding method does not generate new translations. It maintains the essential of MBR methods to seek translations from existing search spaces. Hypothesis selection method (Hildebrand and Vogel, 2008) resembles more our method in making use of *n*-gram statistics. Yet their work does not belong to the MBR framework and treats all systems equally. Li *et al.* (2009a) presents a co-decoding method, in which *n*-gram agreement and disagreement statistics between translations of multiple decoders are employed to re-rank both full and partial hypotheses during decoding. Liu *et al.* (2009) proposes a joint-decoding method to combine multiple SMT models into one decoder and integrate translation hypergraphs generated by different models. Both of the last two methods work in a white-box way and need to implement a more complicated decoder to integrate multiple SMT models to work together; meanwhile our method can be conveniently used as a second-pass decoding procedure, without considering any system implementation details.

6 Conclusions and Future Work

In this paper, we have presented a novel MMMBR decoding approach that makes use of a mixture distribution of multiple SMT systems to improve translation accuracy. Compared to single system-based MBR decoding methods, our method can achieve significant improvements on both dev and test sets. What is more, MMMBR decoding approach also outperforms a state-of-the-art system combination method. We have empirically verified that the success of our method comes from both the mixture modeling of translation hypotheses and the combined search space for translation selection.

In the future, we will include more SMT systems with more complicated models into our MMMBR decoder and employ more general MERT algorithms on hypergraphs and lattices (Kumar *et al.*, 2009) for parameter optimization.

References

Chiang David. 2007. *Hierarchical Phrase Based Translation*. *Computational Linguistics*, 33(2): 201-228.

DeNero John, David Chiang, and Kevin Knight. 2009. *Fast Consensus Decoding over Translation Forests*. In *Proc. of 47th Meeting of the Association for Computational Linguistics*, pages 567-575.

Hildebrand Almut Silja and Stephan Vogel. 2008. *Combination of Machine Translation Systems via Hypothesis Selection from Combined N-best lists*. In *Proc. of the Association for Machine Translation in the Americas*, pages 254-261.

Huang Liang and David Chiang. 2005. *Better k-best Parsing*. In *Proc. of 7th International Conference on Parsing Technologies*, pages 53-64.

Huang Liang. 2008. *Forest Reranking: Discriminative Parsing with Non-Local Features*. In *Proc. of 46th Meeting of the Association for Computational Linguistics*, pages 586-594.

Koehn Philipp. 2004a. *Phrase-based Model for SMT*. *Computational Linguistics*, 28(1): 114-133.

Koehn Philipp. 2004b. *Statistical Significance Tests for Machine Translation Evaluation*. In *Proc. of Empirical Methods on Natural Language Processing*, pages 388-395.

Kumar Shankar and William Byrne. 2004. *Minimum Bayes-Risk Decoding for Statistical Machine Translation*. In *Proc. of the North American Chapter of the Association for Computational Linguistics*, pages 169-176.

Kumar Shankar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. *Efficient Minimum Error Rate Training and Minimum Bayes-Risk Decoding for Translation Hypergraphs and Lattices*. In *Proc. of 47th Meeting of the Association for Computational Linguistics*, pages 163-171.

Li Mu, Nan Duan, Dongdong Zhang, Chi-Ho Li, and Ming Zhou. 2009a. *Collaborative Decoding: Partial Hypothesis Re-Ranking Using Translation Consensus between Decoders*. In *Proc. of 47th Meeting of the Association for Computational Linguistics*, pages 585-592.

Liu Yang, Haitao Mi, Yang Feng, and Qun Liu. 2009. *Joint Decoding with Multiple Translation Models*. In *Proc. of 47th Meeting of the Association for Computational Linguistics*, pages 576-584.

Li Chi-Ho, Xiaodong He, Yupeng Liu, and Ning Xi. 2009b. *Incremental HMM Alignment for MT system Combination*. In *Proc. of 47th Meeting of the Association for Computational Linguistics*, pages 949-957.

Mi Haitao, Liang Huang, and Qun Liu. 2008. *Forest-Based Translation*. In *Proc. of 46th Meeting of the Association for Computational Linguistics*, pages 192-199.

Macherey Wolfgang and Franz Och. 2007. *An Empirical Study on Computing Consensus Translations from multiple Machine Translation Systems*. In *Proc. of Empirical Methods on Natural Language Processing*, pages 986-995.

Och Franz. 2003. *Minimum Error Rate Training in Statistical Machine Translation*. In *Proc. of 41th Meeting of the Association for Computational Linguistics*, pages 160-167.

Och Franz and Hermann Ney. 2004. *The Alignment template approach to Statistical Machine Translation*. *Computational Linguistics*, 30(4): 417-449.

Rosti Antti-Veikko, Spyros Matsoukas, and Richard Schwartz. 2007. *Improved Word-Level System Combination for Machine Translation*. In *Proc. of 45th Meeting of the Association for Computational Linguistics*, pages 312-319.

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. *Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation*. In *Proc. of Empirical Methods on Natural Language Processing*, pages 620-629.

Ueffing Nicola, Franz Och, and Hermann Ney. 2002. *Generation of Word Graphs in Statistical Machine Translation*. In *Proc. of Empirical Methods on Natural Language Processing*, pages 156-163.

Wu Dekai. 1997. *Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora*. *Computational Linguistics*, 23(3): 377-404.

Xiong Deyi, Qun Liu, and Shouxun Lin. 2006. *Maximum Entropy based Phrase Reordering Model for Statistical Machine Translation*. In *Proc. of 44th Meeting of the Association for Computational Linguistics*, pages 521-528.

Multi-Sentence Compression: Finding Shortest Paths in Word Graphs

Katja Filippova
Google Inc.
katjaf@google.com

Abstract

We consider the task of summarizing a cluster of related sentences with a short sentence which we call *multi-sentence compression* and present a simple approach based on shortest paths in word graphs. The advantage and the novelty of the proposed method is that it is syntax-lean and requires little more than a tokenizer and a tagger. Despite its simplicity, it is capable of generating grammatical and informative summaries as our experiments with English and Spanish data demonstrate.

1 Introduction

Sentence compression (henceforth SC) is a task where the goal is to produce a summary of a single sentence which would preserve the important part of the content and be grammatical. Starting from the early work of Jing & McKeown (2000), in the last decade SC has received considerable attention in the NLP community. Ubiquitous use of mobile devices is an obvious example of where SC could be applied–a longer text of an email, news or a Wikipedia article can be compressed sentence by sentence to fit into a limited display (Corston-Oliver, 2001). Another reason why SC is so popular is its potential utility for extractive text summarization, single or multi-document (Mani, 2001). There, a standard approach is to rank sentences by importance, cluster them by similarity, and select a sentence from the top ranked clusters. Selected sentences almost always require revision and can be reformulated succinctly as it is often only a part of the sentence which is of interest. It is this multi-document summarization scenario which motivates our work.

Given a cluster of similar, or related, sentences, we aim at summarizing the most salient theme of it in a short single sentence. We refer to this task as *multi-sentence compression*. Defined this way, it comes close to sentence fusion which was originally introduced as a text-to-text generation technique of expressing content common to most of the input sentences in a single sentence (Barzilay & McKeown, 2005). However, since then the technique has been extended so that now fusion also stands for uniting complementary content in a single concise sentence (Filippova & Strube, 2008b; Krahmer et al., 2008). Since our method is not designed for the "union" kind of fusion, we think it is more appropriate to classify it as a sentence compression technique.

Two challenges of SC as well as text summarization are (i) important content selection and (ii) its readable presentation. Most existing systems use syntactic information to generate grammatical compressions. Incidentally, syntax also provides clues to what is likely to be important–e.g., the subject and the verb of the main clause are more likely to be important than a prepositional phrase or a verb from a relative clause. Of course, syntax is not the only way to gauge word or phrase importance. In the case of sentence compression being used for text summarization, one disposes of a rich context to identify important words or phrases. For example, recurring or semantically

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 322–330,
Beijing, August 2010

similar words are likely to be relevant, and this information has been used in earlier SC systems (Hori et al., 2003; Clarke & Lapata, 2007, inter alia). Still, syntactic parsers are assumed to be indispensable tools for both sentence compression and fusion because syntactic constraints (hand-crafted or learned from the data) seem to be the only way to control the grammaticality of the output. In this paper we are going to question this well-established belief and argue that just like in some cases syntax helps to find important content (e.g., when the input is an isolated sentence), in the multi-sentence case redundancy provides a reliable way of generating grammatical sentences. In particular, the important and novel points of our work are as follows:

- We present a simple and robust word graph-based method of generating succinct compressions which requires as little as a part of speech tagger and a list of stopwords.
- To our knowledge, it is the first method which requires neither a parser, nor hand-crafted rules, nor a language model to generate reasonably grammatical output.
- In an extensive evaluation with native speakers we obtain encouraging results for English as well as for Spanish.

In the following section we present our approach to sentence compression (Sec. 2); then we introduce the baseline (Sec. 3) and the data (Sec. 4). In Section 5 we report about our experiments and discuss the results. Finally, Section 6 gives an overview of related work.

2 Multi-sentence Compression

A well-known challenge for extractive multi-document summarization systems is to produce non-redundant summaries. There are two standard ways of avoiding redundancy: either one adds sentences to the summary one-by-one and each time checks whether the sentence is significantly different from what is already there (e.g., using MMR), or one clusters related sentences and selects only one from each cluster. In both cases a selected sentence may include irrelevant information, so one wishes to compress it, usually by taking syntactic and lexical factors into account. However, we think this approach is suboptimal in this case and explore a different way. Instead of compressing a single sentence, we build a *word graph* from all the words of the related sentences and compress this graph.

A word graph is a directed graph where an edge from word *A* to word *B* represents an *adjacency* relation. It also contains the *start* and *end* nodes. Word graphs have been widely used in natural language processing for building language models, paraphrasing, alignment, etc. (see Sec. 6). Compared with dependency graphs, their use for sentence generation has been left largely unexplored, presumably because it seems that almost all the grammatical information is missing from this representation. Indeed, a link between a finite verb and an article does not correspond to any grammatical relation between the two. However, the premise for our work is that redundancy should be sufficient to identify not only important words but also salient links between words. In this section we present our approach to word graph compression. We begin by explaining the graph construction process and continue with the details of two compression methods.

2.1 Word Graph Construction

Given a set of related sentences $S = \{s_1, s_2, ...s_n\}$, we build a word graph by iteratively adding sentences to it. As an illustration, consider the four sentences below and the graph in Figure 1 obtained from them. Edge weights are omitted and italicized fragments from the sentences are replaced with dots for clarity.

(1) *The wife of a former U.S. president Bill* Clinton Hillary Clinton visited China last Monday.

(2) Hillary Clinton wanted to visit China last month *but postponed her plans* till Monday last week.

(3) Hillary Clinton paid *a visit to the People Republic* of China on Monday.

(4) Last week the *Secretary of State* Ms. Clinton visited Chinese officials.

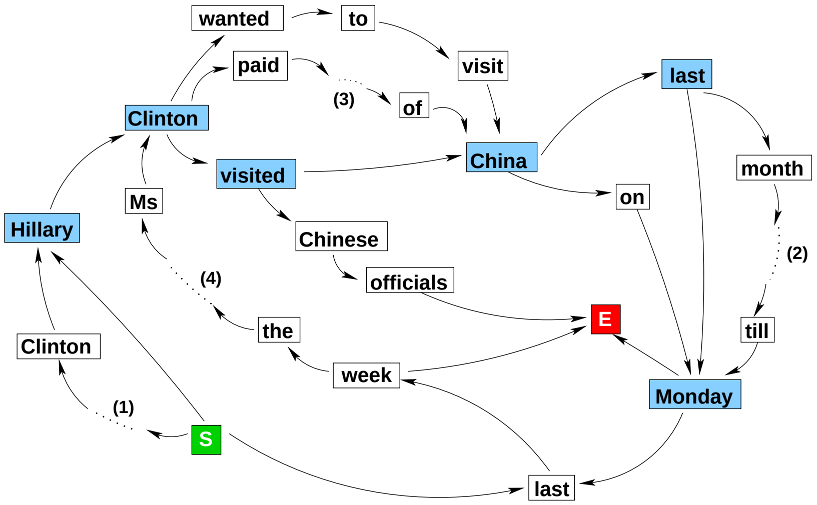


Figure 1: Word graph generated from sentences (1-4) and a possible compression path.

After the first sentence is added the graph is simply a string of word nodes (punctuation is excluded) plus the start and the end symbols (*S* and *E* in Fig. 1). A word from the following sentences is mapped onto a node in the graph provided that they have the exact same lowercased word form and the same part of speech[1] and that no word from this sentence has already been mapped onto this node. Using part of speech information reduces chances of merging verbs with nouns (e.g., *visit*) and generating ungrammatical sequences. If there is no candidate in the graph a new node is created.

Word mapping/creation is done in three steps for the following three groups of words: (1) non-stopwords[2] for which no candidate exists in the graph or for which an unambiguous mapping is possible; (2) non-stopwords for which there are either several possible candidates in the graph or which occur more than once in the sentence; (3) stopwords.

This procedure is similar to the one used by Barzilay & Lee (2003) in that we also first identify "backbone nodes" (unambiguous alignments) and then add mappings for which several possibilities exist. However, they build lattices, i.e., directed acyclic graphs, whereas our graphs may contain cycles. For the last two groups of words where mapping is ambiguous we check the immediate context (the preceding and following words in the sentence and the neighboring nodes in the graph) and select the candidate which has larger overlap in the context, or the one with a greater frequency (i.e., the one which has more words mapped onto it). For example, in Figure 1 when sentence (4) is to be added, there are two candidate nodes for *last*. The one pointing to *week* is selected as *week* is the word following *last* in (4). Stopwords are mapped only if there is some overlap in non-stopword neighbors, otherwise a new node is created.

Once all the words from the sentence are in place, we connect words adjacent in the sentence with directed edges. For newly created nodes, or nodes which were not connected before, we add an edge with a default weight of one. Edge weights between already connected nodes are increased by one. The same is done with the start and end nodes. Nodes store id's of the sentences their words come from as well as all their offset positions in those sentences.

The described alignment method is fairly simple and guarantees the following properties of the word graph: (i) every input sentence corresponds to a loopless path in the graph; (ii) words referring to the same entities or actions are likely to end up in one node; (iii) stopwords are only joined

[1] We use the OpenNLP package for tagging: `http://opennlp.sourceforge.net`.

[2] We generate a list of about 600 news-specific stopwords for English (including, e.g., *said*, *seems*) and took a publicly available list of about 180 stopwords for Spanish from `www.ranks.nl/stopwords/spanish.html`.

in one node if there is an overlap in context. The graph may generate a potentially endless amount of incomprehensible sequences connecting *start* and *end*. It is also likely to contain paths corresponding to good compressions, like the path connecting the nodes highlighted with blue in Figure 1. In the following we describe two our methods of finding the best path, that is, the best compression for the input sentences.

2.2 Shortest Path as Compression

What properties are characteristic of a good compression? It should neither be too long, nor too short. It should go through the nodes which represent important concepts but should not pass the same node several times. It should correspond to a likely word sequence. To satisfy these constraints we invert edge weights, i.e., link frequencies, and search for the shortest path (i.e., lightest in terms of the edge weights) from *start* to *end* of a predefined minimum length. This path is likely to mention salient words from the input and put together words found next to each other in many sentences. This is the first method we consider. We set a minimum path length (in words) to eight which appeared to be a reasonable threshold on a development set–paths shorter than seven words were often incomplete sentences.

Furthermore, to produce *informative* summaries which report about the main event of the sentence cluster, we filter paths which do not contain a verb node. For example, *Ozark's "Winter's Bone" at the 2010 Sundance Film Festival* might be a good title indicating what the article is about. However, it is not as informative as *"Winter's Bone" earned the grand jury prize at Sundance* which indeed conveys the gist of the event. Thus, we generate *K* shortest paths and filter all those which are shorter than eight words or do not contain a verb. The path with the minimum total weight is selected as the summary.

2.3 Improved Scoring and Reranking

The second configuration of our system employs a more sophisticated weighting function. The purpose of this function is two-fold: (i) to generate a grammatical compression, it favors strong links, i.e., links between words which appear significantly often in this order; (ii) to generate an informative compression, it promotes paths passing through salient nodes.

Strong links: Intuitively, we want the compression path to follow edges between words which are strongly associated with each other. Inverted edge frequency is not sufficient for that because it ignores the overall frequency of the nodes the edge connects. For example, edge frequency of three should count more if the edge connects two nodes with frequency of three rather than if their frequencies are much higher. Thus, we redefine edge weight as follows:

$$w(e_{i,j}) = \frac{\mathtt{freq}(i) + \mathtt{freq}(j)}{\mathtt{freq}(e_{i,j})} \tag{1}$$

Furthermore, we also promote a connection between two nodes if there are multiple paths between them. For example, if some sentences speak of *president Barack Obama* or *president of the US Barack Obama*, and some sentences are about *president Obama*, we want to add some reward to the edge between *president* and *Obama*. However, longer paths between words are weak signals of word association. Therefore, the weight of an edge between the nodes i and j is reduced for every possible path between them but reduced proportionally to its length:

$$w'(e_{i,j}) = \frac{\mathtt{freq}(i) + \mathtt{freq}(j)}{\sum_{s \in S} \mathtt{diff}(s,i,j)^{-1}} \tag{2}$$

where the function $\mathtt{diff}(s,i,j)$ refers to the distance between the offset positions ($\mathtt{pos}(s,i)$) of words i and j in sentence s and is defined as follows:

$$\mathtt{diff}(s,i,j) = \begin{cases} \mathtt{pos}(s,i) - \mathtt{pos}(s,j) & \text{if } \ \mathtt{pos}(s,i) < \mathtt{pos}(s,j) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Salient words: The function above only indicates how strong the association between two words is. It assigns equal weights to edges connecting words encountered in a single sentence and words encountered next to each other in every sentence. To generate a summary concerning the most salient events and entities, we force the path to go through most frequent nodes by decreasing edge weight with respect to the frequency of the nodes it connects. Thus, we further redefine edge weight as follows:

$$w''(e_{i,j}) = \frac{w'(e_{i,j})}{\texttt{freq}(i) \times \texttt{freq}(j)} \qquad (4)$$

We implement the K-shortest paths algorithm to find the fifty shortest paths from *start* to *end* using the weighting function in (4). We filter all the paths which are shorter than eight words and which do not pass a verb node. Finally, we rerank the remaining paths by normalizing the total path weight over its length. This way we obtain the path which has the lightest average edge weight.

3 Baseline

As a first baseline we are searching for the most probable string with respect to the sentence cluster. In particular, we use the Viterbi algorithm to find the sequence of words of a predefined length n which maximizes the bigram probability (MLE-based):

$$p(w_{1,n}) = p(w_1|\mathsf{s})p(w_2|w_1)...p(\mathsf{e}|w_n) \qquad (5)$$

Similar to the shortest path implementation, we specify compression length and set it also here to eight tokens. However, the compressions obtained with this method are often unrelated to the main theme. The reason for that is that a token subsequence encountered in a single sentence is likely to get a high probability–all transition probabilities are equal to one–provided that the probability of entering this sequence is not too low. To amend this problem and to promote frequent words (i.e., words which are likely to be related to the main theme) we maximize the following baseline score which takes into account both the bigram probabilities and the token likelihood, $p(w_i)$, which is also estimated from the sentence cluster:

$$b(w_{1,n}) = p(w_1|\mathsf{s})p(w_2|w_1)...p(\mathsf{e}|w_n) \prod_i p(w_i) \qquad (6)$$

4 Data Sources

As data for our experiments we use news articles presented in clusters on Google News[3]. The main reason for why we decided to use this service is that it is freely available and does the job of news classification and clustering with a production quality. Apart from that, it is a rich source of multilingual data.

We collected news clusters in English and Spanish, 10-30 articles each, 24 articles on average. To get sets of similar sentences we aggregated first sentences from every article in the cluster, removing duplicates. The article-initial sentence is known to provide a good summary of the article and has become a standard competitive baseline in summarization[4]. Hence, given that first sentences summarize the articles they belong to, which are in turn clustered as concerning the same event, those sentences are likely although not necessarily need to be similar.

From the total of 150 English clusters we reserved 70 for development and 80 for testing. For Spanish we collected 40 clusters, all for testing. We stripped off bylines and dates from the beginning of every sentence with a handful of regular expressions before feeding them to the baseline and our compression methods.

The data we use has two interesting properties: (i) article-initial sentences are on average longer than other sentences. In our case average sentence lengths for English and Spanish (without bylines) are 28 and 35 tokens, respectively. (ii) such sentence clusters are noisier than what one would expect in a summarization pipeline. Both properties make the task realistically hard and pose a challenge for the robustness of a compression method. If we show that reasonable compressions can be generated even from noisy clusters acquired from a publicly available news service, then we have a good reason to believe that the method will perform at least comparable on more carefully constructed clusters of shorter sentences.

[3] `http://news.google.com`

[4] See DUC/TAC competitions: `http://www.nist.gov/tac`

5 Evaluation

5.1 Experiment Design

The performance of the systems was assessed in an experiment with human raters, all native speakers. They were presented with a list of snippets of the articles from one cluster – first sentence and title linked to the original document. The raters were allowed to look up the articles if they need more background on the matter but this was not obligatory.

The first question concerned the quality of the sentence cluster. The raters were asked whether the cluster contained a single prevailing event, or whether it was too noisy and no theme stood out. Given how simple our sentence grouping procedure was, most clusters informed about more than one event. However, to answer the question positively it would be enough to identify one prevailing theme.

Below that, a summary and two further questions concerning its quality were displayed. Similar to most preceding work, we were interested in grammaticality and informativity of summaries. With respect to grammaticality, following Barzilay & McKeown (2005), we asked the raters to give one of the three possible ratings: *perfect* if the summary was a complete grammatical sentence (2 pts); *almost* if it required a minor editing, e.g., one mistake in articles or agreement (1 pt); *ungrammatical* if it was none of above (0 pts). We explicitly asked the raters to ignore lack or excess of capitalization or punctuation. Furthermore, based on the feedback from a preliminary evaluation, we provided an example in which we made clear that summaries consisting of a few phrases which cannot be reformulated as a complete sentence (e.g., *Early Monday a U.S. Navy ship.*) should not count as grammatical.

The final question, concerning informativity, had four possible options: *n/a* if the cluster is too noisy and unsummarizable in the first place; *perfect* if it conveys the gist of the main event and is more or less like the summary the person would produce himself (2 pts); *related* if it is related to the the main theme but misses something important (1 pt); *unrelated* if the summary is not related to the main theme (0 pts).

For each of the 80 sentence clusters (40 for Spanish) we generated three summaries with the three systems. Most summaries were rated by four raters, a few got only three ratings; no rater saw the same cluster twice.

5.2 Results

We report average grammaticality and informativity scores in Table 1. However, averaging system ratings over all clusters and raters is not justified in our case. It is important to remember that the score assignments (i.e., *0, 1, 2*) are arbitrary and that the score of one with respect to grammaticality (i.e., a minor mistake) is in fact closer to two than to zero. One could set the scores differently but even then, strictly speaking, it is not correct to average the scores as ratings do not define a metric space.

System	Gram	Info
Baseline	0.70 / 0.61	0.62 / 0.53
Shortest path	1.30 / 1.27	1.16 / 0.79
Shortest path++	1.44 / 1.25	1.30 / 1.25

Table 1: Average ratings for English / Spanish.

Therefore in Table 2 we present distributions over the three scores for both grammaticality and informativity together with average summary lengths in tokens. For both grammaticality and informativity, for every summary-cluster pair we did majority voting and resolved ties by assigning the lower score. For example, if a system got the ratings *1, 1, 2, 2* for a certain cluster, we counted this as *1*. We dismissed cases where the tie was between the maximum and the minimum score–this happened with some summaries which got just three scores (i.e., *0, 1, 2*) and accounted for $<$ 4% of the cases. To obtain the informativity distribution we considered only clusters which were classified as containing a single prevailing event by at least ten raters. For English 75 out of 80 clusters qualified as such (37 out of 40 for Spanish). Similar to above, we dismissed about 3% tie cases where the ratings diverged significantly (e.g., *0, 1, 2*).

System	Gram-2	Gram-1	Gram-0	Info-2	Info-1	Info-0	Avg. Len.
Baseline (EN)	21%	15%	65%	18%	10%	73%	8
Shortest path (EN)	52%	16%	32%	36%	33%	31%	10
Shortest path++ (EN)	64%	13%	23%	52%	32%	16%	12
Baseline (ES)	12%	15%	74%	9%	19%	72%	8
Shortest path (ES)	58%	21%	21%	23%	26%	51%	10
Shortest path++ (ES)	50%	21%	29%	40%	40%	20%	12

Table 2: Distribution over possible ratings and average length for English and Spanish.

5.3 Discussion

The difference between the baseline and our shortest path systems is striking. Although more than 20% of the baseline summaries are perfectly grammatical, the gap to the improved version of shortest paths is significant, about 43%. The same holds for the percentage of informative summaries (18% vs. 52%). Both numbers are likely to be understated as we chose to resolve all ties not in our favor. 84% of the summaries generated by the improved method are related to the main theme of the cluster, and more than 60% of those (52% of the total summaries) convey the very gist of it without missing any important information. Comparing the two configurations we have proposed, improved scoring function and reranking we added on top of the shortest path method were both rewarding. Interestingly, even the straightforward approach of choosing the shortest path of a minimum length already guarantees a grammatical summary in more than half of the cases.

An interesting difference in the performance for Spanish and English is that shortest path generates more grammatical sentences than the improved version of it. However, the price for higher grammaticality scores is a huge drop in informativity: half of such summaries are not related to the main theme at all, whereas 40% of the summaries generated by the improved version got the highest rating. A possible reason for the poorer performance for Spanish is that we used a much smaller list of stopwords which did not include news-specific words like, e.g., *dijo* (*said*) which resulted in denser graphs. In the future, we would like to apply the method to more languages and experiment with longer lists of stopwords.

One may notice that the summaries produced by the baseline are shorter than those generated by the shortest paths which might look like a reason for its comparatively poor performance. However, the main source of errors for the baseline was its inability to keep track of the words already present in the summary, so it is unlikely that longer sequences would be of a much higher quality. The sentences generated by the baseline were often repetitive, e.g., *The food tax on food tax on food*. This is not an issue with the shortest path approaches as they never include loops when edge weights are strictly positive.

The reranking we added to the shortest path method is the reason for why the summaries generated by the improved version of the system are on average slightly longer than those produced by the simpler version. The average lengths for both systems are drastically shorter than the average length of the sentences served as input (10/12 vs. 28 tokens in English or 35 tokens for Spanish). This corresponds to the compression rate of 36-43% (29-34% for Spanish) which is comparatively “aggressive” as it usually varies between 50-80% in other systems.

6 Comparison with Related Work

6.1 Sentence Compression

In the last ten years a lot of research has been devoted to sentence compression. Most studies share two properties: (1) they rely on syntax, and (2) they are supervised. The degree of syntax-dependence varies between methods. Some utilize a parser to identify and later keep certain important relations but do not require a complete parse (Clarke & Lapata, 2008), or use a syntactic representation to extract features (McDonald, 2006). For other approaches correct syntactic trees are crucial to obtain grammatical compressions (Galley & McKeown, 2007; Filippova & Strube, 2008a; Cohn & Lapata, 2009). Handcrafted rules (Dorr et al., 2003) as well as language models also have been utilized to generate fluent compressions (Hori et al., 2003; Clarke & Lapata, 2008).

6.2 Sentence Generation

To date the work on sentence fusion is completely dependency syntax-based. Input sentences are parsed into trees, from those trees a new dependency structure is generated, and this structure is finally converted into a sentence (Barzilay & McKeown, 2005; Filippova & Strube, 2008b; Wan et al., 2009). Parser quality is of crucial importance for such methods, and to our knowledge no attempt has been made to generate novel sentences without adhering to dependency representations. In the future, it would be of interest to compare our method with a syntax-based fusion method. Syntax-lean methods have been explored for headline generation (Banko et al., 2000; Dorr et al., 2003; Jin & Hauptmann, 2003). However, they do not aim at generating complete sentences or informative summaries but rather to indicate what the news is about.

6.3 Word Graphs and Lattices

Perhaps the work of Barzilay & Lee (2003) who align comparable sentences to generate sentence-level paraphrases seems closest to ours in that we both use word graphs for text generation. However, this is a fairly general similarity, as both the goal and the implementation are different. While we search for an optimal weighting function in noisy graphs to identify readable and informative compressions, they induce paraphrase patterns from unweighted paths in much smaller DAGs obtained from highly similar sentences. Shen et al. (2006) is another example of using word lattices to find paraphrases. Unlike Barzilay & Lee (2003), they propose to use syntax to obtain accurate alignments. Numerous examples of the utility of word lattices come from the field of finite state automata, language modeling, speech recognition, parsing and machine translation (Mohri, 1997, inter alia).

7 Conclusions

We considered the task of generating a short informative summary for a set of related sentences, called multi-sentence compression, which arises naturally in the context of multi-document text summarization. We presented a simple but robust method which proceeds by finding shortest paths in word graphs. The novelty of our work is that we demonstrated that reasonable compressions can be obtained without any syntactic information if a good weighting function is defined. This distinguishes our work from earlier research on sentence fusion and compression which relies on syntactic representations and/or language models. We provided the details of an extensive evaluation on English and Spanish data and reported high grammaticality as well as informativity scores. In the future we would like to experiment with other languages and eschew using part-of-speech information.

Acknowledgements: I am thankful to Keith Hall for the discussions on this work and the very helpful feedback on an earlier draft of this paper.

References

Banko, M., V. O. Mittal & M. J. Witbrock (2000). Headline generation based on statistical translation. In *Proc. of ACL-00*, pp. 318–325.

Barzilay, R. & L. Lee (2003). Learning to paraphrase: An unsupervized approach using multi-sequence alignment. In *Proc. of HLT-NAACL-03*, pp. 16–23.

Barzilay, R. & K. R. McKeown (2005). Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–327.

Clarke, J. & M. Lapata (2007). Modelling compression with discourse constraints. In *Proc. of EMNLP-CoNLL-07*, pp. 1–11.

Clarke, J. & M. Lapata (2008). Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research*, 31:399–429.

Cohn, T. & M. Lapata (2009). Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34:637–674.

Corston-Oliver, S. H. (2001). Text compaction for display on very small screens. In *Proceedings of the NAACL Workshop on Automatic Summarization*, Pittsburg, PA, 3 June 2001, pp. 89–98.

Dorr, B., D. Zajic & R. Schwartz (2003). Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the Text Summarization Workshop at HLT-NAACL-03*, Edmonton, Alberta, Canada, 2003, pp. 1–8.

Filippova, K. & M. Strube (2008a). Dependency tree based sentence compression. In *Proc. of INLG-08*, pp. 25–32.

Filippova, K. & M. Strube (2008b). Sentence fusion via dependency graph compression. In *Proc. of EMNLP-08*, pp. 177–185.

Galley, M. & K. R. McKeown (2007). Lexicalized Markov grammars for sentence compression. In *Proc. of NAACL-HLT-07*, pp. 180–187.

Hori, C., S. Furui, R. Malkin, H. Yu & A. Waibel (2003). A statistical approach to automatic speech summarization. *EURASIP Journal on Applied Signal Processing*, 2:128–139.

Jin, R. & A. G. Hauptmann (2003). Automatic title generation for spoken broadcast news. In *Proc. of HLT-01*, pp. 1–3.

Jing, H. & K. McKeown (2000). Cut and paste based text summarization. In *Proc. of NAACL-00*, pp. 178–185.

Krahmer, E., E. Marsi & P. van Pelt (2008). Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion. In *Proc. of ACL-HLT-08*, pp. 193–196.

Mani, I. (2001). *Automatic Summarization.* Amsterdam, Philadelphia: John Benjamins.

McDonald, R. (2006). Discriminative sentence compression with soft syntactic evidence. In *Proc. of EACL-06*, pp. 297–304.

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Shen, S., D. Radev, A. Patel & G. Erkan (2006). Adding syntax to dynamic programming for aligning comparable texts for generation of paraphrases. In *Proc. of COLING-ACL-06*, pp. 747–754.

Wan, S., M. Dras, R. Dale & C. Paris (2009). Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proc. of EACL-09*, pp. 852–860.

Enriching Dictionaries with Images from the Internet - Targeting Wikipedia and a Japanese Semantic Lexicon: Lexeed -

Sanae Fujita
NTT Communication Science Lab.
sanae@cslab.kecl.ntt.co.jp

Masaaki Nagata
NTT Communication Science Lab.
nagata.masaaki@lab.ntt.co.jp

Abstract

We propose a simple but effective method for enriching dictionary definitions with images based on image searches. Various query expansion methods using synonyms/hypernyms (or related words) are evaluated. We demonstrate that our method is effective in obtaining high-precision images that complement dictionary entries, even for words with abstract or multiple meanings.

1 Introduction

The Internet is an immense resource for images. If we can form connections between these images and dictionary definitions, we can create rich dictionary resources with multimedia information. Such dictionaries have the potential to provide educational (Popescu et al., 2006), cross-langauge information retrieval (Hayashi et al., 2009) or assistive communication tools especially for children, language learners, speakers of different languages, and people with disabilities such as dyslexia (Mihalcea and Leong, 2008; Goldberg et al., 2009).

Additionally, a database of typical images connected to meanings has the potential to fill the gaps between images and meanings (semantic gap). There are many studies which aim to cross the semantic gap (Ide and Yanai, 2009; Smeulders et al., 2000; Barnard et al., 2003) from the point of view of image recognition. However the semantic classes of target images are limited (e.g. Caltech-101, 256[1]). Yansong and Lapata (2008) tried to construct image databases annotated with keywords from Web news images with their captions and articles, though the semantic coverage is unknown. In this paper, we aim to supply several suitable images for dictionary definitions. We propose a simple but effective method based on an Internet image search.

There have been several studies related to supplying images for a dictionary or thesaurus. Bond et al. (2009) applied images obtained from the Open Clip Art Library (**OCAL**) to Japanese **WordNet**.[2] They obtained candidate images by comparing the hierarchical structures of **OCAL** and **WordNet**, and then judged whether or not the image was suitable for the synset by hand. **OCAL** benefits from being in the public domain; however, it cannot cover a wide variety of meanings because of the limited number of available images.

Fujii and Ishikawa (2005) collected images and text from the Internet by querying lemma, and linked them to an open encyclopedia, **CYCLONE**.[3] They guessed the meaning of the images by disambiguating the surrounding text. This is a straightforward approach, but it is difficult to use it to collect images with minor meanings, because in most cases the Internet search querying lemma only provides images related to the most common meaning. For example, lemma アーチ *arch* may mean ``architecture'' or ``home run'' in Japanese, but a lemma search provided no image of the latter at least in the top 500.

There are some resources which link images to target synsets selected from **WordNet** (Fellbaum, 1998). For example, **PicNet** (Borman et al., 2005), **ImageNet** (Deng et al., 2009) and image ontology (Popescu et al., 2006, 2007; Zinger et al., 2006) collect candidate images from the Internet. **PicNet** and **ImageNet** ask Web users to judge their suitability, and Zinger et al. (2006); Popescu et al. (2007) automatically filtered out unsuitable images using visual characteristics. These approaches can

[1] http://www.vision.caltech.edu/Image_Datasets/Caltech101, 256/

[2] http://nlpwww.nict.go.jp/wn-ja/

[3] http://cyclone.cl.cs.titech.ac.jp/

INDEX		アーチ *arch* (POS: noun)	
SENSE 1	DEFINITION	上部$_1$ を 弓$_1$ の 形$_1$ に した 建物$_1$ 。 また$_9$ 、 その$_3$ 建築$_1$ 様式$_2$ *Buildings with bow-shaped top. Or its architectural style.*	IMAGE
	EXAMPLE	あの$_2$ 橋$_1$ は 2 つ の アーチ$_1$ で 出来$_4$ て いる 。 *That bridge has 2 arches.*	
	HYPERNYM	建物$_1$ *building*, 様式$_2$ *style*	
	SEM. CLASS	⟨`865:house (main building)`⟩ (⊂ ⟨`2:concrete`⟩), ⟨`2435:pattern, method`⟩ (⊂ ⟨`1000:abstract`⟩)	
SENSE 3	DEFINITION	野球$_1$ で 、 本塁打$_1$ 。 ホームラン$_1$ 。 *A home run in baseball.*	IMAGE
	EXAMPLE	バッター$_1$ が ライト$_4$ スタンド$_2$ に 逆転$_3$ の アーチ$_3$ を 放っ$_4$ た *A batter blasted the ball over the right-field wall.*	
	HYPERNYM	本塁打$_1$ *honruida*	
	SYNONYM	ホームラン$_1$ *home run*, DOMAIN 野球$_1$ *baseball*	
	SEM. CLASS	⟨`1680:sport`⟩ (⊂ ⟨`1000:abstract`⟩)	

Figure 1: Simplified Entry for **Lexeed** & **Hinoki**:アーチ *arch*

collect a large number of highly accurate images. However, target synsets are limited at present, and the coverage of polysemous words is unknown. We present a comparison with **ImageNet** and image ontology (Popescu et al., 2006) in § 3.

In this paper, to cover a broad range of meanings, we use an Internet search. In advance, we expand the number of queries per meaning using information extracted from definition sentences. In § 3, we investigate the usability and effectiveness of several types of information targeting two different types of dictionaries, a Japanese Semantic Lexicon: **Lexeed** and a Web Dictionary: Japanese **Wikipedia**[4] (§ 2). We show that our method is simple but effective. We also analyze senses that are difficult to portray using images.

2 Resources

2.1 Japanese Semantic Lexicon: Lexeed

We use **Lexeed**, a Japanese Semantic Lexicon (Kasahara et al., 2004) as a target dictionary (see Figure 1). **Lexeed** includes the 29,000 most familiar words in Japanese, split into 48,000 senses. Each entry contains the word itself and its part of speech (POS) along with definition and example sentences and links to the Goi-Taikei (**GT**) Japanese Ontology (Ikehara et al., 1997). In addition, we extracted related words such as hypernyms, synonyms, and domains, from the definitions (called **Hinoki** Ontology). The images in Figure 1 are samples provided using our method.

Table 1: Size of **Lexeed** and Japanese **Wikipedia** (disambiguation)

No.	**Lexeed**	**Wikipedia**	Shared Lemma
Entries	29,272	33,299	2,228
Senses	48,009	197,912[1]	19,703
Ave. Senses/Entry	1.6	5.9	8.8
Max. Senses/Entry	57	320	148
Monosemous	19,080	74	2
Ave. Words/Definition[2]	14.4	10.7	11.0

[1] From the all 215,883 lists, we extracted lists showing senses obtained by heuristics (see lines 2,3,4,6,7,9 and 10 for Figure 2).
[2] Analyzed by Mecab, http://mecab.sourceforge.net/

2.2 Web Dictionary :Japanese Wikipedia

We used **Wikipedia**'s disambiguation pages,[5] as a target dictionary (see Figure 2). A disambiguation page lists articles (eg. `‘‘European Union’’`, `‘‘Ehime University’’`) associated with the same lemma (eg. “EU”). Our goal is to provide images for each article listed. As shown in Figure 2, they include various writing styles.

2.3 Comparison of Lexeed and Wikipedia

Table 1 shows the sizes of **Lexeed** and **Wikipedia**'s disambiguation pages, and the shared entries. Shared entries are rare, and account for less than

[4] http://ja.wikipedia.org/

[5] Version 20091011.

Original (in Japanese)

```
1  '''EU'''
2  * [[欧州連合]]
3  * [[Europa Universalis]]シリーズ - [[パラ
    ドックスインタラクティブ]]の[[歴史シミュレー
    ションゲーム]]
4  * [[愛媛大学]](Ehime University) - [[愛媛
    県]][[松山市]]にある日本の[[国立大学]]
5  '''Eu'''
6  * [[ユウロピウム]]の元素記号
7  * [[ユーフォニアム]] - 金管楽器
8 '''eu'''
9 * [[.eu]] - 欧州連合の[[国別ドメイン]]
10 * [[バスク語]]の[[ISO 639|ISO 639-1
    言語コード]]
```

Gloss

```
1 '''EU'''
2 * [[European Union]]
3 * [[Europa Universalis]] series - a [[histori-
  cal computer game]] by [[Paradox Interactive]]
4 * [[Ehime University]] - a [[National Univer-
  sity]] in [[Matsuyama]],[[Ehime Prefecture]]
5 '''Eu'''
6 * [[Europium]]'s chemical element symbol
7 * [[euphonium]] - a brass instrument
8 '''eu'''
9 * [[.eu]] - [[country-code top-level domain]]
   for the European Union
10 * [[ISO 639|ISO 639-1 language code]] of
   [[Basque]]
```

[[ ]] shows a link in **Wikipedia**. And we assign each line a number for easy citation.

Figure 2: Simplified Example of **Wikipedia**'s Disambiguation Page: "EU (disambiguation)"

10 % of the total [6][7]. As regards **Lexeed**, 16,685 entries (57 %) do not appear in any of **Wikipedia**'s lemmas, not only in disambiguation pages.[8]

As shown in Table 1, **Wikipedia** has many senses, but most of them are proper nouns. For example, in **Lexeed**, ヒマワリ *sunflower* is monosemous, but in **Wikipedia**, 67 senses are listed, including 65 proper nouns besides ``plant'' and ``sunflower oil''. On the other hand, in **Wikipedia**, アーチ *arch* has only one sense, ``architecture'' corresponding to **Lexeed**'s アーチ$_1$ *arch*, and has no disambiguation page.

As mentioned above, **Lexeed** and **Wikipedia** have very different types of entries and senses. This research aims to investigate the possibility of supplying appropriate images for such different senses, and a method for obtaining better images.

3 Experiment to Supply Images for Word Senses

In this paper, we propose a simple method for supplying appropriate images for each dictionary sense of a word. We collect candidate images from the Internet by using a querying image search. To obtain images even for minor senses, we expand the query by appending queries extracted from definitions for each sense.

In this paper, we investigated two main types of expansion, that is, the appending of mainly synonyms (SYN), and related words including hypernyms (LNK). For information retrieval, query expansion using synonyms has been adopted in several studies (Voorhees, 1994; Fang and Zhai, 2006; Unno et al., 2008). Our LNK is similar to methods used in Deng et al. (2009), but we note that their goal is not to give images to polysemous words (which is our intention). Popescu et al. (2006) also used synonyms (all terms in a synset) and hypernyms (immediate supertype in **WordNet**), but they did not investigate the effectiveness of each expansion and they forcus only on selected object synsets.

3.1 Experimental and Evaluation Method

We collected five candidate images for each sense from the Internet by querying an image search engine.[9] Then we manually evaluated the suitability of the image for explaining the target sense. The evaluator determined whether or not the image was appropriate (**T**), acceptable (**M**), or inappropriate (**F**). The evaluator also noted the reasons for **F**.

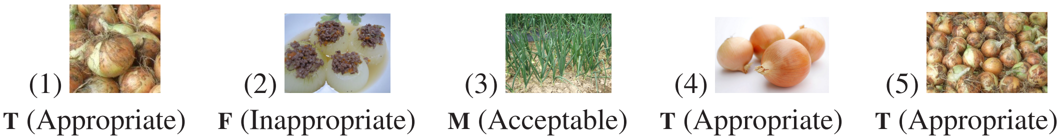
Figure 3 shows an example for たまねぎ *onion*. As shown in Figure 3, the evaluator determined **T**, **M** or **F** for each candidate image.

[6] Shared lemmas are そば *buckwheat noodle*, サイクル *cycle*, フクロウ *owl*, etc.

[7] Lemmas only in **Wikipedia** are イソップ *Aesop*, ビオ *Biot/Veoh*, 竜門の滝 *fall name*, etc.

[8] Lemmas only in **Lexeed** are 後払い *pay later*, ユーモラス *humorous*, 抜擢 *selection*, etc.

[9] We used Google AJAX images API, http://code.google.com/intl/ja/apis/ajaxsearch/



(1) **T** (Appropriate) (2) **F** (Inappropriate) (3) **M** (Acceptable) (4) **T** (Appropriate) (5) **T** (Appropriate)

Figure 3: Examples of Candidate Images and Evaluations for たまねぎ *onion*

Table 2: Data for **Hinoki** Ontology

Type	No.	%	Example	
			Lemma	Related Word
Hypernym	47,054	69.1	アーチ$_1$ *arch*	様式
Synonym	14,068	20.6	アーチ$_3$ *arch*	ホームラン *homer*
Domain	1,868	2.7	アーチ$_3$ *arch*	野球 *baseball*
Hyponym	757	1.1	売り買い$_1$ *buy and sell*	売る *sell*
Meronym	686	1.0	赤身$_1$ *lean*	魚肉 *fish meat*
Abbreviation	383	0.6	亜$_2$ *A(sia)*	アジア *Asia*
Other name	216	0.3	差し込み$_2$ *shave*	コンセント *plug outlet*
Other	3102	4.6	包み焼き$_1$ *papillote*	魚 *fish*
Total	68,134	100		

For an image that is related but that does not explain the sense, the evaluation is **F**. For example, for たまねぎ *onion*, the images of onion dishes such as (2) in Figure 3 are **F**. On the other hand, the images that show onions themselves such as (1), (4) and (5) in Figure 3 are **T**. With (3) in Figure 3, the image may show the onion itself or a field of onions, therefore the evaluation is **M**.

One point of judgment, specifically between **T** and **M**, is whether the image is typical or not. With たまねぎ *onion*, most typical images are similar to (1), (4) and (5). The image (3) may not be typical but is helpful for understanding, and (2) may lead to a misunderstanding if this is the only image shown to the dictionary user. This is why (3) is judged to be **M** and (2) is judged to be **F**.

We evaluated 200 target senses for **Lexeed**, and 100 for **Wikipedia**.[10]

3.2 Experiment: Lexeed

In this paper, we expand queries using the **Hinoki** Ontology (Bond et al., 2004), which includes related words extracted from the definition sentences. Table 2 shows the data for the **Hinoki** Ontology.

For SYN, we expand queries using synonyms, abbreviations, other names in Table 2, and variant spellings found in the dictionary. On the other hand, for LNK, we use all the remaining relations, namely hypernyms, domains, etc. Additionally, we use only normal spellings with no expansion, when the target words are monosemous (MONO). One exception should be noted. When the normal spelling employs hiragana (Japanese syllabary characters), we expand it using a variant spelling. For example, とんぼ *dragonfly* is expanded by the variant spelling 蜻蛉 *dragonfly*.

To investigate the trends and difficulties based on various conditions, we split the **Lexeed** senses into four types, namely, concrete and monosemous (MC), or polysemous (PC), not concrete and monosemous (MA), or polysemous (PA). We selected 50 target senses for evaluation randomly for each type. The target senses were randomly selected without distinguishing them in terms of their POS.

Note that we regard the sense as being something concrete that is linked to **GT**'s semantic classes subsumed by ⟨2:concrete⟩, such as たまねぎ *onion* (⊂ ⟨677:crop/harvest/farm products⟩ ⊂ ⟨2:concrete⟩).

3.3 Results and Discussion: Lexeed

Table 3 shows the ratio of **T** (appropriate), **M** (acceptable) and **F** (inappropriate) images for the target sense. We calculated the ratio using all five candidate images, for example, in Figure 3, the

[10] We performed an image search in September 2009 for **Lexeed**, and in December 2009 for **Wikipedia**.

ratio of appropriate images is 60 % (three of five).

In Table 3, the baseline shows a case where the query only involves the lemma (normal spelling). As shown in Table 3, SYN has higher precision than LNK. This means that SYN can focus on the appropriate sense. With polysemous words (PC, PA), expansion works more effectively, and helps to supply appropriate images for each sense. However, with MC, both LNK and SYN have less precision. This is because the target senses of MC are majorities, so expansion is adversely affected. Although MONO alone has good precision, because hiragana is often used as readings and has high ambiguity, appending the variant spelling helps us to focus on the appropriate sense.

Here, we focus on LNK of PC, and then analyze the reasons for **F** (Table 5). In Table 5, in 24.3% of cases it is “difficult to portray the sense using images” (The numbers of senses for which it is “difficult to portray the sense using images” are, 3 of MC, 9 of PC, 10 of MA, and 16 of PA. We investigate such senses in more detail in § 3.4.).

For such senses, no method can provide suitable images, as might be expected. Therefore, we exclude targets where it is “difficult to portray the sense using images”, then we recalculated the ratio of appropriate images. Table 4 shows the capability of our proposed method for senses that can be explored using images. This leads to 66.3 % precision (15.3% improvement) even for most difficult target type, PA.

Again, when we look at Table 5, reasons 2-5 (33.3 %) will be improved. In particular, “hypernym leads to ambiguity” makes up more than 10%. Hypernyms sometimes work well, but sometimes they lead to other words included in the hypernyms. For example, appending the hypernym 食品 *foods* to 煮干し *boiled-dried fish* leads to images of “foods made with boiled-dried fish”. This is why SYN obtained better results than LNK. Then, with “expanded by minor sense” and when the original sense is dominant majority, expansion reduced the precision. Therefore, we should expand using only words with major senses.

3.4 Discussion: Senses can/cannot be shown by images

As described above, the target senses are randomly selected without being distinguished by their POS, because we also want to investigate the features of senses that can be shown by images. Table 6 shows the ratio of senses judged as “difficult to portray the sense using images” (labeled as “Not Shown”) for each POS. As regards POS, the majority of selected senses are nouns, followed by verbal nouns and verbs. We expected that the majority of nouns and verbal nouns whould be “Shown”, but did not expect that a majority of verb is also “Shown”. Other POSs are too rare to judge, although they tend to fall in the “Not Shown” category.

Furthermore, in Table 7, for nouns and verbal nouns, we show the ratio of senses for each type (“Concrete” or “not Concrete”) judged in terms of “difficult to portray the sense using images”. We classified the senses into “Concrete” or “not Concrete” based on **GT**’s semantic classes, as described in § 3.2.

Table 6: Ratio of Senses judged as “difficult to portray the sense using images” for each POS

POS	Shown		Not Shown		Total
	No.	%	No.	%	No.
Noun	132	85.2	23	14.8	155
Verbal Noun	15	78.9	4	21.1	19
Verb	9	81.8	2	18.2	11
Affix	4	57.1	3	42.9	7
Pronoun	0	0	2	100	2
Adjective	1	50	1	50	2
Adverb	0	0	2	100	2
Interjection	1	100	0	0	1
Conjunction	0	0	1	100	1
Total	162	81	38	19	200

Table 7: Ratio of Concrete/Not Concrete Senses judged as “difficult to portray the sense using images”: for Nouns and Verbal Nouns

Type	Shown		Not Shown		Total
	No.	%	No.	%	No.
Concrete	114	90.5	12	9.5	126
Not Concrete	33	68.8	15	31.3	48
Total	147	84.5	27	15.5	174

Table 3: Ratio of Appropriate Images for Sense (Precision): **Lexeed**

Target Type		Expanding Method	**F** (Inappropriate)		**T** (Appropriate)		**M** (Acceptable)		**T+M**		
			No.	%	No.	%	No.	%	No.	%	Total
Concrete	Monosemous (MC)	SYN	18	24.0	36	48.0	21	28.0	57	76.0	75
		LNK	82	**33.5**	112	45.7	51	20.8	163	66.5	245
		MONO	42	16.8	181	**72.4**	27	10.8	208	**83.2**	250
		baseline	46	18.4	171	68.4	33	13.2	204	81.6	250
	Polysemous (PC)	SYN	94	38.7	88	36.2	61	25.1	149	**61.3**	243
		LNK	111	**44.4**	92	36.8	47	18.8	139	55.6	250
		baseline	180	**72.0**	53	21.2	17	6.8	70	28.0	250
not Concrete	Monosemous (MA)	SYN	32	42.7	21	28.0	22	29.3	43	**57.3**	75
		LNK	138	**57.5**	54	22.5	48	20.0	102	42.5	240
		MONO	98	40.0	98	40.0	49	20.0	147	**60.0**	245
		baseline	112	44.8	86	34.4	52	20.8	138	55.2	250
	Polysemous (PA)	SYN	122	49.0	64	25.7	63	25.3	127	**51.0**	249
		LNK	150	**60.2**	52	20.9	47	18.9	99	39.8	249
		baseline	201	**80.7**	36	14.5	12	4.8	48	19.3	249

Table 4: Ratio of Appropriate Images for Sense (Precision), excluding senses that are difficult to portray using images: **Lexeed**

Target Type		Expanding Method	**F** (Inappropriate)		**T** (Appropriate)		**M** (Acceptable)		**T+M**		
			No.	%	No.	%	No.	%	No.	%	Total
Concrete	Monosemous (MC)	SYN	15	**21.4**	36	51.4	19	27.1	55	**78.6**	70
		LNK	71	**30.9**	112	48.7	47	20.4	159	69.1	230
		MONO	29	12.3	180	76.6	26	11.1	206	**87.7**	235
		baseline	35	14.9	170	72.3	30	12.8	200	85.1	235
	Polysemous (PC)	SYN	61	30.8	85	42.9	52	26.3	137	**69.2**	198
		LNK	84	40.0	89	42.4	37	17.6	126	60.0	210
		baseline	139	67.8	53	25.9	13	6.3	66	32.2	205
not Concrete	Monosemous (MA)	SYN	17	34.0	20	40.0	13	26.0	33	**66.0**	50
		LNK	101	**51.8**	54	27.7	40	20.5	94	48.2	195
		MONO	65	33.3	94	48.2	36	18.5	130	**66.7**	195
		baseline	72	36	85	42.5	43	21.5	128	64.0	809
	Polysemous (PA)	SYN	57	33.7	63	37.3	49	29	112	**66.3**	169
		LNK	81	47.9	52	30.8	36	21.3	88	52.1	169
		baseline	122	72.2	36	21.3	11	6.5	47	27.8	169

Table 5: Reasons for **F**: PC, LNK:**Lexeed**

No.	Reason	No.	%	Example
1	difficult to portray the sense using images	27	24.3	これ *me* ``humble expressions used for oneself''
2	hypernym leads to ambiguity	12	10.8	煮干し *boiled-dried fish* (⊂ 食品 *foods*)
3	expanded by minor sense	11	9.9	リンク *link* (⊂ リンクス *links*, usually means *lynx*)
4	no expansion is better	8	7.2	カメラマン *cameraman* (⊂ 部員 *staff*)
5	original sense is TOO minor	6	5.4	海 *lake* (⊂ 湖 *lake*), 海 usually means *sea*
6	Other	47	42.3	
	Total	111	100	

As shown in Table 7, 90.5 % of "Concrete" nouns are judged as "Shown", and only 9.5 % of senses are judged as "Not Shown" [11]. However 68.8 % of "not Concrete" nouns are also judged as "Shown".

Therefore, both POS and type ("Concrete" or "not Concrete") are helpful, but not perfect features as regards knowing the sense is "difficult to portray the sense using images". In future work we will undertake further analysis to determine the critical features.

3.5 Experiment: **Wikipedia**

For LNK we use the **Wikipedia** hyperlinks (shown as [[ ]] in Fig 2). 95.5 % of all senses include [[ ]], 85.4 % linked to an actual page, and [[ ]] appeared 0.95 times per sense. Note that we do not use time expression links such as [[2010]] and [[1990s]].

With SYN, we use synonyms extracted with heuristics. Table 8 shows the main rules that we used to extract synonyms. We extracted synonyms for 98.0 % of 197,912 senses.

Then we randomly selected 50 target senses for evaluation from lemmas shared/unshared by **Lexeed**.

3.6 Results and Discussion: **Wikipedia**

We do not show the baseline in Table 9, but it is always below 10%. For all target senses, expansion provides more suitable images. Because there are so many senses in **Wikipedia**, no target sense is in the majority. As shown in Table 9, there are few differences between SYN and LNK, because most of the synonyms used for SYN are also links. However, SYN has slightly superior precision as regards **T** (Appropriate), which means the process of extracting synonyms helped to reject links that were poorly with the target senses.

Also in **Lexeed**, expansion using synonyms (SYN) had higher precision than hypernyms (LNK). Because we do not know the total number of suitable images for the target senses on the Internet, we cannot estimate the recall with this evaluation method. However, we speculate that hypernyms provide higher recall. Deng et al. (2009) undertook expansion using hypernyms and this may be an appropriate way to obtain many more images for each sense. However, because our aim is employ several suitable images for each sense, high precision is preferable to high recall.

Now, we focus on LNK shared by **Lexeed**, and then we analyze the reasons for **F** (Table 10). In contrast to **Lexeed**, no sense is classified as "difficult to portray the sense using images". However, there are many senses where it is difficult to decide what kind of images "explain the target sense". For example, in Table 10, with "maybe **T** (Appropriate)", the target sense was a personal name and the image was his/her representative work. In this paper, for personal names, only the images of the person are judged to be **T**, despite the fact that supplying images of representative work for novelists or artists may be suitable.

In this study, we obtained five images per sense, but only one image was sufficient for some senses, for example, an image of an album cover for the name of an album. In contrast, several different types of images are needed for some senses. For example, for the name of a city, images of maps, landscapes, city offices, symbols of the city, etc. are all suitable. Therefore, it may be better to estimate a rough class first, such as the name of an album, artist and place, and then obtain preassigned types of images.

4 Conclusions

The goal of this work was to supply several suitable images for dictionary definitions. The target dictionaries were **Lexeed** and **Wikipedia**, which have very different characteristics. To cover a wide range of senses, we collected candidate images from the Internet by querying an image search engine. Then, to obtain suitable and different images for each sense, we expanded the queries by appending related words extracted from the definition sentences. In this paper, we tried two types of expansion, one mainly using synonyms (SYN), and one mainly using hypernyms or related links (LNK).

The results show that SYN provided better precision than LNK, especially for **Lexeed**. Also, query expansion provided a substantial improvement for

[11] For example, 学会 *conference* (⊂ ⟨373:organization, etc.⟩ ⊂ ⟨2:concrete⟩), 親代わり *parental surrogate* (⊂ ⟨342:agent/representative⟩ ⊂ ⟨2:concrete⟩), and so on.

Table 8: Rules for Extracting Synonyms for SYN: **Wikipedia**

Rule	Example	
	Lemma	Definition sentences
head parts separated by hyphen (- or –)	EU	[[euphonium]] - a brass instrument (line 7 in Figure 2)
whole definitions appear as a chunk	EU	[[European Union]] (line 2 in Figure 2)
parts indicated by		
arrow (→)	イヌ *dog*	One of [[Oriental Zodiac]]→[[戌 *dog*]]
quotation key words, 参照 *See* etc.	イヌ *dog*	[[Chinese character]]'s [[radical parts]], See [[犬部 *inu-bu*]]
parts in parentheses or " " including		
whole lemma	Einstein	"Albert Einstein"
alphameric characters, for katakana lemma	サンバ	"samba"
characters of alpha-numeral lemma	CS	コンピュータ科学 (computer science)

underlined parts show the extracted synonyms.

Table 9: Ratio of Appropriate Images for Sense (Precision): **Wikipedia**

Target Type	Expanding Method	**F** (Inappropriate) No.	%	**T** (Appropriate) No.	%	**M** (Acceptable) No.	%	**T+M** No.	%	Total
Shared by	SYN	98	40.8	119	**49.6**	23	9.6	142	**59.2**	240
Lexeed	LNK	92	41.8	107	48.6	21	9.5	128	58.2	220
NOT shared	SYN	100	41.2	103	**42.4**	40	16.5	143	58.8	243
by **Lexeed**	LNK	96	41.0	93	39.7	45	19.2	138	**59.0**	234

Table 10: Reasons for **F**: Shared by **Lexeed**, LNK: **Wikipedia**

No.	Reason	No.	%	Example	
				Lemma	Links
7	lack of queries (available words in def.)	14	15.2	ふえ *fue (reading)*	フエ *Hue, city name in Vietnam*
8	inappropriate queries (available words in def.)	10	10.9	レギュラー *regular*	出場選手登録 *active roster*
2	hypernym lead to ambiguity	5	5.4	キャッシュ *cache*	ジオキャッシング *geocaching*
9	maybe **T** (Appropriate)	5	5.4	モンキー *monkey*	モンキー・パンチ *Monkey Punch*
6	Other	58	63		
	Total	92	100		

polysemous words. Our proposed method is simple but effective for our purpose, that is supplying suitable and different images for each sense.

In future work we intend to analyze senses that are difficult/easy to portray using images in more detail, using not only semantic charactaristics but also visual features(Csurka et al., 2004). We also intend to improve the expansion method. One way to achieve this is to filter out expansions with minor senses. As for **Wikipedia**, we should approximate the class first, such as the name of an album, artist and place, then obtain preassigned types of images.

References

Kobus Barnard, Pinar Duygulu, Nando de Freitas, David Forsyth, David Blei, and Michael I. Jordan. 2003. Matching Words and Pictures. *Journal of Machine Learning Research*, Vol. 3, pp. 1107–1135.

Francis Bond, Hitoshi Isahara, Sanae Fujita, Kiyotaka Uchimoto, Takayuki Kuribayashi, and Kyoko Kanzaki. 2009. Enhancing the Japanese WordNet. In *The 7th Workshop on Asian Language Resources, in conjunction with ACL-IJCNLP-2009*, pp. 1–8.

Francis Bond, Eric Nichols, Sanae Fujita, and Takaaki Tanaka. 2004. Acquiring an Ontology for a Fundamental Vocabulary. In *Proceedings of the 20th International Conference on Computational Linguistics: COLING-2004*, pp. 1319–1325.

Andy Borman, Rada Mihalcea, and Paul Tarau. 2005. PicNet: Pictorial Representations for Illustrated Semantic Networks. In *Proceedings of the AAAI Spring Symposium on Knowledge Collection from Volunteer Contributors*.

Gabriela Csurka, Cedric Bray, Chris Dance, and Lixin Fan. 2004. Visual categorization with bags of keypoints. In *ECCV International Workshop on Statistical Learning in Computer Vision*, pp. 59–74.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.

Hui Fang and ChengXiang Zhai. 2006. Semantic term matching in axiomatic approaches to information retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pp. 115–122. ACM.

Christine Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Atsushi Fujii and Tetsuya Ishikawa. 2005. Image Retrieval and Disambiguation for Encyclopedic Web Search. In *Proceedings of the International Joint Conference on Artificial Intelligence: IJCAI-2005*, pp. 1598–1599.

Andrew B. Goldberg, Jake Rosin, Xiaojin Zhu, and Charles R. Dyer. 2009. Toward Text-to-Picture Synthesis. In *NIPS 2009 Mini-Symposia on Assistive Machine Learning for People with Disabilities*.

Yoshihiko Hayashi, Savas Bora, and Masaaki Nagata. 2009. Utilizing Images for Assisting Cross-language Information Retrieval on the Web. In *International Workshop on Web Information Retrieval Support Systems*, pp. 100–103.

Ichiro Ide and Keiji Yanai. 2009. Crossing the Semantic Gap : Towards the Understanding of Image and Video Contents. *Journal of Japanese Society for Artificial Intelligence*, Vol. 24, No. 5, pp. 691–699. (in Japanese).

Satoru Ikehara, Masahiro Miyazaki, Satoshi Shirai, Akio Yokoo, Hiromi Nakaiwa, Kentaro Ogura, Yoshifumi Ooyama, and Yoshihiko Hayashi. 1997. *Goi-Taikei — A Japanese Lexicon*. Iwanami Shoten, Tokyo. 5 volumes/CD-ROM.

Kaname Kasahara, Hiroshi Sato, Francis Bond, Takaaki Tanaka, Sanae Fujita, Tomoko Kanasugi, and Shigeaki Amano. 2004. Construction of a Japanese Semantic Lexicon: Lexeed. In *IEICE Technical Report: 2004-NLC-159*, pp. 75–82. (in Japanese).

Rada Mihalcea and Chee Wee Leong. 2008. Toward communicating simple sentences using pictorial representations. *Machine Translation*, Vol. 22, No. 3, pp. 153–173.

Adrian Popescu, Christophe Millet, and Pierre-Alain Moëllic. 2007. Ontology Driven Content Based Image Retrieval. In *Proceedings of the ACM International Conference on Image and Video Retrieval*.

Adrian Popescu, Christophe Millet, Pierre-Alain Moëllic, Patrick Hède, and Gregory Grefenstette. 2006. Automatic Construction of a Grounded Multimedia Ontology of Objects to Illustrate Concepts in a Learning Process. In *NETTIES 2006 Conference: Advanced Educational Technologies for a Future e-Europe*.

Arnold W.M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. 2000. Content-based Image Retrieval at the End of the Early Years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 12, pp. 1349–1380.

Yuya Unno, Yusuke Miyao, and Jyunichi Tujii. 2008. Information Retrieval using Automatically Extracted paraphrases. In *Proceedings of the 14th Annual Meeting of The Association for Natural Language Processing: NLP-2008*, pp. 123–126. (in Japanese).

Ellen M. Voorhees. 1994. Query Expansion using Lexical-Semantic Relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pp. 61–69.

Feng Yansong and Mirella Lapata. 2008. Automatic image annotation using auxiliary text information. In *Proceedings of ACL-08: HLT*, pp. 272–280. Association for Computational Linguistics.

Svitlana Zinger, Christophe Millet, Benoit Mathieu, Gregory Grefenstette, Patrick Hède, and Pierre-Alain Moëllic. 2006. Clustering and semantically filtering web images to create a large-scale image ontology. In *SPIE 18th Annual Symposium Electronic Imaging, Internet Imaging VII*, Vol. 6061, pp. 89–97.

Opinosis: A Graph-Based Approach to Abstractive Summarization of Highly Redundant Opinions

Kavita Ganesan and **ChengXiang Zhai** and **Jiawei Han**
Department of Computer Science
University of Illinois at Urbana-Champaign
{kganes2,czhai,hanj}@cs.uiuc.edu

Abstract

We present a novel graph-based summarization framework (Opinosis) that generates concise abstractive summaries of highly redundant opinions. Evaluation results on summarizing user reviews show that Opinosis summaries have better agreement with human summaries compared to the baseline extractive method. The summaries are readable, reasonably well-formed and are informative enough to convey the major opinions.

1 Introduction

Summarization is critically needed to help users better digest the large amounts of opinions expressed on the web. Most existing work in Opinion Summarization focus on predicting sentiment orientation on an entity (Pang et al., 2002) (Pang and Lee, 2004) or attempt to generate aspect-based ratings for that entity (Snyder and Barzilay, 2007) (Lu et al., 2009)(Lerman et al., 2009)(Titov and Mcdonald, 2008). Such summaries are very informative, but it is still hard for a user to understand why an aspect received a particular rating, forcing a user to read many, often highly redundant sentences about each aspect. To help users further digest the opinions in each aspect, it is thus desirable to generate a concise textual summary of such redundant opinions.

Indeed, in many scenarios, we will face the problem of summarizing a large number of highly redundant opinions; other examples include summarizing the 'tweets' on Twitter or comments made about a blog or news article. Due to the subtle variations of redundant opinions, typical extractive methods are often inadequate for summarizing such opinions. Consider the following sentences:

1. *The iPhone's battery lasts long, only had to charge it once every few days.*
2. *iPhone's battery is bulky but it is cheap..*
3. *iPhone's battery is bulky but it lasts long!*

With extractive summarization, no matter which single sentence of the three is chosen as a summary, the generated summary would be biased. In such a case, an abstractive summary such as '*iPhone's battery is cheap, lasts long but is bulky*' is a more complete summary, conveying all the necessary information. Extractive methods also tend to be verbose and this is especially problematic when the summaries need to be viewed on smaller screens like on a PDA. Thus, an informative and concise abstractive summary would be a better solution.

Unfortunately, abstractive summarization is known to be difficult. Existing work in abstractive summarization has been quite limited and can be categorized into two categories: (1) approaches using prior knowledge (Radev and McKeown, 1998) (Finley and Harabagiu, 2002) (DeJong, 1982) and (2) approaches using Natural Language Generation (NLG) systems (Saggion and Lapalme, 2002) (Jing and McKeown, 2000). The first line of work requires considerable amount of manual effort to define schemas such as frames and templates that can be filled with the use of information extraction techniques. These systems were mainly used to summarize news articles. The second category of work uses deeper NLP analysis with special techniques for text regeneration. Both approaches either heavily rely on manual effort or are domain dependent.

In this paper, we propose a novel flexible summarization framework, Opinosis, that uses graphs to produce abstractive summaries of highly redundant opinions. In contrast with the previous work, Opinosis assumes no domain knowledge and uses shallow NLP, leveraging mostly the word order in the existing text and its inherent redundancies to generate informative abstractive summaries. The key idea of Opinosis is to first construct a textual graph that represents the text to be summarized. Then, three unique properties of this graph are used to explore and score various subpaths that help in generating candidate abstractive summaries.

Evaluation results on a set of user reviews show that Opinosis summaries have reasonable agreement with human summaries. Also, the gener-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 340–348,
Beijing, August 2010

ated summaries are readable, concise and fairly well-formed. Since Opinosis assumes no domain knowledge and is highly flexible, it can be potentially used to summarize any highly redundant content and could even be ported to other languages. (All materials related to this work including the dataset and demo software can be found at `http://timan.cs.uiuc.edu/downloads.html`.)

2 Opinosis-Graph

Our key idea is to use a graph data structure (called Opinosis-Graph) to represent natural language text and cast this abstractive summarization problem as one of finding appropriate paths in the graph. Graphs have been commonly used for extractive summarization (e.g., LexRank (Erkan and Radev, 2004) and TextRank (Mihalcea and Tarau, 2004)), but in these works the graph is often *undirected* with *sentences as nodes* and similarity as edges. Our graph data structure is different in that each node represents a *word unit* with *directed edges* representing the structure of sentences. Moreover, we also attach positional information to nodes as will be discussed later.

Algorithm 1 (A1): $OpinosisGraph(Z)$

```
1:  Input: Topic related sentences to be summarized: $Z = \{z_i\}_{i=1}^n$
2:  Output: $G = (V, E)$
3:  for $i = 1$ to $n$ do
4:    $w \leftarrow Tokenize(z_i)$
5:    $sent\_size \leftarrow SizeOf(w)$
6:    for $j = 1$ to $sent\_size$ do
7:      $LABEL \leftarrow w_j$
8:      $PID \leftarrow j$
9:      $SID \leftarrow i$
10:     if $ExistsNode(G, LABEL)$ then
11:       $v_j \leftarrow GetExistingNode(G, LABEL)$
12:       $PRI_{v_j} \leftarrow PRI_{v_j} \cup (SID, PID)$
13:     else
14:       $v_j \leftarrow CreateNewNode(G, LABEL)$
15:       $PRI_{v_j} \leftarrow (SID, PID)$
16:     end if
17:     if not $ExistsEdge(v_{j-1} \rightarrow v_j, G)$ then
18:       $AddEdge(v_{j-1} \rightarrow v_j, G)$
19:     end if
20:   end for
21: end for
```

Our graph representation is closer to that used by Barzilay and Lee (Barzilay and Lee, 2003) for the task of paraphrasing, wherein each node in the graph represents a unique word. However, in their work, such a graph is used to identify regions of commonality and variability amongst similar sentences. Thus, the positional information is not required nor is it maintained. In contrast, we maintain positional information at each node as this is critical for the selection of candidate paths.

Algorithm **A1** outlines the steps involved in building an Opinosis-Graph. We start with a set of sentences relevant to a specific topic, which can be obtained in different ways depending on the application. For example, they may be all sentences related to the *battery life* of the *iPod Nano*. We denote these sentences as $Z = \{z_i\}_{i=1}^n$ where each z_i is a sentence containing part-of-speech (POS) annotations. (A1:4) Each $z_i \in Z$ is split into a set of *word units*, where each unit, w_j consists of a word and its corresponding POS annotation (e.g. "*service:nn*", "*good:adj*"). (A1:7-9) Each unique w_j will form a node, v_j, in the Opinosis-Graph, with w_j being the label. Also, since we only have one node per unique word unit, each node keeps track of all sentences that it is a part of using a *sentence identifier* (SID) along with its *position of occurrence* in that sentence (PID). (A1:10-16) Each node will thus carry a *Positional Reference Information* (PRI) which is a list of $\{$SID:PID$\}$ pairs representing the node's membership in a sentence. (A1:17-19) The original structure of a sentence is recorded with the use of directed edges. Figure 1 shows a resulting Opinosis-Graph based on four sentences.

The Opinosis-Graph has some unique properties that are crucial in generating abstractive summaries. We highlight some of the core properties by drawing examples from Figure 1:

Property 1. *(**Redundancy Capture**). Highly redundant discussions are naturally captured by subgraphs.*

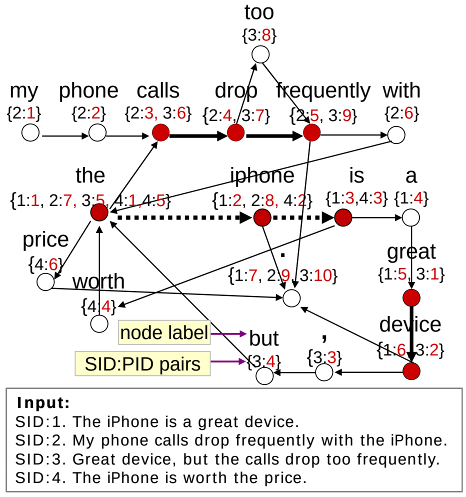
Figure 1 shows that although the phrase '*great device*' was mentioned in different parts of sentences (1) and (3), this phrase forms a relatively heavy sub-path in the resulting graph. This is a good indication of salience.

Property 2. *(**Gapped Subsequence Capture**). Existing sentence structures introduce **lexical links** that facilitate the discovery of new sentences or reinforce existing ones.*

The main point conveyed by sentences (2) and (3) in Figure 1 is that *calls drop frequently*. However, this is expressed in slightly different ways and is reflected in the resulting subgraph. Since sentence (2) introduces a *lexical link* between '*drop*' and '*frequently*', the word '*too*' can be ignored for sentence (3) as the same amount of information is retained. This is analogous to capturing a *repetitive gapped subsequence* where similar sequences with minor variations are captured. With this, the subgraph *calls drop frequently* can be considered redundant.

Property 3. *(**Collapsible Structures**). Nodes that resemble hubs are possibly collapsible.*

In Figure 1 we see that the subgraph '*the iPhone is*', is fairly heavy and the 'is' node acts like a



Input:
SID:1. The iPhone is a great device.
SID:2. My phone calls drop frequently with the iPhone.
SID:3. Great device, but the calls drop too frequently.
SID:4. The iPhone is worth the price.

Figure 1: Sample *Opinosis-Graph*. Thick edges indicate salient paths.

'*hub*' where it connects to various other nodes. Such a structure is naturally captured by the Opinosis-Graph and is a good candidate for compression to generate a summary such as '*The iPhone is a great device and is worth the price*'. Also, certain word POS (e.g. linking verbs like 'is' and 'are') often carry hub-like properties that can be used in place of the outlink information.

3 Opinosis Summarization Framework

In this section, we describe a general framework for generating abstractive summaries using the Opinosis-Graph. We also describe our implementation of the components in this framework.

At a high level, we generate an abstractive summary by repeatedly searching the Opinosis graph for appropriate subgraphs that both encode a valid sentence (thus meaningful sentences) and have high redundancy scores (thus representative of the major opinions). The sentences encoded by these subgraphs would then form an abstractive summary.

Going strictly by the definition of true abstraction (Radev et al., 2002), our problem formulation is still more extractive than abstractive because the generated summary can only contain words that occur in the text to be summarized; our problem definition may be regarded as a word-level (finer granularity) extractive summarization. However, compared to the conventional sentence-level extractive summarization, our formulation has flavors of abstractive summarization wherein we have elements of *fusion* (combining extracted portions) and *compression* (squeezing out unimportant material from a sentence). Hence, the sentences in the generated summary are generally not the same as any original sentence. Such a "shallow" abstractive summarization problem is more tractable, enabling us to develop a general solution to the problem. We now describe each component in such a summarization framework.

3.1 Valid Path

A valid path intuitively refers to a path that corresponds to a meaningful sentence.

Definition 1. *(**Valid Start Node - VSN**). A node v_q is a valid start node if it is a natural starting point of a sentence.*

We use the positional information of a node to determine if it is a VSN. Specifically, we check if $Average(PID_{v_q}) \leq \sigma_{vsn}$, where σ_{vsn} is a parameter to be empirically set. With this, we only qualify nodes that tend to occur early on in a sentence.

Definition 2. *(**Valid End Node - VEN**). A node v_s is a valid end point if it completes a sentence.*

We use the natural ending points in the text to be summarized as hints to which node may be a valid end point of a path (i.e., a sentence). Specifically, a node is a *valid end node* if (1) the node is a punctuation such as *period* and *comma* or (2) the node is any coordinating conjunction (e.g., *'but'* and *'yet'*).

Definition 3. *(**Valid Path**). A path $W = \{v_q...v_s\}$ is valid if it is connected by a set of directed edges such that (1) v_q is a **VSN**, (2) v_s is a **VEN**, and (3) W satisfies a set of well-formedness POS constraints.*

Since not every path starting with a VSN and ending at a VEN encodes a meaningful sentence, we further require a valid path to satisfy the following POS constraints (expressed in regular-expression) to ensure that a valid path encodes a well-formed sentence:

1. $.*(/nn)+.*(/vb)+.*(/jj)+.*$
2. $.*(/jj)+.*(/to)+.*(/vb).*$
3. $.*(/rb)*.*(/jj)+.*(/nn)+.*$
4. $.*(/rb)+.*(/in)+.*(/nn)+.*$

This also provides a way (if needed) for the application to generate only specific type of sentences like *comparative sentences* or *strictly opinionated sentences*. These rules are thus application specific.

3.2 Path Scoring

Intuitively, to generate an abstractive summary, we should select a valid path that can represent most of the redundant opinions well. We would thus favor a valid path with a high redundancy score.

Definition 4. *(**Path Redundancy**). Let $W = \{v_q...v_s\}$ be a path from an Opinosis-Graph. The path redundancy of W, $r(q, s)$, is the number of overlapping sentences covered by this path, i.e.,*

$$r(q,s) = n_q \bar{\cap} n_{q+1} ... \bar{\cap} n_s,$$

where $n_i = PRI_{v_i}$ and $\bar{\cap}$ is the intersection between two sets of SIDs such that the difference between the corresponding PIDs is no greater than σ_{gap}, and $\sigma_{gap} > 0$ is a parameter.

Path redundancies provide good indication of how many sentences discuss something similar at each point in the path. The σ_{gap} parameter controls the maximum allowed gaps in discovering these redundancies. Thus, a common sentence X between nodes v_q and v_r, will be considered a valid intersect if $(PID_{v_{r_x}} - PID_{v_{q_x}}) \leq \sigma_{gap}$.

Based on path redundancy, we propose several ways to score a path for the purpose of selecting a good path to include in the summary:

1. $S_{basic}(W) = \frac{1}{|W|} \sum_{k=i+1,i}^{s} r(i,k)$

2. $S_{wt\_len}(W) = \frac{1}{|W|} \sum_{k=i+1,i}^{s} |v_i, v_k| * r(i,k)$

3. $S_{wt\_loglen}(W) = \frac{1}{|W|}(r(i,i+1) + \sum_{k=i+2,i+1}^{s} log_2 |v_i, v_k| * r(i,k))$

v_i is the first node in the path being scored and v_s is the last node. $|v_i, v_k|$ is the length from node v_i to v_k. $|W|$ is the length of the entire path being scored. The S_{basic} scoring function scores a path purely based on the level of redundancy. One could also argue that high redundancy on a longer path is intuitively more valuable than high redundancy on a shorter path as the former would provide better coverage than the latter. This intuition is factored in by the S_{wt_len} and S_{wt_loglen} scoring functions where the level of *redundancy* is *weighted* by the *path length*. S_{wt_loglen} is similar to S_{wt_len} only that it scales down the path length so that it does not entirely dominate.

3.3 Collapsed paths

In some cases, paths in the Opinosis-Graph may be collapsible (as explained in Section 2). In such a case, the collapse operation is performed and then the path scores are computed. We will now explain a few concepts related to collapsible structures. Let $\widehat{W} = \{v_i...v_k\}$ be a path from the Opinosis-Graph.

Definition 5. ***(Collapsible Node).*** *Node v_k is a candidate for collapse if its POS is a* verb.

We only attempt to collapse nodes that are *verbs* due to the heavy usage of verbs in opinion text and the ease with which the structures can be combined to form a new sentence. However, as mentioned earlier other properties like the outlink information can be used to determine if a node is collapsible.

Definition 6. ***(Collapsed Candidates, Anchor).*** *Let v_k be a collapsible node. The collapsed candidates of v_k (denoted by $CC = \{cc_i\}_{i=1}^{m}$) are the remaining paths after v_k in all the valid paths going through $v_i...v_k$. The prefix $v_i...v_k$ is called the anchor, denoted as $C_{anchor} = \{v_i...v_k\}$. Each path $\{v_i...v_n\}$, where v_n is the last node in each $cc_i \in CC$, is an individually valid path.*

C_{anchor}	CC	Connector
a. the sound quality is	cc_1 : really good cc_2 : clear	and
b. the iphone is	cc_1 : great cc_2 : expensive	but

Table 1: Example of anchors, collapsed candidates and suitable connectors

Table 1 shows a simplistic example of anchors and corresponding collapsed candidates. Once the anchor and collapsed candidates have been identified, the task is then to combine all of these to form a new sentence.

Definition 7. ***(Stitched Sentence)*** *A stitched sentence is one that combines C_{anchor} and CC to form a combined, logical sentence.*

We will now describe the stitching procedure that we use, by drawing examples from Table 1. Since we are dealing with verbs, C_{anchor} can be combined with the corresponding CC with commas to separate each $cc_i \in CC$ with one exception - the correct sentence connector has to be used for the last cc_i. For C_{anchor_a}, the phrases *really good* and *clear* can be connected by 'and' due to the same sentiment orientation. For C_{anchor_b}, the collapsed candidate phrases are well connected by the word 'but'. We use the existing Opinosis-Graph to determine the most appropriate connector. We do this by looking at all *coordinating conjunction* (e.g. 'but', 'yet') nodes (v_{cconj}) that are connected to the first node of the last collapsed candidate, cc_m. This would be the node labeled '*clear*' for C_{anchor_a} and '*expensive*' for C_{anchor_b}. We denote these nodes as v_{0,cc_m}. The v_{cconj}, with the highest *path redundancy* with v_{0,cc_m}, will be selected as the connector.

Definition 8. ***(Collapsed Path Score)*** *The final path score after the entire collapse operation is the average across path scores computed from v_i to the last node in each $cc_i \in CC$.*

The collapsed path score essentially involves computing the *path scores* of the individual sentences assuming that they are not collapsed and then averaging them.

3.4 Generation of summary

Once we can score all the valid paths as well as all the collapsed paths, the generation of an abstractive summary can be done in two steps: First, we rank all the paths (including the collapsed paths) in descending order of their scores. Second, we eliminate duplicated (or extremely similar) paths by using a similarity measure (in our experiments, we used Jaccard). We then take the top few remaining paths as the generated summary, with the number of paths to be chosen controlled by a parameter σ_{ss}, which represents summary size.

Although conceptually we enumerate all the valid paths, in reality we can use a redundancy score threshold, σ_r to prune many non-promising paths. This is reasonable because we are only interested in paths with high redundancy scores.

4 Summarization Algorithm

Algorithms **A2** and **A3** describe the steps involved in Opinosis Summarization. A2 is the starting point of the Opinosis Summarization and A3 is a subroutine where path finding takes place, invoked from within A2.

Algorithm 2 (A2): $OpinosisSummarization(Z)$

```
 1: Input: Topic related sentences to be summarized: $Z = \{z_i\}_{i=1}^{n}$
 2: Output: $\mathcal{O}$ ={Opinosis Summaries}
 3: $g \leftarrow OpinosisGraph(Z)$
 4: $node\_size \leftarrow SizeOf(g)$
 5: for $j = 1$ to $node\_size$ do
 6:   if $VSN(v_j)$ then
 7:     $pathLen \leftarrow 1$
 8:     $score \leftarrow 0$
 9:     $cList \leftarrow CreateNewList()$
10:     Traverse$(cList, v_j, score, PRI_{v_j}, label_{v_j}, pathLen)$
11:     $candidates \leftarrow \{candidates \cup cList\}$
12:   end if
13: end for
14: $\mathcal{C} \leftarrow EliminateDuplicates(candidates)$
15: $\mathcal{C} \leftarrow SortByPathScore(\mathcal{C})$
16: for $i = 1$ to $\sigma_{ss}$ do
17:   $\mathcal{O} = \{\mathcal{O} \cup PickNextBestCandidate(\mathcal{C})\}$
18: end for
```

(A2:3) Opinosis Summarization starts with the construction of the Opinosis-Graph, described in detail in Section 2. This is followed by the *depth first* traversal of this graph to locate *valid paths* that become *candidate summaries*. (A2:6-12) To achieve this, each node v_j in the Opinosis-Graph is examined to determine if it is a *VSN* and, if it is, path finding will start from this node by invoking subroutine A3. A3 takes the following as input: *list* - a list to hold candidate summaries; v_i - the node to continue traversal from; *score* - the accumulated path score; $PRI_{overlap}$ - the intersect between PRIs of all nodes visited so far (see Definition 4); *sentence* - the summary sentence formed so far; *len* - the current path length. (A2:7-10) Before invoking A3 from A2, the path length is set to '1', path score is set to '0' and a new list is created to store candidate summaries generated from node v_j. (A2:11) All candidate summaries generated from v_j will be stored in a common pool of candidate summaries.

Algorithm 3 (A3): $Traverse(...)$

```
 1: Input: $list, v_k \subseteq V, score, PRI_{overlap}, sentence, len$
 2: Output: A set of candidate summaries
 3: $redundancy \leftarrow SizeOf(PRI_{overlap})$
 4: if $redundancy \geq \sigma_r$ then
 5:   if $VEN(v_k)$ then
 6:     if $ValidSentence(sentence)$ then
 7:       $finalScore \leftarrow \frac{score}{len}$
 8:       $AddCandidate(list, sentence, finalScore)$
 9:     end if
10:   end if
11:   for $v_n \in Neighbors_{v_k}$ do
12:     $PRI_{new} \leftarrow PRI_{overlap} \;\bar{\cap}\; PRI_{v_n}$
13:     $redundancy \leftarrow SizeOf(PRI_{new})$
14:     $newSent \leftarrow Concat(sentence, label_{v_n})$
15:     $L \leftarrow len + 1$
16:     $newScore \leftarrow score + PathScore(redundancy, L)$
17:     if $Collapsible(v_n)$ then
18:       $C_{anchor} \leftarrow newSent$
19:       $tmp \leftarrow CreateNewList()$
20:       for $v_x \in Neighbors_{v_n}$ do
21:         Traverse$(tmp, v_x, 0, PRI_{new}, label_{v_x}, L)$
22:         $CC \leftarrow EliminateDuplicates(tmp)$
23:         $CCPathScore \leftarrow AveragePathScore(CC)$
24:         $finalScore \leftarrow newScore + CCPathScore$
25:         $stitchedSent \leftarrow Stitch(C_{anchor}, CC)$
26:         $AddCandidate(list, stitchedSent, finalScore)$
27:       end for
28:     else
29:       Traverse$(list, v_n, newScore, PRI_{new}, newSent, L)$
30:     end if
31:   end for
32: end if
```

(A3:3-4) Algorithm A3 starts with a check to ensure that the minimum path redundancy requirement is satisfied (see definition 4). For the very first node sent from A2, the path redundancy is the size of the raw PRI. (A3:5-10) If the redundancy requirement is satisfied, a few checks are done to determine if a *valid path* has been found. If it has, then the resulting sentence and its final score are added to the list of candidate summaries.

(A3:11-31) Traversal proceeds recursively through the exploration of all neighboring nodes of the current node, v_k. (A3:12-16) For every neighboring node, v_n the *PRI* overlap information, path length, summary sentence and path score are updated before the next recursion. (A3:29) If a v_n is not collapsible, then a regular traversal takes place. (A3:17-27) However, if v_n is collapsible, the updated sentence in A3:14, will now serve as an *anchor* in A3:18. (A3:21) A3 will then attempt to start a recursive traversal from all neighboring nodes of v_n in order to find corresponding collapsed candidates. (A3:22-26) After this, duplicates are eliminated from the *collapsed candidates* and the *collapsed path score* is computed. The resulting *stitched sentence* and its *final score* are then added to the original list of candidate summaries.

(A2:14-18) Once all paths have been explored for candidate generation, duplicate candidates are removed and the remaining are sorted in descending order of their path scores. The best σ_{ss} candidates are 'picked' as final Opinosis summaries.

5 Experimental Setup

We evaluate this abstractive summarization task using reviews of *hotels*, *cars* and various *products*[1]. Based on these reviews, 2 humans were asked to construct 'opinion seeking' queries which would consist of an *entity name* and a *topic of interest*. Example of such queries are: *Amazon Kindle:buttons*, *Holiday Inn, Chicago: staff*, and so on. We compiled a set of 51 such queries. We create one review document per query by collecting all review sentences that contain the query words for the given entity. Each review document thus consists of a set of unordered, redundant review sentences related to the query. There are approximately 100 sentences per review document.

We use ROUGE (Lin, 2004b) to quantitatively assess the agreement of Opinosis summaries with human composed summaries. ROUGE is based on an n-gram co-occurrence between machine summaries and human summaries and is a widely accepted standard for evaluation of summarization tasks. In our experiments, we use ROUGE-1, ROUGE-2 and ROUGE-SU4 measures. ROUGE-1 and ROUGE-2 have been shown to have most correlation with human summaries (Lin and Hovy, 2003) and higher order ROUGE-N scores ($N > 1$) estimate the fluency of summaries.

We use multiple reference (human) summaries in our evaluation since it can achieve better correlation with human judgment (LIN, 2004a). We leverage Amazon's Online Workforce[2] to get 5 different human workers to summarize each review document. The workers were asked to be concise and were asked to summarize the major opinions in the review document presented to them. We manually reviewed each set of reference summaries and dropped summaries that had little or no correlation with the majority. This left us with around 4 reference summaries for each review document.

To allow performance comparison between humans, Opinosis and the baseline method, we implemented a Jackknifing procedure where, given K references, the ROUGE score is computed over K sets of K-1 references. With this, average human performance is computed by treating each reference summary as a 'system' summary, computing ROUGE scores over the remaining K-1 reference summaries.

Due to the limited work in abstractive summarization, no natural baseline could be used for comparison. The existing work in this area is mostly domain dependent and requires too much manual effort (explained in Section 1). The next best baseline is to use a state of the art extractive method. Thus, we use MEAD (Radev et al., 2000) as our baseline. MEAD is an extractive summarizer based on cluster centroids. It uses a collection of the most important words from the whole cluster to select the best sentences for summarization. By default, the scoring of sentences in MEAD is based on 3 parameters - *minimum sentence length*, *centroid*, and *position in text*. MEAD was ideal for our task because a good summary in our case would be one that could capture the most essential information. This is exactly what centroid-based summarization aims to achieve. Also, since the *position in text* parameter is irrelevant in our case, we could easily turn this off with MEAD.

We introduce a **readability test** to understand if Opinosis summaries are in fact readable. Suppose we have N sentences from a system-generated summary and M sentences from corresponding human summaries. We mix all these sentences and then ask a human assessor to pick at most N sentences that are *least readable* as the prediction of system summary.

$$readability(\mathcal{O}) = 1 - \frac{\#CorrectPick}{N}$$

If the human assessor often picks out system generated summaries as being least readable, then the readability of system summaries is poor. If not, then the system generated summaries are no different from human summaries.

6 Results

The baseline method (MEAD) selects 2 most representative sentences as summaries. To give a fair comparison, we fix the Opinosis summary size, $\sigma_{ss} = 2$. We also fix $\sigma_{vsn} = 15$. The best Opinosis configuration with $\sigma_{ss} = 2$ and $\sigma_{vsn} = 15$ is called Opinosis$_{best}$ ($\sigma_{gap} = 4, \sigma_r = 2$, S_{wt_loglen}). ROUGE scores reported are with the use of *stemming* and *stopword removal*.

Performance comparison between humans, Opinosis and baseline. Table 2 shows the performance comparison between humans, Opinosis$_{best}$ and the baseline method. First, we see that the baseline method has very high recall scores compared to Opinosis. This is because extractive methods that just 'select' sentences tend to be much longer resulting in higher recall. However, these summaries tend to carry information that may not be significant and is clearly reflected by the poor

[1] Reviews collected from Tripadvisor, Amazon, Edmunds

[2] https://www.mturk.com

Recall				
	ROUGE-1	ROUGE-2	ROUGE-SU4	Avg # Words
Human	0.3184	**0.1106**	0.1293	17
Opinosis	0.2831	0.0853	0.0851	15
Baseline	**0.4932**	0.1058	**0.2316**	75

Precision				
	ROUGE-1	ROUGE-2	ROUGE-SU4	Avg # Words
Human	0.3434	0.1210	0.1596	17
Opinosis	**0.4482**	**0.1416**	**0.2261**	15
Baseline	0.0916	0.0184	0.0102	75

F-score				
	ROUGE-1	ROUGE-2	ROUGE-SU4	Avg # Words
Human	0.3088	**0.1069**	**0.1142**	17
Opinosis	**0.3271**	0.0998	0.1027	15
Baseline	0.1515	0.0308	0.0189	75

Table 2: Performance comparison between Humans, Opinosis$_{best}$ and Baseline.

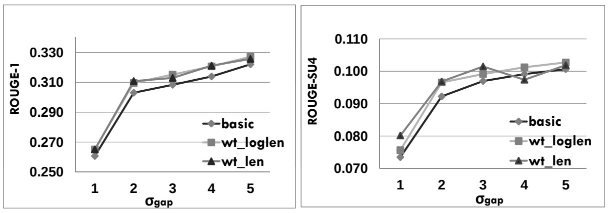


Figure 2: ROUGE scores (f-measure) at different levels of σ_{gap}, $\sigma_r = 2$.

precision scores.

Next, we see that humans have reasonable agreement amongst themselves given that these are independently composed summaries. This agreement is especially clear with the ROUGE-2 recall score where the recall is better than Opinosis but comparable to the baseline even though the summaries are much shorter. It is also clear that Opinosis is closer in performance to humans than to the baseline method. The recall scores of Opinosis summaries are slightly lower than that achieved by humans, while the precision scores are higher (Wilcoxon test shows that the increase in precision is statistically more significant than the decrease in recall). In terms of f-scores, Opinosis has the best ROUGE-1 score and its ROUGE-2 and ROUGE-SU4 scores are comparable with human performance. The baseline method has the lowest f-scores. The difference between the f-scores of Opinosis and that of humans is statistically insignificant.

Comparison of scoring functions. Next, we look into the performance of the three scoring functions, S_{basic}, S_{wt_len} and S_{wt_loglen} described in Section 3. Figure 2 shows ROUGE scores of these scoring methods at varying levels of σ_{gap}. First, it can be observed that S_{wt_basic} which does not use *path length* information, performs the worst. This is due to the effect of heavily favoring redundant paths over longer but reasonably redundant ones that can provide more coverage. We also see that S_{wt_len} and S_{wt_loglen} are similar in performance with S_{wt_loglen} marginally outperforming S_{wt_len} when $\sigma_{gap} > 2$. Since S_{wt_len} uses the raw *path length* in its scoring function, it may be inflating the path scores of long but insignificant paths. S_{wt_loglen} scales down the path length, thus providing a reasonable tradeoff between redundancy and the length of the selected path. The three scoring functions are not influenced by different levels of σ_r as shown in Figure 3.

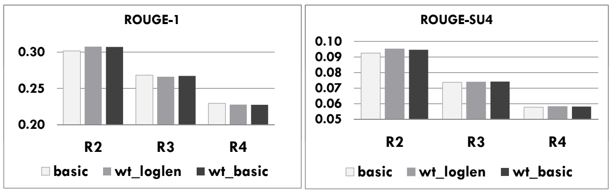


Figure 3: ROUGE scores (f-measure) at different levels of σ_r averaged across $\sigma_{gap} \in [1, 5]$

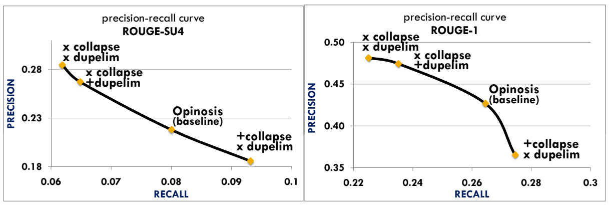
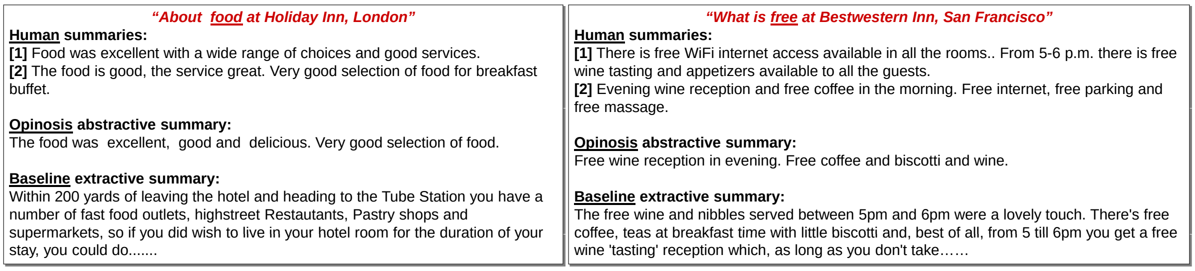


Figure 4: Precision-Recall comparison with different Opinosis features turned off.

Effect of gap setting (σ_{gap})**.** Now, we will examine the effect of σ_{gap} on the generated summaries. Based on Figure 2, we see that setting σ_{gap}=1 yields in relatively low performance. This is because σ_{gap}=1 implies immediate adjacency between the PIDs of two nodes and such strict adjacency enforcements prevent redundancies from being discovered. When σ_{gap} is increased to 2, there is a big jump in performance, after which improvements are observed in smaller amounts. A very large gap setting could increase the possibility of generating ill-formed sentences, thus we recommend that σ_{gap} is set between 2-5.

Effect of redundancy requirement (σ_r) **.** Figure 3 shows the ROUGE scores at different levels of σ_r. It is clear that when $\sigma_r > 2$, the quality of summaries is negatively impacted. Since we only have about 100 sentences per review document, $\sigma_r > 2$ severely restricts the number of paths that can be explored, yielding in lower ROUGE scores. Since the scoring function can account for the level of redundancy, σ_r should be set according to the size of the input data. For our dataset, $\sigma_r = 2$ was ideal.



"About food at Holiday Inn, London"

Human summaries:

[1] Food was excellent with a wide range of choices and good services.

[2] The food is good, the service great. Very good selection of food for breakfast buffet.

Opinosis abstractive summary:

The food was excellent, good and delicious. Very good selection of food.

Baseline extractive summary:

Within 200 yards of leaving the hotel and heading to the Tube Station you have a number of fast food outlets, highstreet Restautants, Pastry shops and supermarkets, so if you did wish to live in your hotel room for the duration of your stay, you could do.......

"What is free at Bestwestern Inn, San Francisco"

Human summaries:

[1] There is free WiFi internet access available in all the rooms.. From 5-6 p.m. there is free wine tasting and appetizers available to all the guests.

[2] Evening wine reception and free coffee in the morning. Free internet, free parking and free massage.

Opinosis abstractive summary:

Free wine reception in evening. Free coffee and biscotti and wine.

Baseline extractive summary:

The free wine and nibbles served between 5pm and 6pm were a lovely touch. There's free coffee, teas at breakfast time with little biscotti and, best of all, from 5 till 6pm you get a free wine 'tasting' reception which, as long as you don't take……

Figure 5: Sample results comparing Opinosis summaries with human and baseline summaries.

Effect of collapsed structures and duplicate elimination. So far, it has been assumed that all features used in Opinosis are required to generate reasonable summaries. To test this hypothesis, we use Opinosis$_{best}$ as a baseline and then we turn off different features of Opinosis. We turn off the *duplicate elimination feature*, then the *collapsible structure feature*, and finally *both*. Figure 4 shows the resulting precision-recall curve. From this graph, we see that without duplicate elimination and when collapsing is turned off, the precision is highest but recall is lowest. No collapsing implies shorter sentences and thus lower recall, which is clearly reflected in Figure 4. On top of this, if duplicates are allowed, the overall information coverage is low, further affecting the recall. Notice that the presence of duplicates with the collapse feature turned on results in very high recall (even higher than the baseline). This is caused by the presence of similar phrases that were not eliminated from the collapsed candidates, resulting in long sentences that artificially boost recall. The Opinosis baseline which uses duplicate elimination and the collapsible structure feature, offers a reasonable tradeoff between precision and recall.

Readability of Summaries. To test the readability of Opinosis summaries, we conducted a *readability test* (described in Section 5) using summaries generated from Opinosis$_{best}$. A human assessor picked the 2 least readable sentences from each of the 51 test sets (based on 51 summaries). Collectively, there were 565 sentences out of which 102 were Opinosis generated. Out of these, the human assessor picked only 34 of the sentences as being least readable, resulting in an average readability score of 0.67. This shows that more than 60% of the generated sentences are indistinguishable from human composed sentences. Of the 34 sentences with problems, 11 contained *no information* or were *incomprehensible*, 12 were *incomplete* possibly due to false positives when the sentence validity check was done, and 8 had *conflicting information* such as '*the hotel room is clean and dirty*'. This happens due to mixed feelings about the same topic and can be resolved using sentiment analysis. The remaining 3 sentences were found to contain *poor grammar*, possibly caused by the gaps allowed in finding redundant paths.

Sample Summaries. Finally, in Figure 5 we show two sample summaries on two different topics. Notice that the Opinosis summaries are concise, fairly well-formed and have closer resemblance to human summaries than to the baseline summaries.

7 Conclusion

In this paper, we described a novel summarization framework (Opinosis) that uses textual graphs to generate abstractive summaries of highly redundant opinions. Evaluation results on a set of review documents show that Opinosis summaries have better agreement with human summaries compared to the baseline extractive method. The Opinosis summaries are concise, reasonably well-formed and communicate essential information. Our readability test shows that more than 60% of the generated sentences are no different from human composed sentences.

Opinosis is a flexible framework in that many of its modules can be easily improved or replaced with other suitable implementation. Also, since Opinosis is domain independent and relies on minimal external resources, it can be used with any corpus containing high amounts of redundancies.

Our graph representation naturally ensures the coherence of a summary, but such a graph emphasizes too much on the surface order of words. As a result, it cannot group sentences at a deep semantic level. To address this limitation, we can use a similar idea to overlay parse trees and this would be a very interesting future research.

8 Acknowledgments

We thank the anonymous reviewers for their useful comments. This paper is based upon work supported in part by an IBM Faculty Award, an Alfred P. Sloan Research Fellowship, an AFOSR MURI Grant FA9550-08-1-0265, and by the National Science Foundation under grants IIS-0347933, IIS-0713581, IIS-0713571, and CNS-0834709.

References

[Barzilay and Lee2003] Barzilay, Regina and Lillian Lee. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 16–23, Morristown, NJ, USA. Association for Computational Linguistics.

[DeJong1982] DeJong, Gerald F. 1982. An overview of the FRUMP system. In Lehnert, Wendy G. and Martin H. Ringle, editors, *Strategies for Natural Language Processing*, pages 149–176. Lawrence Erlbaum, Hillsdale, NJ.

[Erkan and Radev2004] Erkan, Günes and Dragomir R. Radev. 2004. Lexrank: graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479.

[Finley and Harabagiu2002] Finley, Sanda Harabagiu and Sanda M. Harabagiu. 2002. Generating single and multi-document summaries with gistexter. In *Proceedings of the workshop on automatic summarization*, pages 30–38.

[Jing and McKeown2000] Jing, Hongyan and Kathleen R. McKeown. 2000. Cut and paste based text summarization. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 178–185, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Lerman et al.2009] Lerman, Kevin, Sasha Blair-Goldensohn, and Ryan Mcdonald. 2009. Sentiment summarization: Evaluating and learning user preferences. In *12th Conference of the European Chapter of the Association for Computational Linguistics (EACL-09)*.

[Lin and Hovy2003] Lin, Chin-Yew and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proc. HLT-NAACL*, page 8 pages.

[LIN2004a] LIN, Chin-Yew. 2004a. Looking for a few good metrics : Rouge and its evaluation. *proc. of the 4th NTCIR Workshops, 2004*.

[Lin2004b] Lin, Chin-Yew. 2004b. Rouge: a package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004), Barcelona, Spain*.

[Lu et al.2009] Lu, Yue, ChengXiang Zhai, and Neel Sundaresan. 2009. Rated aspect summarization of short comments. In *18th International World Wide Web Conference (WWW2009)*, April.

[Mihalcea and Tarau2004] Mihalcea, R. and P. Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*, July.

[Pang and Lee2004] Pang, Bo and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278.

[Pang et al.2002] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86.

[Radev and McKeown1998] Radev, DR and K. McKeown. 1998. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):469–500.

[Radev et al.2000] Radev, Dragomir, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: Sentence extraction, utility-based evaluation, and user studies. In *In ANLP/NAACL Workshop on Summarization*, pages 21–29.

[Radev et al.2002] Radev, Dragomir R., Eduard Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization.

[Saggion and Lapalme2002] Saggion, Horacio and Guy Lapalme. 2002. Generating indicative-informative summaries with sumum. *Computational Linguistics*, 28(4):497–526.

[Snyder and Barzilay2007] Snyder, Benjamin and Regina Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL*, pages 300–307.

[Titov and Mcdonald2008] Titov, Ivan and Ryan Mcdonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL-08: HLT*, pages 308–316, Columbus, Ohio, June. Association for Computational Linguistics.

EMDC: A Semi-supervised Approach for Word Alignment

Qin Gao
Language Technologies Institute
Carnegie Mellon University
qing@cs.cmu.edu

Francisco Guzman
Centro de Sistemas Inteligentes
Tecnológico de Monterrey
guzmanhe@gmail.com

Stephan Vogel
Language Technologies Institute
Carnegie Mellon University
stephan.vogel@cs.cmu.edu

Abstract

This paper proposes a novel semi-supervised word alignment technique called EMDC that integrates discriminative and generative methods. A discriminative aligner is used to find high precision partial alignments that serve as constraints for a generative aligner which implements a constrained version of the EM algorithm. Experiments on small-size Chinese and Arabic tasks show consistent improvements on AER. We also experimented with moderate-size Chinese machine translation tasks and got an average of 0.5 point improvement on BLEU scores across five standard NIST test sets and four other test sets.

1 Introduction

Word alignment is a crucial component in statistical machine translation (SMT). From a Machine Learning perspective, the models for word alignment can be roughly categorized as generative models and discriminative models. The widely used word alignment tool, i.e. GIZA++ (Och and Ney, 2003), implements the well-known IBM models (Brown et al., 1993) and the HMM model (Vogel et al., 1996), which are generative models. For language pairs such as Chinese-English, the word alignment quality is often unsatisfactory. There has been increasing interest on using manual alignments in word alignment tasks, which has resulted in several discriminative models. Ittycheriah and Roukos (2005) proposed to use only manual alignment links in a maximum entropy model, which is considered supervised. Also, a number of semi-supervised word aligners have been proposed (Taskar et al., 2005; Liu et al., 2005; Moore, 2005; Blunsom and Cohn, 2006; Niehues and Vogel, 2008). These methods use held-out manual alignments to tune weights for discriminative models, while using the model parameters, model scores or alignment links from unsupervised word aligners as features. Callison-Burch et. al. (2004) proposed a method to interpolate the parameters estimated by sentence-aligned and word-aligned corpus. Also, there are recent attempts to combine multiple alignment sources using alignment confidence measures so as to improve the alignment quality (Huang, 2009).

In this paper, the question we address is whether we can jointly improve discriminative models and generative models by feeding the information we get from the discriminative aligner back into the generative aligner. Examples of this line of research include Model 6 (Och and Ney, 2003) and the EMD training approach proposed by Fraser and Marcu (2006) and its extension called LEAF aligner (Fraser and Marcu, 2007). These approaches use labeled data to tune additional parameters to weight different components of the IBM models such as the lexical translation model, the distortion model and the fertility model. These methods are proven to be effective in improving the quality of alignments. However, the discriminative training in these methods is restricted in using the model components of generative models, in other words, incorporating new features is difficult.

Instead of using discriminative training methods to tune the weights of generative models, in this paper we propose to use a discriminative word aligner to produce reliable constraints for the EM algorithm. We call this new training scheme EMDC (**E**xpectation-**M**aximization-**D**iscrimination-**C**onstraint). The methodology can be viewed as a variation of bootstrapping. It enables the generative models to interact with discriminative models at the data level instead of the model level. Furthermore, with a discriminative word aligner that uses generative word aligner's output as features, we create a feedback loop that can iteratively improve the quality of both aligners. The major contributions of this paper are: 1) The EMDC training scheme, which ties the generative and discriminative aligners together and enables future research on integrating other discriminative aligners. 2) An extended generative aligner based on GIZA++ that allows to perform constrained EM training.

In Section 2, we present the EMDC training scheme. Section 3 provides details of the constrained EM algorithm. In Section 4, we introduce the discriminative aligner and link filtering. Section 5 provides the experiment set-up and the results. Section 6 concludes the paper.

2 EMDC Training Scheme

The EMDC training scheme consists of three parts, namely **EM**, **D**iscrimination, and **C**onstraints. As illustrated in Figure 1, a large unlabeled training set is first aligned with a generative aligner (GIZA++ for the purpose of this paper). The generative aligner outputs the model parameters and the Viterbi alignments for both source-to-target and target-to-source directions. Afterwards, a discriminative aligner (we use the one described in (Niehues and Vogel, 2008)), takes the lexical translation model, fertility model and Viterbi alignments from both directions as features, and is tuned to optimize the AER on a small manually aligned tuning set. Afterwards, the alignment links generated by the discriminative aligner are filtered according to their likelihood, resulting in a subset of links that has high precision and low recall. The next step is to put these high precision alignment links back into the generative aligner as constraints. A conventional generative word aligner does not support this type of constraints. Thus we developed a constrained EM algorithm that can use the links from a partial alignment as constraints and estimate the model parameters by marginalizing likelihoods.

After the constrained EM training is performed, we repeat the procedure and put the *updated* generative models and Viterbi alignment back into the discriminative aligner. We can either fix the number of iterations, or stop the procedure when the gain on AER of a small held-out test set drops below a threshold.

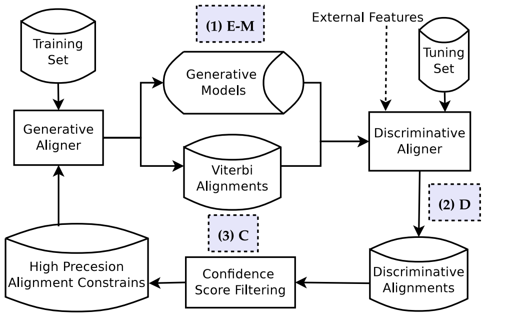


Figure 1: Illustration of EMDC training scheme

The key components for the system are:

1. A generative aligner that can make use of reliable alignment links as constraints and improve the models/alignments.
2. A discriminative aligner that outputs confidence scores for alignment links, which allows to obtain high-precision-low-recall alignments.

While in this paper we derive the reliable links by filtering the alignment generated by a discriminative aligner, such partial alignments may be obtained from other sources as well: manual alignments, specific named entity aligner, noun-phrase aligner, etc.

As we mentioned in Section 1, the discriminative aligner is not restricted to use features parameters of generative models and Viterbi alignments. However, including the features from generative models is required for iterative training, because the improvement on the quality of these features can in turn improve the discriminative aligner. In our experiments, the discriminative aligner makes heavy use of the Viterbi alignment and the model parameters from the generative aligner. Nonetheless, one can easily replace the discriminative aligner or add new features to it without modifying the training scheme. The open-ended property of the training scheme makes it a promising method to integrate different aligners.

In the next two sections, we will describe the key components of this framework in detail.

3 Constrained EM algorithm

In this section we will briefly introduce the constrained EM algorithm we used in the experiment, further details of the algorithm can be found in (Gao et al., 2010).

The IBM Models (Brown et al., 1993) are a series of generative models for word alignment. GIZA++ (Och and Ney, 2003), the most widely used implementation of IBM models and HMM (Vogel et al., 1996), employs EM algorithm to estimate the model parameters. For simpler models such as Model 1 and Model 2, it is possible to obtain sufficient statistics from all possible alignments in the E-step. However, for fertility-based models such as Models 3, 4, and 5, enumerating all possible alignments is NP-complete. To overcome this limitation, GIZA++ adopts a greedy hill-climbing algorithm, which uses simpler models such as HMM or Model 2 to generate a "center alignment" and then tries to find better alignments among its neighbors. The neighbors of an alignment $a_1^J = [a_1, a_2, \cdots, a_J]$ with $a_j \in [0, I]$ are defined as alignments that can be generated from a_1^J by one of the following two operators:

1. The move operator $m_{[i,j]}$, that changes $a_j := i$, i.e. arbitrarily sets word f_j in the target sentence to align to the word e_i in source sentence;
2. The swap operator $s_{[j_1,j_2]}$ that exchanges a_{j_1} and a_{j_2}.

The algorithm will update the center alignment as long as a better alignment can be found, and finally outputs a local optimal alignment. The neighbor alignments of the final center alignment are then used in collecting the counts for the M-Step. Och and Ney (2003) proposed a fast implementation of the hill-climbing algorithm that employs two matrices, i.e. Moving Matrix $M_{I \times J}$ and Swapping Matrix $S_{J \times J}$. Each cell of the matrices stores the value of likelihood difference after applying the corresponding operator.

We define a partial alignment constraint of a sentence pair (f_1^J, e_1^I) as a set of links: $\alpha_I^J = \{(i,j) | 0 \leq i < I, 0 \leq j < J\}$. Given a set of constraints, an alignment $a_1^J = [a_1, a_2, \cdots, a_j]$ on the sentence pair f_1^J, e_1^I, the translation probability of $Pr(f_1^J | e_1^I)$ will be zero if the alignment is inconsistent with the constraints. Constraints $(0, j)$ or $(i, 0)$ are used to explicitly represent that word f_j or e_i is aligned to the empty word.

Under the assumptions of the IBM models, there are two situations that a_1^J is inconsistent with α_I^J:

1. Target word misalignment: The IBM models assume that one target word can only be aligned to one source word. Therefore, if the target word f_j aligns to a source word e_i, while the constraint α_I^J suggests f_j should be aligned to $e_{i'}$, the alignment violates the constraint and thus is considered inconsistent.
2. Source word to empty word misalignment: if a source word is aligned to the empty word, it cannot be aligned to any concrete target word.

However, the partial alignments, which allow n-to-n alignments, may already violate the 1-to-n alignment restriction of the IBM models. In these cases, we relax the condition in situation 1 that if the alignment link a_{j^*} is consistent with any one of the conflicting target-to-source constraints, it will be considered consistent. Also, we arbitrarily assign the source word to empty word constraints higher priorities than other constraints, because unlike situation 1, it does not have the problem of conflicting with other constraints.

3.1 Constrained hill-climbing algorithm

To ensure that resulting center alignment be consistent with the constraints, we need to split the hill-climbing algorithm into two stages: 1) optimize towards the constraints and 2) optimize towards the optimal alignment under the constraints.

From a seed alignment, we first move the alignment towards the constraints by choosing a move or swap operator that:

1. produces the alignment that has the highest likelihood among alignments generated by other operators,
2. eliminates at least one inconsistent link.

We iteratively update the alignment until no other inconsistent link can be removed. The algorithm implies that we force the seed alignment to be closer to the constraints while trying to find the best consistent alignment. Figure 2 demonstrates the idea, given the constraints shown in (a), and the seed alignment shown as solid links in (b), we

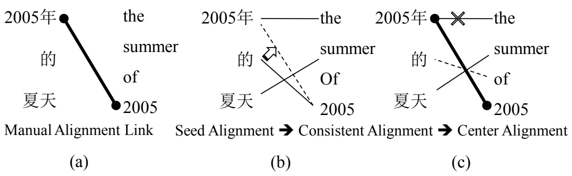


Figure 2: Illustration of Algorithm 1

move the inconsistent link to the dashed link by a move operation.

After we find the consistent alignment, we proceed to optimize towards the optimal alignment under the constraints. The algorithm sets the value of the cells in moving/swapping matrices to negative if the corresponding operators will lead to an inconsistent alignment. The moving matrix needs to be processed only once, whereas the swapping matrix needs to be updated every iteration, since once the alignment is updated, the possible violations will also change.

If a source word e_i is aligned to the empty word, we set $M_{i,j} = -1, \forall j$. The swapping matrix does not need to be modified in this case because the swapping operator will not introduce new links.

Because the cells that can lead to violations are set to negative, the operators will never be picked when updating the center alignments. This ensures the consistency of the final center alignment.

3.2 Count Collection

After finding the center alignment, we need to collect counts from neighbor alignments so that the M-step can normalize the counts to produce the model parameters for the next step. In this stage, we want to make sure all the inconsistent alignments in the neighbor set of the center alignment be ruled out from the sufficient statistics, i.e. have zero probability. Similar to the constrained hill climbing algorithm, we can manipulate the moving/swapping matrices to effectively exclude inconsistent alignments. Since the original count collection algorithm depends only on moving and swapping matrices, we just need to bypass all the cells which hold negative values, i.e. represent inconsistent alignments.

We can also view the algorithm as forcing the posteriors of inconsistent alignments to zero, and therefore increase the posteriors of consistent alignments. When no constraint is given, the algorithm falls back to conventional EM, and when all the alignments are known, the algorithm becomes fully supervised. And if the alignment quality can be improved if high-precision partial alignment links is given as constraints. In (Gao et al., 2010) we experimented with using a dictionary to generate such constraints, and in (Gao and Vogel, 2010) we experimented with manual word alignments from Mechanical Turk. And in this paper we try to use an alternative method that uses a discriminative aligner and link filtering to generate such constraints.

4 Discriminative Aligner and Link Filtering

We employ the CRF-based discriminative word aligner described in (Niehues and Vogel, 2008). The aligner can use a variety of knowledge sources as features, such as: the fertility and lexical translation model parameters from GIZA++, the Viterbi alignment from both source-to-target and target-to-source directions. It can also make use of first-order features which model the dependency between different links, the Parts-of-Speech tagging features, the word form similarity feature and the phrase features. In this paper we use all the features mentioned above except the POS and phrase features.

The aligner is trained using a belief-propagation (BP) algorithm, and can be optimized to maximize likelihood or directly optimize towards AER on a tuning set. The aligner outputs confidence scores for alignment links, which allows us to control the precision and recall rate of the resulting alignment. Guzman et al. (2009) experimented with different alignments produced by adjusting the filtering threshold for the alignment links and showed that they could get high-precision-low-recall alignments by having a higher threshold. Therefore, we replicated the confidence filtering procedures to produce the partial alignment constraints. Afterwards we iterate by putting the partial alignments back to the constrained word alignment algorithm described in section 3.

Although the discriminative aligner performs well in supplying high precision constraints, it does not model the null alignment explicitly.

	Num. of Sentences	**Num. of Words** *Source*	*Target*	**Num. of Links**
Ch-En	21,863	424,683	524,882	687,247
Ar-En	29,876	630,101	821,938	830,349

Table 1: Corpus statistics of the manual aligned corpora

Threshold		**P**	**R**	**AER**
Ch-En	0.6	71.30	58.12	35.96
	0.7	75.24	54.03	37.11
	0.8	85.66	44.19	41.70
	0.9	93.70	37.95	45.98
Ar-En	0.6	72.35	59.87	34.48
	0.7	77.55	55.58	35.25
	0.8	80.07	50.89	37.77
	0.9	83.74	44.16	42.17

Table 2: The qualities of the constraints

Hence we are currently not able to provide source word to empty word alignment constraints which have been proven to be effective in improving the alignment quality in (Gao et al., 2010). Due to space limitation, please refer to: (Niehues and Vogel, 2008; Guzman et al., 2009) for further details of the aligner and link filtering, respectively.

5 Experiments

To validate the proposed training scheme, we performed two sets of experiments. First of all, we experimented with a small manually aligned corpus to evaluate the ability of the algorithm to improve the AER. The experiment was performed on Chinese to English and Arabic to English tasks. Secondly, we experimented with a moderate size corpus and performed translation tasks to observe the effects in translation quality.

5.1 Effects on AER

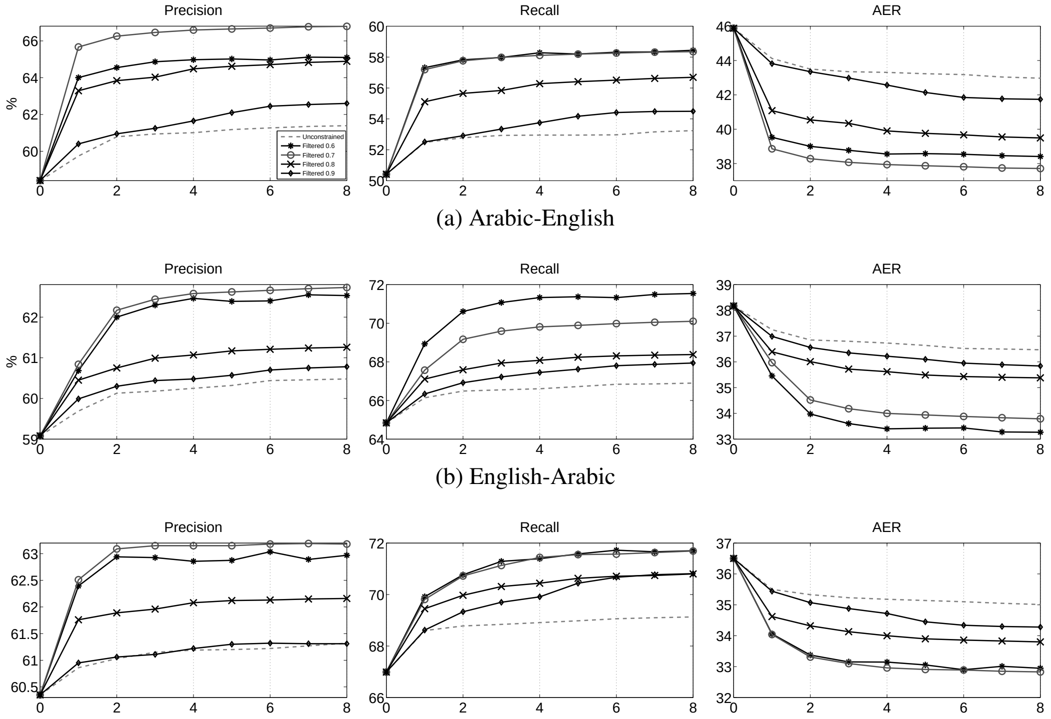
In order to measure the effects of EMDC in alignment quality, we experimented with Chinese-English and Arabic-English manually aligned corpora. The statistics of these sets are shown in Table 1. We split the data into two fragments, the first 100 sentences (Set A) and the remaining (Set B). We trained generative IBM models using the Set B, and tuned the discriminative aligner using the Set A. We evaluated the AER on Set B, but in any of the training steps the manual alignments of Set B were not used.

In each iteration of EDMC, we load the model parameters from the previous step and continue training using the new constraints. Therefore, it is important to compare the performance of continuous training against an unconstrained baseline, because variation in alignment quality could be attributed to either the effect of more training iterations or to the effect of semi-supervised training scheme. In Figures 3 and 4 we show the alignment quality for each iteration. Iteration 0 is the baseline, which comes from standard GIZA++ training[1]. The grey dash curves represent unconstrained Model 4 training, and the curves with start, circle, cross and diamond markers are constrained EM alignments with 0.6, 0.7, 0.8 and 0.9 filtering thresholds respectively. As we can see from the results, when comparing only the mono-directional trainings, the alignment qualities improve over the unconstrained training in all the metrics (precision, recall and AER). From Table 2, we observe that the quality of discriminative aligner also improved. Nonetheless, when we consider the heuristically symmetrized alignment[2], we observe mixed results. For instance, for the Chinese-English case we observe that AER improves over iterations, but this is the result of a increasingly higher recall rate in detriment of precision. Ayan and Dorr (2006) pointed out that *grow-diag-final* symmetrization tends to output alignments with high recall and low precision. However this does not fully explain the tendency we observed between iterations. The characteristics of the alignment modified by EDMC that lead to larger improvements in mono-directional trainings but a precision drop with symmetrization heuristics needs to be addressed in future work.

Another observation is how the filtering thresholds affect the results. As we can see in Table 3, for Chinese to English word alignment, the largest gain on the alignment quality is observed when the threshold was set to 0.8, while for Arabic to English, the threshold of 0.7 or 0.6 works better. Table 2 shows the precision, recall, and AER of the constraint links used in the constrained EM al-

[1] We run 5, 5, 3, 3 iterations of Model 1, HMM, Model 3 and Model 4 respectively.

[2] We used grow-diag-final-and



(a) Arabic-English

(b) English-Arabic

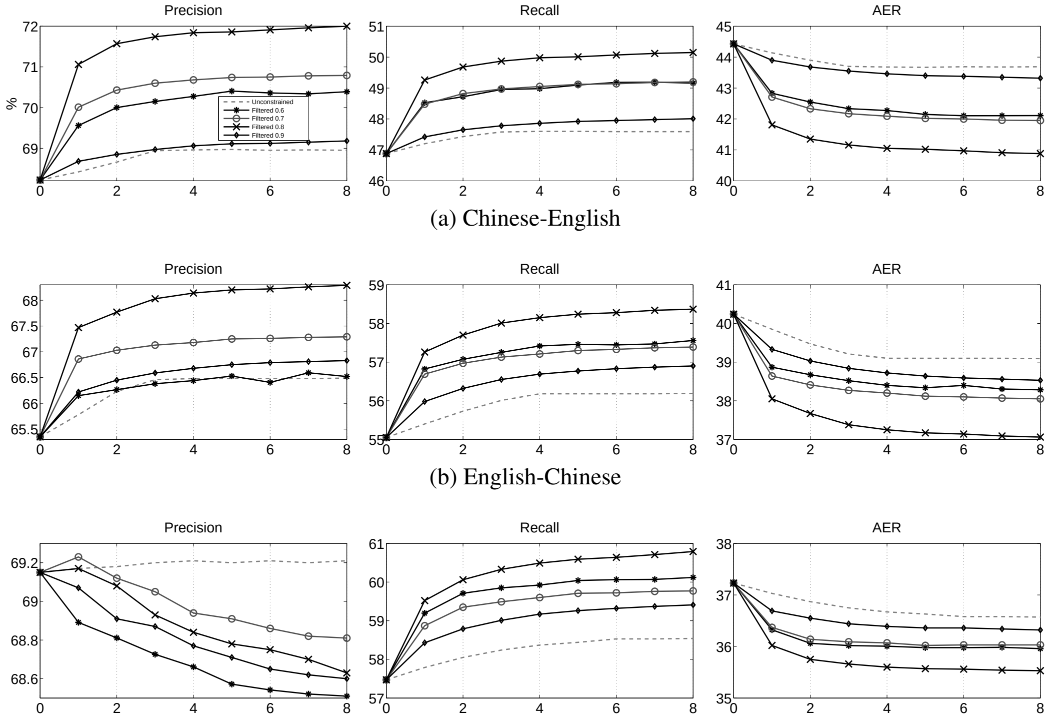
(c) Heuristically-symmetrized

Figure 3: Alignment qualities of each iteration for Arabic-English word alignment task. The grey dash curves represent unconstrained Model 4 training, and the curves with star, circle, cross and diamond markers are constrained EM alignments with 0.6, 0.7, 0.8 and 0.9 filtering thresholds respectively.

		Source-Target			**Target-Source**			**Heuristic**			**Discriminative**		
		P	R	AER	P	R	AER	P	R	AER	P	R	AER
Ch	BL	68.22	46.88	44.43	65.35	55.05	40.25	69.15	57.47	37.23	67.45	59.77	36.62
	NC	+0.73	+0.71	-0.74	+1.14	+1.14	-1.15	**+0.06**	+1.07	-0.66	+0.15	+0.64	-0.42
	0.6	+2.17	+2.28	-2.32	+1.17	+2.51	-1.97	-0.64	+2.65	-1.27	-0.39	+1.89	-0.87
	0.7	+2.57	+2.32	-2.48	+1.94	+2.34	-2.19	*-0.34*	+2.30	-1.20	*-0.28*	+1.60	-0.76
	0.8	**+3.78**	**+3.27**	**-3.55**	**+2.94**	**+3.32**	**-3.18**	*-0.52*	**+3.32**	**-1.70**	**+0.69**	**+0.14**	**-0.89**
	0.9	+0.98	+1.13	-1.11	+1.48	+1.85	-1.71	*-0.55*	+1.94	-0.90	*-0.58*	+1.45	-0.54
Ar	BL	58.41	50.42	45.88	59.08	64.84	38.17	60.35	66.99	36.50	68.93	63.94	33.66
	NC	+2.98	+2.92	-2.96	+1.40	+2.06	-1.70	+0.97	+2.14	-1.49	*-0.87*	+2.37	-0.83
	0.6	+6.69	**+8.02**	-7.47	+3.45	**+6.70**	**-4.90**	+2.62	**+4.71**	-3.55	+0.58	-0.55	+0.03
	0.7	**+8.38**	+7.93	**-8.16**	**+3.65**	+5.26	-4.38	**+2.83**	+4.70	**-3.67**	**+2.46**	*-0.42*	-0.88
	0.8	+6.48	+6.27	-6.39	+2.18	+3.54	-2.80	+1.81	+3.81	-2.70	+1.67	+2.30	-2.01
	0.9	+4.02	+4.07	-4.07	+1.70	+3.10	-2.33	+0.62	+3.82	-2.03	+1.33	**+2.70**	**-2.06**

Table 3: Improvement on word alignment quality on small corpus after 8 iterations. BL stands for baseline, and NC represents unconstrained Model 4 training, and 0.9, 0.8, 0.7, 0.6 are the thresholds used in alignment link filtering.

gorithm, the numbers are averaged across all iterations, the actual numbers of each iteration only have small differences. Although one might expect that the quality of resulting alignment from constrained EM be proportional to the quality of constraints, from the numbers in Table 2 and 3, we are not able to induce a clear relationship between them, and it could be language- or corpus-dependent. However, in practice we nonetheless use a held-out test set to tune this parameter. The



(a) Chinese-English

(b) English-Chinese

(c) Heuristically-symmetrized

Figure 4: Alignment qualities of each iteration for Chinese-English word alignment task. The grey dash curves represent unconstrained Model 4 training, and the curves with star, circle, cross and diamond markers are constrained EM alignments with 0.6, 0.7, 0.8 and 0.9 filtering thresholds respectively.

	Ch-En			**En-Ch**			**Heuristic**			**Discriminative**		
	P	R	AER	P	R	AER	P	R	AER	P	R	AER
BL	73.51	50.14	40.38	68.82	57.66	37.31	72.98	60.23	34.01	72.10	61.63	33.55
NC	73.23	50.38	40.30	68.30	58.00	37.27	72.39	60.99	33.80	72.07	61.81	33.45
0.8	76.27	52.90	37.53	70.26	60.26	35.11	72.75	63.49	32.19	72.64	63.29	32.35

Table 4: Improvement on word alignment quality on moderate-size corpus, where BL and NC represents baseline and non-constrained Model 4 training

relationship between quality of constraints and alignment results is an interesting topic for future research.

5.2 Effects on translation quality

In this experiment we run the whole machine translation pipeline and evaluate the system on BLEU score. We used the corpus LDC2006G05 which contains 25 million words as training set, the same discriminative tuning set as previously used (100 sentence pairs) and the remaining 21,763 sentence pairs from the hand-aligned corpus of the previous experiment are held-out test set for alignment qualities. A 4-gram language model trained from English GigaWord V1 and V2 corpus was used. The AER scores on the held-out test set are also provided for every iteration. Based on the observation in last experiment, we adopt the filtering threshold of 0.8.

Similar to previous experiment, the heuristically symmetrized alignments have lower precisions than their EMDC counterparts, however the gaps are smaller as shown in Table 4. We observe 2.85 and 2.21 absolute AER reduction on two directions, after symmetrization the gain on AER is 1.82. Continuing Model 4 training appears to have minimal effect on AER, and the improve-

I	M	NIST							GALE				
		mt06	mt02	mt03	mt04	mt05	mt08	*ain*	db-nw	db-wb	dd-nw	dd-wb	*aia*
0	G	31.00	31.80	29.89	32.63	29.33	24.24		26.92	24.48	28.44	24.26	
1	D	30.65	31.60	30.04	32.89	29.34	24.52	0.12	27.43	24.72	28.32	24.30	0.14
	G	31.35	31.91	30.35	32.75	29.40	24.16	0.15	27.39	24.50	28.22	24.60	0.15
2	D	**31.61**	32.31	30.40	33.06	29.49	24.11	0.33	**28.17**	24.42	28.58	24.36	0.34
	G	31.14	31.94	30.42	32.86	29.49	24.15	0.20	27.31	24.51	27.50	24.02	0.03
3	D	31.29	**32.39**	30.28	33.19	29.60	24.41	0.43	27.64	**25.32**	28.55	24.71	0.47
	G	30.94	31.95	30.15	32.71	29.38	24.22	0.12	27.63	24.61	28.80	25.05	0.29
4	D	30.80	32.04	30.51	33.24	29.49	24.61	0.46	27.61	25.27	28.72	24.98	0.53
	G	30.68	31.81	30.33	33.05	29.28	24.41	0.26	27.20	24.79	28.43	24.50	0.24
5	D	30.93	31.89	29.96	32.89	29.37	**24.50**	0.17	27.75	24.50	**29.05**	24.90	0.33
	G	31.16	32.28	**30.72**	**33.30**	**29.83**	24.30	**0.51**	27.32	25.05	28.60	**25.44**	**0.54**

Table 5: Improvement on translation alignment quality on moderate-size corpus, The column *ain* shows the average improvement of BLEU scores for all NIST test sets (excluding the tuning set MT06), and column *aia* is the average improvement on all unseen test sets. The column *M* indicates the alignment source, *G* means the alignment comes from generative aligner, and *D* means discriminative aligner respectively. The number of iterations is shown in column *I*.

ment mainly comes from the constraints.

In the experiment, we use the Moses toolkit to extract phrases, tune parameters and decode. We use the NIST MT06 test set as the tuning set, NIST MT02-05 and MT08 as unseen test sets. We also include results for four additional unseen test sets used in GALE evaluations: DEV07-Dev newswire part (dd-nw, 278 sentences) and Weblog part (dd-wb, 345 sentences), Dev07-Blind newswire part (db-nw, 276 sentences and Weblog part (db-wb, 312 sentences). Table 5 presents the average improvement on BLEU scores in each iteration. As we can see from the results, in all iterations we got improvement on BLEU scores, and the largest gain we have gotten is on the fifth iteration, which has 0.51 average improvement on five NIST test sets, and 0.54 average improvement across all nine test sets.

6 Conclusion

In this paper we presented a novel training scheme for word alignment task called EMDC. We also presented an extension of GIZA++ that can perform constrained EM training. By integrating it with a CRF-based discriminative word aligner and alignment link filtering, we can improve the alignment quality of both aligners iteratively. We experimented with small-size Chinese-English and Arabic English and moderate-size Chinese-English word alignment tasks, and observed in all four mono-directional alignments more than 3% absolute reduction on AER, with the largest improvement being 8.16% absolute on Arabic-to-English comparing to the baseline, and 5.90% comparing to Model 4 training with the same numbers of iterations. On a moderate-size Chinese-to-English tasks we also evaluated the impact of the improved alignment on translation quality across nine test sets. The 2% absolute AER reduction resulted in 0.5 average improvement on BLEU score.

Observations on the results raise several interesting questions for future research, such as 1) What is the relationship between the precision of the constraints and the quality of resulting alignments after iterations, 2) The effect of using different discriminative aligners, 3) Using aligners that explicitly model empty words and null alignments to provide additional constraints. We will continue exploration on these directions.

The extended GIZA++ is released to the research community as a branch of MGIZA++ (Gao and Vogel, 2008), which is available online[3].

Acknowledgement

This work is supported by NSF CluE Project (NSF 08-560) and DARPA GALE project.

[3] Accessible on Source Forge, with the URL: `http://sourceforge.net/projects/mgizapp/`

References

Ayan, Necip Fazil and Bonnie J. Dorr. 2006. Going beyond aer: an extensive analysis of word alignments and their impact on mt. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 9–16.

Blunsom, Phil and Trevor Cohn. 2006. Discriminative word alignment with conditional random fields. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 65–72.

Brown, Peter F., Vincent J.Della Pietra, Stephen A. Della Pietra, and Robert. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. In *Computational Linguistics*, volume 19(2), pages 263–331.

Callison-Burch, C., D. Talbot, and M. Osborne. 2004. Statistical machine translation with word- and sentence-aligned parallel corpora. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 175–183.

Fraser, Alexander and Daniel Marcu. 2006. Semi-supervised training for statistical word alignment. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 769–776.

Fraser, Alexander and Daniel Marcu. 2007. Getting the structure right for word alignment: LEAF. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 51–60.

Gao, Qin and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Proceedings of the ACL 2008 Software Engineering, Testing, and Quality Assurance Workshop*, pages 49–57.

Gao, Qin and Stephan Vogel. 2010. Consensus versus expertise : A case study of word alignment with mechanical turk. In *NAACL 2010 Workshop on Creating Speech and Language Data With Mechanical Turk*, pages 30–34.

Gao, Qin, Nguyen Bach, and Stephan Vogel. 2010. A semi-supervised word alignment algorithm with partial manual alignments. In *In Proceedings of the ACL 2010 joint Fifth Workshop on Statistical Machine Translation and Metrics MATR (ACL-2010 WMT)*.

Guzman, Francisco, Qin Gao, and Stephan Vogel. 2009. Reassessment of the role of phrase extraction in pbsmt. In *The twelfth Machine Translation Summit*.

Huang, Fei. 2009. Confidence measure for word alignment. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 932–940.

Ittycheriah, Abraham and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 89–96.

Liu, Yang, Qun Liu, and Shouxun Lin. 2005. Log-linear models for word alignment. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 459–466.

Moore, Robert C. 2005. A discriminative framework for bilingual word alignment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 81–88.

Niehues, Jan. and Stephan. Vogel. 2008. Discriminative word alignment via alignment matrix modeling. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 18–25.

Och, Franz Joseph and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. In *Computational Linguistics*, volume 1:29, pages 19–51.

Taskar, Ben, Simon Lacoste-Julien, and Dan Klein. 2005. A discriminative matching approach to word alignment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 73–80.

Vogel, Stephan, Hermann Ney, and Christoph Tillmann. 1996. HMM based word alignment in statistical machine translation. In *Proceedings of 16th International Conference on Computational Linguistics)*, pages 836–841.

A Large Scale Ranker-Based System for Search Query Spelling Correction

Jianfeng Gao
Microsoft Research, Redmond
jfgao@microsoft.com

Xiaolong Li
Microsoft Corporation
xiaolong.li@microsoft.com

Daniel Micol
Microsoft Corporation
danielmi@microsoft.com

Chris Quirk
Microsoft Research, Redmond
chrisq@microsoft.com

Xu Sun
University of Tokyo
xusun@mist.i.u-tokyo.ac.jp

Abstract

This paper makes three significant extensions to a noisy channel speller designed for standard written text to target the challenging domain of search queries. First, the noisy channel model is subsumed by a more general ranker, which allows a variety of features to be easily incorporated. Second, a distributed infrastructure is proposed for training and applying Web scale *n*-gram language models. Third, a new phrase-based error model is presented. This model places a probability distribution over transformations between multi-word phrases, and is estimated using large amounts of query-correction pairs derived from search logs. Experiments show that each of these extensions leads to significant improvements over the state-of-the-art baseline methods.

1 Introduction

Search queries present a particular challenge for traditional spelling correction methods. New search queries emerge constantly. As a result, many queries contain valid search terms, such as proper nouns and names, which are not well established in the language. Therefore, recent research has focused on the use of Web corpora and search logs, rather than human-compiled lexicons, to infer knowledge about spellings and word usages in search queries (e.g., Whitelaw et al., 2009; Cucerzan and Brill, 2004).

The spelling correction problem is typically formulated under the framework of the noisy channel model. Given an input query $Q = q_1 ... q_I$, we want to find the best spelling correction $C = c_1 ... c_J$ among all candidates:

$$C^* = \underset{C}{\operatorname{argmax}} P(C|Q) \quad (1)$$

Applying Bayes' Rule, we have

$$C^* = \underset{C}{\operatorname{argmax}} P(Q|C)P(C) \quad (2)$$

where the error model $P(Q|C)$ models the transformation probability from C to Q, and the language model (LM) $P(C)$ models the likelihood that C is a correctly spelled query.

This paper extends a noisy channel speller designed for regular text to search queries in three ways: using a ranker (Section 3), using Web scale LMs (Section 4), and using phrase-based error models (Section 5).

First of all, we propose a ranker-based speller that covers the noisy channel model as a special case. Given an input query, the system first generates a short list of candidate corrections using the noisy channel model. Then a feature vector is computed for each query and candidate correction pair. Finally, a ranker maps the feature vector to a real-valued score, indicating the likelihood that this candidate is a desirable correction. We will demonstrate that ranking provides a flexible modeling framework for incorporating a wide variety of features that would be difficult to model under the noisy channel framework.

Second, we explore the use of Web scale LMs for query spelling correction. While traditional LM research focuses on how to make the model "smarter" via how to better estimate the probability of unseen words (Chen and Goodman, 1999); and how to model the grammatical structure of language (e.g., Charniak, 2001), recent studies show that significant improvements can be achieved using "stupid" *n*-gram models trained on very large corpora (e.g., Brants et al., 2007). We adopt the latter strategy in this study. We present a distributed infrastructure to efficiently train and apply Web scale LMs. In addition, we observe that search queries are composed in a language style different from that of regular text. We thus train multiple LMs using different texts associated with Web corpora and search queries.

Third, we propose a phrase-based error model that captures the probability of transforming one multi-term phrase into another multi-term phrase. Compared to traditional error models that account for transformation probabilities between single characters or substrings (e.g., Kernighan et al., 1990; Brill and Moore, 2000), the phrase-based error model is more effective in that it captures inter-term dependencies crucial for correcting real-word errors, prevalent in search queries. We also present a novel method of extracting large amounts of query-correction pairs from search logs. These pairs, implicitly judged by millions of users, are used for training the error models.

Experiments show that each of the extensions leads to significant improvements over its baseline methods that were state-of-the-art until this work, and that the combined method yields a system which outperforms the noisy channel speller by a large margin: a 6.3% increase in accuracy on a human-labeled query set.

2 Related Work

Prior research on spelling correction for regular text can be grouped into two categories: correcting non-word errors and real-word errors. The former focuses on the development of error models based on different edit distance functions (e.g., Kucich, 1992; Kernighan et al., 1990; Brill and Moore, 2000; Toutanova and Moore, 2002). Brill and Moore's substring-based error model, considered to be state-of-the-art among these models, acts as the baseline against which we compare our models. On the other hand, real-word spelling correction tries to detect incorrect usages of a valid word based on its context, such as "peace" and "piece" in the context "a _ of cake". N-gram LMs and naïve Bayes classifiers are commonly used models (e.g., Golding and Roth, 1996; Mangu and Brill, 1997; Church et al., 2007).

While almost all of the spellers mentioned above are based on a pre-defined dictionary (either a lexicon against which the edit distance is computed, or a set of real-word confusion pairs), recent research on query spelling correction focuses on exploiting noisy Web corpora and query logs to infer knowledge about spellings and word usag in queries (Cucerzan and Brill 2004; Ahmad and Kondrak, 2005; Li et al., 2006; Whitelaw et al., 2009). Like those spellers designed for regular text, most of these query spelling systems are also based on the noisy channel framework.

3 A Ranker-Based Speller

The noisy channel model of Equation (2) does not have the flexibility to incorporate a wide variety of features useful for spelling correction, e.g., whether a candidate appears as a Wikipedia document title. We thus generalize the speller to a ranker-based system. Let **f** be a feature vector of a query and candidate correction pair (Q, C). The ranker maps **f** to a real value y that indicates how likely C is a desired correction. For example, a linear ranker maps **f** to y with a weight vector **w** such as $y = \mathbf{w} \cdot \mathbf{f}$, where **w** is optimized for accuracy on human-labeled (Q, C) pairs. Since the logarithms of the LM and error model probabilities can be included as features, the ranker covers the noisy channel model as a special case.

For efficiency, our speller operates in two distinct stages: candidate generation and re-ranking.

In candidate generation, an input query is first tokenized into a sequence of terms. For each term q, we consult a lexicon to identify a list of spelling suggestions c whose edit distance from q is lower than some threshold. Our lexicon contains around 430,000 high frequency query unigram and bigrams collected from 1 year of query logs. These suggestions are stored in a lattice.

We then use a decoder to identify the 20-best candidates from the lattice according to Equation (2), where the LM is a backoff bigram model trained on 1 year of query logs, and the error model is approximated by weighted edit distance:

$$-\log P(Q|C) \propto \text{EditDist}(Q, C) \qquad (3)$$

The decoder uses a standard two-pass algorithm. The first pass uses the Viterbi algorithm to find the best C according to the model of Equations (2) and (3). The second pass uses the A-star algorithm to find the 20-best corrections, using the Viterbi scores computed at each state in the first pass as heuristics.

The core component in the second stage is a ranker, which re-ranks the 20-best candidate corrections using a set of features extracted from (Q, C). If the top C after re-ranking is different from Q, C is proposed as the correction. We use 96 features in this study. In addition to the two features derived from the noisy channel model, the rest of the features can be grouped into the following 5 categories.

1. **Surface-form similarity features**, which check whether C and Q differ in certain patterns,

e.g., whether C is transformed from Q by adding an apostrophe, or by adding a stop word at the beginning or end of Q.

2. **Phonetic-form similarity features**, which check whether the edit distance between the metaphones (Philips, 1990) of a query term and its correction candidate is below some thresholds.

3. **Entity features**, which check whether the original query is likely to be a proper noun based on an in-house named entity recognizer.

4. **Dictionary features**, which check whether a query term or a candidate correction are in one or more human-compiled dictionaries, such as the extracted Wiki, MSDN, and ODP dictionaries.

5. **Frequency features**, which check whether the frequency of a query term or a candidate correction is above certain thresholds in different datasets, such as query logs and Web documents.

4 Web Scale Language Models

An n-gram LM assigns a probability to a word string $w_1^L = (w_1, \ldots, w_L)$ according to

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i | w_1^{i-1}) \approx \prod_{i=1}^{L} P(w_i | w_{i-n+1}^{i-1}) \quad (4)$$

where the approximation is based on a Markov assumption that each word depends only upon the immediately preceding n-1 words. In a speller, the log of n-gram LM probabilities of an original query and its candidate corrections are used as features in the ranker.

While recent research reports the benefits of large LMs trained on Web corpora on a variety of applications (e.g. Zhang et al., 2006; Brants et al., 2007), it is also clear that search queries are composed in a language style different from that of the body or title of a Web document. Thus, in this study we developed a set of large LMs from different text streams of Web documents and query logs. Below, we first describe the n-gram LM collection used in this study, and then present a distributed n-gram LM platform based on which these LMs are built and served for the speller.

4.1 Web Scale Language Models

Table 1 summarizes the data sets and Web scale n-gram LMs used in this study. The collection is built from high quality English Web documents containing trillions of tokens, served by a popular commercial search engine. The collection consists of several data sets built from different Web sources, including the different text fields from the Web documents (i.e., body, title, and anchor texts) and search query logs. The raw texts extracted from these different sources were pre-processed in the following manner: texts are tokenized based on white-space and upper case letters are converted to lower case. Numbers are retained, and no stemming/inflection is performed. The n-gram LMs are word-based backoff models, where the n-gram probabilities are estimated using Maximum Likelihood Estimation with smoothing. Specifically, for a trigram model, the smoothed probability is computed as

Dataset	**Body**	**Anchor**	**Title**	**Query**
Total tokens	1.3T	11.0B	257.2B	28.1B
Unigrams	1.2B	60.3M	150M	251.5M
Bigrams	11.7B	464.1M	1.1B	1.3B
Trigrams	60.0B	1.4B	3.1B	3.1B
4-grams	148.5B	2.3B	5.1B	4.6B
Size on disk#	12.8TB	183GB	395GB	393GB

# N-gram entries as well as other model parameters are stored.

Table 1: Statistics of the Web n-gram LMs collection (count cutoff = 0 for all models). These models will be accessible at Microsoft (2010).

$$P(w_i | w_{i-2} w_{i-1}) = \quad (5)$$

$$\begin{cases} \dfrac{C(w_{i-2} w_{i-1} w_i) - D\big(C(w_{i-2} w_{i-1} w_i)\big)}{C(w_{i-2} w_{i-1})} & \text{if } C(w_{i-2} w_{i-1} w_i) \\ \alpha(w_{i-2} w_{i-1}) P(w_i | w_{i-1}) & \text{otherwise} \end{cases}$$

where $C(\cdot)$ is the count of the n-gram in the training corpus and α is a normalization factor. $D(C)$ is a discount function for smoothing. We use modified absolute discounting (Gao et al., 2001), whose parameters can be efficiently estimated and performance converges to that of more elaborate state-of-the-art techniques like Kneser-Ney smoothing in large data (Nguyen et al. 2007).

4.2 Distributed N-gram LM Platform

The platform is developed on a distributed computing system designed for storing and analyzing massive data sets, running on large clusters consisting of hundreds of commodity servers connected via high-bandwidth network.

We use the SCOPE (Structured Computations Optimized for Parallel Execution) programming model (Chaiken et al., 2008) to train the Web scale n-gram LMs shown in Table 1. The SCOPE scripting language resembles SQL which many programmers are familiar with. It also supports

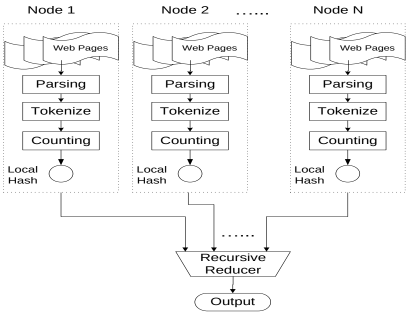


Figure 1. Distributed 5-gram counting.

C# expressions so that users can easily plug-in customized C# classes. SCOPE supports writing a program using a series of simple data transformations so that users can simply write a script to process data in a *serial* manner without wondering how to achieve parallelism while the SCOPE compiler and optimizer are responsible for translating the script into an efficient, parallel execution plan. We illustrate the usage of SCOPE for building LMs using the following example of counting 5-grams from the body text of English Web pages. The flowchart is shown in Figure 1.

The program is written in SCOPE as a step-by- step of computation, where a command takes the output of the previous command as its input.

```
ParsedDoc=SELECT docId, TokenizedDoc
   FROM @"/shares/…/EN_Body.txt"
   USING DefaultTextExtractor;

NGram=PROCESS ParsedDoc
   PRODUCE NGram, NGcount
   USING NGramCountProcessor(-stream
   TokenizedDoc -order 5 –bufferSize
   20000000);

NGramCount=REDUCE NGram
   ON NGram
   PRODUCE NGram, NGcount
   USING NGramCountReducer;

OUTPUT TO @"Body-5-gram-count.txt";
```

The first SCOPE command is a SELECT statement that extracts parsed Wed body text. The second command uses a build-in Processor (NGramCountProcessor) to map the parsed documents into separate *n*-grams together with their counts. It generates a local hash at each node (i.e., a core in a multi-core server) to store the (*n*-gram, count) pairs. The third command (REDUCE) aggregates counts from different nodes according to the key (*n*-gram string). The final command (OUTPUT) writes out the resulting to a data file.

C:	"disney theme park"	*correct query*
S:	["disney", "theme park"]	*segmentation*
T:	["disnee", "theme part"]	*translation*
M:	(1 → 2, 2→ 1)	*permutation*
Q:	"theme part disnee"	*misspelled query*

Figure 2: Example demonstrating the generative procedure behind the phrase-based error model.

The smoothing method can be implemented similarly by the customized smoothing Processor/Reducer. They can be imported from the existing C# codes (e.g., developed for building LMs in a single machine) with minor changes.

It is straightforward to apply the built LMs for the ranker in the speller. The *n*-gram platform provides a DLL for *n*-gram batch lookup. In the server, an *n*-gram LM is stored in the form of multiple lists of key-value pairs, where the key is the hash of an *n*-gram string and the value is either the *n*-gram probability or backoff parameter.

5 Phrase-Based Error Models

The goal of an error model is to transform a correctly spelled query *C* into a misspelled query *Q*. Rather than replacing single words in isolation, the phrase-based error model replaces sequences of words with sequences of words, thus incorporating contextual information. The training procedure closely follows Sun et al. (2010). For instance, we might learn that "*theme part*" can be replaced by "*theme park*" with relatively high probability, even though "*part*" is not a misspelled word. We use this generative story: first the correctly spelled query *C* is broken into *K* non-empty word sequences $\mathbf{c}_1, \ldots, \mathbf{c}_k$, then each is replaced with a new non-empty word sequence $\mathbf{q}_1, \ldots, \mathbf{q}_k$, finally these phrases are permuted and concatenated to form the misspelled *Q*. Here, **c** and **q** denote consecutive sequences of words.

To formalize this generative process, let *S* denote the segmentation of *C* into *K* phrases $\mathbf{c}_{1\ldots}\mathbf{c}_K$, and let *T* denote the *K* replacement phrases $\mathbf{q}_1\ldots\mathbf{q}_K$ – we refer to these ($\mathbf{c}_i$, $\mathbf{q}_i$) pairs as *bi-phrases*. Finally, let *M* denote a permutation of *K* elements representing the final reordering step. Figure 2 demonstrates the generative procedure.

Next let us place a probability distribution over rewrite pairs. Let *B*(*C*, *Q*) denote the set of *S*, *T*, *M* triples that transform *C* into *Q*. Assuming a uniform probability over segmentations, the phrase-based probability can be defined as:

$$P(Q|C) \propto \sum_{(S,T,M)\in B(C,Q)} P(T|C,S) \cdot P(M|C,S,T) \quad (6)$$

As is common practice in SMT, we use the maximum approximation to the sum:

$$P(Q|C) \approx \max_{(S,T,M)\in B(C,Q)} P(T|C,S) \cdot P(M|C,S,T) \quad (7)$$

5.1 Forced Alignments

Although we have defined a generative model for transforming queries, our goal is not to propose new queries, but rather to provide scores over existing *Q* and *C* pairs that will act as features for the ranker. Furthermore, the word-level alignments between *Q* and *C* can most often be identified with little ambiguity. Thus we restrict our attention to those phrase transformations consistent with a good word-level alignment.

Let *J* be the length of *Q*, *L* be the length of *C*, and $A = a_1 \ldots a_J$ be a hidden variable representing the word alignment between them. Each a_i takes on a value ranging from 1 to *L* indicating its corresponding word position in *C*, or 0 if the *i*th word in *Q* is unaligned. The cost of assigning *k* to a_i is equal to the Levenshtein edit distance (Levenshtein, 1966) between the i^{th} word in *Q* and the k^{th} word in *C*, and the cost of assigning 0 to a_i is equal to the length of the i^{th} word in *Q*. The least cost alignment *A** between *Q* and *C* is computed efficiently using the A-star algorithm.

When scoring a given candidate pair, we further restrict our attention to those *S*, *T*, *M* triples that are consistent with the word alignment, which we denote as $B(\mathrm{C}, Q, A^*)$. Here, consistency requires that if two words are aligned in A^*, then they must appear in the same bi-phrase ($\mathbf{c}_i$, $\mathbf{q}_i$). Once the word alignment is fixed, the final permutation is uniquely determined, so we can safely discard that factor. Thus we have:

$$P(Q|C) \approx \max_{\substack{(S,T,M)\in \\ \mathcal{B}(C,Q,A^*)}} P(T|C,S) \quad (8)$$

For the sole remaining factor *P*(*T*|*C*, *S*), we make the assumption that a segmented query $T = \mathbf{q}_1 \ldots \mathbf{q}_K$ is generated from left to right by transforming each phrase $\mathbf{c}_1 \ldots \mathbf{c}_K$ independently:

$$P(T|C,S) = \prod_{k=1}^{K} P(\mathbf{q}_k|\mathbf{c}_k), \quad (9)$$

where $P(\mathbf{q}_k|\mathbf{c}_k)$ is a phrase transformation probability, the estimation of which will be described in Section 5.2.

```
Google:
   http://www.google.com/search?
   hl=en&source=hp&
   q=harrypotter+sheme+part&aq=f&oq=&aqi=

   http://www.google.com/search?
   hl=en&ei=rnNAS8-oKsWe_AaB2eHlCA&
   sa=X&oi=spell&resnum=0&ct=
   result&cd=1&ved=0CA4QBSgA&
   q=harry+potter+theme+park&spell=1
Yahoo:
   http://search.yahoo.com/search;
   _ylt=A0geu6ywckBL_XIBSDtXNyoA?
   p=harrypotter+sheme+part&
   fr2=sb-top&fr=yfp-t-701&sao=1

   http://search.yahoo.com/search?
   ei=UTF-8&fr=yfp-t-701&
   p=harry+potter+theme+park
   &SpellState=n-2672070758_q-tsI55N6srhZa.
   qORA0MuawAAAA%40%40&fr2=sp-top
Bing:
   http://www.bing.com/search?
   q=harrypotter+sheme+part&form=QBRE&qs=n

   http://www.bing.com/search?
   q=harry+potter+theme+park&FORM=SSRE
```

Figure 3. A sample of query reformulation sessions from 3 popular search engines. These sessions show that a user first issues the query "harrypotter sheme part", and then clicks on the resulting spell suggestion "harry potter theme park".

To find the maximum probability assignment efficiently, we use a dynamic programming approach, similar to the monotone decoding algorithm described in Och (2002).

5.2 Training the Error Model

Given a set of (*Q*, *C*) pairs as training data, we follow a method commonly used in SMT (Och and Ney, 2004) to extract bi- phrases and estimate their replacement probabilities. A detailed description is discussed in Sun et al. (2010).

We now describe how (*Q*, *C*) pairs are generated automatically from massive *query reformulation sessions* of a commercial Web browser.

A query reformulation session contains a list of URLs that record user behaviors that relate to the query reformulation functions, provided by a Web search engine. For example, most commercial search engines offer the "did you mean" function, suggesting a possible alternate interpretation or spelling of a user-issued query. Figure 3 shows a sample of the query reformulation sessions that record the "did you mean" sessions from three of the most popular search engines. These sessions encode the same user behavior: A user first queries for "harrypotter sheme part", and then clicks on the resulting spelling suggestion "harry potter theme park". We can "reverse-engineer" the parameters from the URLs of these sessions, and deduce how each search engine encodes both a query and the fact that a user arrived at a URL by clicking on the spelling suggestion of the query – an strong indication that the spelling suggestion is desired. In this study, from 1 year of sessions, we extracted ~120 million pairs. We found the data set very clean because these spelling corrections are actually clicked, and thus judged implicitly, by many users.

In addition to the "did you mean" functionality, recently some search engines have introduced two new spelling suggestion functions. One is the "auto-correction" function, where the search engine is confident enough to automatically apply the spelling correction to the query and execute it to produce search results. The other is the "split pane" result page, where one half portion of the search results are produced using the original query, while the other half, usually visually separate portion of results, are produced using the auto-corrected query.

In neither of these functions does the user ever receive an opportunity to approve or disapprove of the correction. Since our extraction approach focuses on user-approved spelling suggestions, we ignore the query reformulation sessions recording either of the two functions. Although by doing so we could miss some basic, obvious spelling corrections, our experiments show that the negative impact on error model training is negligible. One possible reason is that our baseline system, which does not use any error model learned from the session data, is already able to correct these basic, obvious spelling mistakes. Thus, including these data for training is unlikely to bring any further improvement.

We found that the error models trained using the data directly extracted from the query reformulation sessions suffer from the problem of underestimating the self-transformation probability of a query $P(Q_2{=}Q_1|Q_1)$, because we only included in the training data the pairs where the query is different from the correction. To deal with this problem, we augmented the training data by including correctly spelled queries, i.e., the pairs (Q_1, Q_2) where $Q_1 = Q_2$. First, we extracted a set of queries from the sessions where no spell suggestion is presented or clicked on. Second, we removed from the set those queries that were recognized as being auto-corrected by a search engine. We do so by running a sanity check of the queries against our baseline noisy channel speller, which will be described in Section 6. If the system consider a query misspelled, we assumed it an obvious misspelling, and removed it. The remaining queries were assumed to be correctly spelled and were added to the training data.

6 Experiments

We perform the evaluation using a manually annotated data set containing 24,172 queries sampled from one year's query logs from a commercial search engine. The spelling of each query is manually corrected by four independent annotators. The average length of queries in the data sets is 2.7 words. We divided the data set into non-overlapped training and test data sets. The training data contain 8,515 (Q, C) pairs, among which 1,743 queries are misspelled (i.e. $Q \neq C$). The test data contain 15,657 (Q, C) pairs, among which 2,960 queries are misspelled.

The speller systems we developed in this study are evaluated using the following metrics.

- **Accuracy:** The number of correct outputs generated by the system divided by the total number of queries in the test set.
- **Precision:** The number of correct spelling corrections for misspelled queries generated by the system divided by the total number of corrections generated by the system.
- **Recall:** The number of correct spelling corrections for misspelled queries generated by the system divided by the total number of misspelled queries in the test set.

We also perform a significance test, a t-test with a significance level of 0.05.

In our experiments, all the speller systems are ranker-based. Unless otherwise stated, the ranker is a two-layer neural net with 5 hidden nodes. The free parameters of the neural net are trained to optimize accuracy on the training data using the back propagation algorithm (Burges et al., 2005) .

6.1 System Results

Table 1 summarizes the main results of different spelling systems. Row 1 is the baseline speller where the noisy channel model of Equations (2)

#	System	Accuracy	Precision	Recall
1	Noisy channel	85.3	72.1	35.9
2	Linear ranker	88.0	74.0	42.8
3	Nonlinear ranker	89.0	74.1	49.6
4	3 + PBEM	90.7	78.7	58.2
5	3 + WLMs	90.4	75.1	58.7
6	3 + PBEM + WLMs	91.6	79.1	63.9

Table 1. Summary of spelling correction results.

and (3) is used. The error model is based on the weighted edit distance function and the LM is a backoff bigram model trained on 1 year of query logs, with count cutoff 30. Row 2 is the speller using a linear ranker to incorporate all ranking features described in Section 3. The weights of the linear ranker are optimized using the Averaged Perceptron algorithm (Freund and Schapire, 1999). Row 3 is the speller where a nonlinear ranker (i.e., 2-layer neural net) is trained atop the features. Rows 4, 5 and 6 are systems that incorporate the additional features derived from the phrase-based error model (PBEM) described in Section 5 and the four Web scale LMs (WLMs) listed in Table 1.

The results show that (1) the ranker is a very flexible modeling framework where a variety of fine-grained features can be easily incorporated, and a ranker-based speller outperforms significantly (p < 0.01) the traditional system based on the noisy channel model (Row 2 vs. Row 1); (2) the speller accuracy can be further improved by using more sophisticated rankers and learning algorithms (Row 3 vs. Row 2); (3) both WLMs and PBEM bring significant improvements (Rows 4 and 5 vs. Row 3); and (4) interestingly, the gains from WLMs and PBEM are additive and the combined leads to a significantly better speller (Row 6 vs. Rows 4 and 5) than that of using either of them individually.

In what follows, we investigate in detail how the WLMs and PBEM trained on massive Web content and search logs improve the accuracy of the speller system. We will compare our models with the state-of-the-art models proposed previously. From now on, the system listed in Row 3 of Table 1 will be used as baseline.

6.2 Language Models

The quality of n-gram LMs depends on the order of the model, the size of the training data, and how well the training data match the test data. Figure 4 illustrates the perplexity results of the four LMs trained on different data sources tested on a random sample of 733,147 queries. The results show that (1) higher order LMs produce lower perplexities, especially when moving beyond unigram models; (2) as expected, the query LMs are most predictive for the test queries, though they are from independent query log snapshots; (3) although the body LMs are trained on much larger amounts of data than the title and anchor LMs, the former lead to much higher perplexity values, indicating that both title and anchor texts are quantitatively much more similar to queries than body texts.

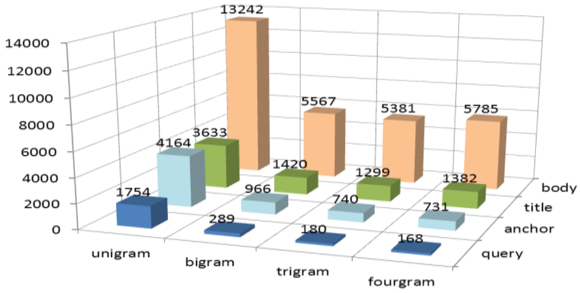


Figure 4. Perplexity results on test queries, using n-gram LMs with different orders, derived from different data sources.

Table 2 summarizes the spelling results using different LMs. For comparison, we also built a 4-gram LM using the Google 1T web 5-gram corpus (Brants and Franz, 2006). This model is referred to as the G1T model, and is trained using the "stupid backoff" smoothing method (Brants et al., 2007). Due to the high count cutoff applied by the Google corpus (i.e., n-grams must appear at least 40 times to be included in the corpus), we found the G1T model results to a higher OOV rate (i.e., 6.5%) on our test data than that of the 4 Web scale LMs (i.e., less than 1%).

The results in Table 2 are more or less consistent with the perplexity results: the query LM is the best performer; there is no significant difference among the body, title and anchor LMs though the body LM is trained on a much larger amount of data; and all the 4 Web scale LMs outperform the G1T model substantially due to the significantly lower OOV rates.

6.3 Error Models

This section compares the phrase-based error model (PBEM) described in Section 5, with one of the state-of-the-art error models, proposed by Brill and Moore (2000), henceforth referred to as

#	System	Accuracy	Precision	Recall
1	Baseline	89.0	74.1	49.6
2	1+ query 4-gram	90.1	75.6	56.3
3	1 + body 4-gram	89.9	75.7	54.4
4	1 + title 4-gram	89.8	75.4	54.7
5	1 + anchor 4-gram	89.9	75.1	55.6
6	1 + G1T 4-gram	89.4	75.1	51.5

Table 2. Spelling correction results using different LMs trained on different data sources.

#	System	Accuracy	Precision	Recall
1	Baseline w/o EM	88.6	72.0	47.0
2	Baseline	89.0	74.1	49.6
3	1 + B&M, L=1	89.0	73.3	50.1
4	1 + B&M, L=3	89.2	73.7	51.0
5	1 + PBEM, L=1	90.1	76.7	55.6
6	1 + PBEM, L=3	90.7	78.5	58.1

Table 3. Spelling correction results using different error models.

the B&M model. B&M is a substring error model. It estimates $P(q|c)$ as

$$P(q|c) \approx \max_{\substack{R,T \\ s.t.|T|=|R|}} \prod_{i=1}^{|R|} P(T_i|R_i), \tag{10}$$

where R is a partitioning of correction term c into adjacent substrings, and T is a partitioning of query term q, such that $|T|=|R|$. The partitions are thus in one-to-one alignment. To train the B&M model, we extracted 1 billion term-correction pairs (q, c) from the set of 120 million query-correction pairs (Q, C), derived from the search logs as described in Section 5.2.

Table 3 summarizes the comparison results. Rows 1 and 2 are our ranker-based baseline systems with and without the error model (EM) feature. The error model is based on weighted edit distance of Eq. (3), where the weights are learned on some manually annotated word-correction pairs (which is not used in this study). Rows 3 and 4 are the B&M models using different maximum substring lengths, specified by L. L=1 reduces B&M to the weighted edit distance model in Row 2. Rows 5 and 6 are PBEMs with different maximum phrase lengths. L=1 reduces PBEM to a word-based error model. The results show the benefits of capturing context information in error models. In particular, the significant improvements resulting from PBEM demonstrate that the dependencies between words are far more effective than that between characters (within a word) for spelling correction. This is largely due to the fact that there are many real-word spelling errors in search queries. We also notice that PBEM is a more powerful model than B&M in that it can benefit more from increasingly larger training data. As shown in Tables 4 and 5, whilst the performance of B&M saturates quickly with the increase of training data, the performance of PBEM does not appear to have peaked – further improvements are likely given a larger data set.

#	# of word pairs	Accuracy	Precision	Recall
1	Baseline w/o EM	88.55	71.95	46.97
2	1M	89.15	73.71	50.74
3	10M	89.22	74.11	50.92
4	100M	89.20	73.60	51.06
5	1B	89.21	73.72	50.99

Table 4. The performance of B&M error model (L=3) as a function of the size of training data (# of word pairs).

#	# of (Q, C) pairs	Accuracy	Precision	Recall
1	Baseline w/o EM	88.55	71.95	46.97
2	5M	89.59	77.01	52.34
3	15M	90.23	77.87	56.67
4	45M	90.45	78.56	57.02
5	120M	90.70	78.49	58.12

Table 5. The performance of PBEM (L=3) as a function of the size of training data (# of (Q, C) pairs).

7 Conclusions and Future Work

This paper explores the use of massive Web corpora and search logs for improving a ranker-based search query speller. We show significant improvements over a noisy channel speller using fine-grained features, Web scale LMs, and a phrase-based error model that captures internword dependencies. There are several techniques we are exploring to make further improvements. First, since a query speller is developed for improving the Web search results, it is natural to use features from search results in ranking, as studied in Chen et al. (2007). The challenge is efficiency. Second, in addition to query reformulation sessions, we are exploring other search logs from which we might extract more (Q, C) pairs for error model training. One promising data source is clickthrough data (e.g., Agichtein et al, 2006; Gao et al., 2009). For instance, we might try to learn a transformation from the title or anchor text of a document to the query that led to a click on that document. Finally, the phrase-based error model is inspired by phrase-based SMT systems. We are introducing more SMT techniques such as alignment and translation rule exaction. In a broad sense, spelling correction can be viewed as a monolingual MT problem where we translate bad English queries into good ones.

Acknowledgments

The authors would like to thank Andreas Bode, Mei Li, Chenyu Yan and Kuansan Wang for the very helpful discussions and collaboration. The work was done when Xu Sun was visiting Microsoft Research Redmond.

References

Agichtein, E., Brill, E. and Dumais, S. 2006. Improving web search ranking by incorporating user behavior information. In *SIGIR*, pp. 19-26.

Ahmad, F., and Kondrak, G. 2005. Learning a spelling error model from search query logs. In *HLT-EMNLP*, pp. 955-962.

Brants, T., and Franz, A. 2006. Web 1T 5-gram corpus version 1.1. Technical report, Google Research.

Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J. 2007. Large language models in machine translation. In *EMNLP-CoNLL*, pp. 858 - 867.

Brill, E., and Moore, R. C. 2000. An improved error model for noisy channel spelling correction. In *ACL*, pp. 286-293.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, and Hullender, G. 2005. Learning to rank using gradient descent. In *ICML*, pp. 89-96.

Chaiken, R., Jenkins, B., Larson, P., Ramsey, B., Shakib, D., Weaver, S., and Zhou, J. 2008. SCOPE: easy and efficient parallel processing f massive data sets. In *Proceedings of the VLDB Endowment*, pp. 1265-1276.

Charniak, E. 2001. Immediate-head parsing for language models. In *ACL/EACL*, pp. 124-131.

Chen, S. F., and Goodman, J. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(10):359-394.

Chen, Q., Li, M., and Zhou, M. 2007. Improving query spelling correction using web search results. In *EMNLP-CoNLL*, pp. 181-189.

Church, K., Hard, T., and Gao, J. 2007. Compressing trigram language models with Golomb coding. In *EMNLP-CoNLL*, pp. 199-207.

Cucerzan, S., and Brill, E. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *EMNLP*, pp. 293-300.

Freund, Y. and Schapire, R. E. 1999. Large margin classification using the perceptron algorithm. In *Machine Learning*, 37(3): 277-296.

Gao, J., Goodman, J., and Miao, J. 2001. The use of clustering techniques for language modeling - application to Asian languages. *Computational Linguistics and Chinese Language Processing*, 6(1):27–60, 2001.

Gao, J., Yuan, W., Li, X., Deng, K., and Nie, J-Y. 2009. Smoothing clickthrough data for web search ranking. In *SIGIR*, pp. 355-362.

Golding, A. R., and Roth, D. 1996. Applying winnow to context-sensitive spelling correction. In *ICML*, pp. 182-190.

Joachims, T. 2002. Optimizing search engines using clickthrough data. In *SIGKDD*, pp. 133-142.

Kernighan, M. D., Church, K. W., and Gale, W. A. 1990. A spelling correction program based on a noisy channel model. In *COLING*, pp. 205-210.

Koehn, P., Och, F., and Marcu, D. 2003. Statistical phrase-based translation. In *HLT/NAACL*, pp. 127-133.

Kucich, K. 1992. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4):377-439.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707-710.

Li, M., Zhu, M., Zhang, Y., and Zhou, M. 2006. Exploring distributional similarity based models for query spelling correction. In *ACL*, pp. 1025-1032.

Mangu, L., and Brill, E. 1997. Automatic rule acquisition for spelling correction. In *ICML*, pp. 187-194.

Microsoft Microsoft web n-gram services. 2010. http://research.microsoft.com/web-ngram

Nguyen, P., Gao, J., and Mahajan, M. 2007. MSRLM: a scalable language modeling toolkit. Technical report TR-2007-144, Microsoft Research.

Och, F. 2002. *Statistical machine translation: from single-word models to alignment templates*. PhD thesis, RWTH Aachen.

Och, F., and Ney, H. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4): 417-449.

Philips, L. 1990. Hanging on the metaphone. *Computer Language Magazine*, 7(12):38-44.

Sun, X., Gao, J., Micol, D., and Quirk, C. 2010. Learning phrase-based spelling error models from clickthrough data. In *ACL*.

Toutanova, K., and Moore, R. 2002. Pronunciation modeling for improved spelling correction. In *ACL*, pp. 144-151.

Whitelaw, C., Hutchinson, B., Chung, G. Y., and Ellis, G. 2009. Using the web for language independent spellchecking and autocorrection. In *EMNLP*, pp. 890-899.

Zhang, Y., Hildebrand, Al. S., and Vogel, S. 2006. Distributed language modeling for n-best list reranking. In *EMNLP*, pp. 216-233.

RTG based surface realisation for TAG

Claire Gardent
CNRS/LORIA
claire.gardent@loria.fr

Laura Perez-Beltrachini
Université Henri Poincaré/LORIA
laura.perez@loria.fr

Abstract

Surface realisation with grammars integrating flat semantics is known to be NP complete. In this paper, we present a new algorithm for surface realisation based on Feature Based Tree Adjoining Grammar (FTAG) which draws on the observation that an FTAG can be translated into a Regular Tree Grammar describing its derivation trees. We carry out an extensive testing of several variants of this algorithm using an automatically produced testsuite and compare the results obtained with those obtained using GenI, another FTAG based surface realiser.

1 Introduction

As shown in (Brew, 1992; Koller and Striegnitz, 2002), Surface Realisation is NP-complete. Various optimisation techniques have therefore been proposed to help improve practical runtimes. For instance, (Kay, 1996) proposes to reduce the number of constituents built during realisation by only considering for combination constituents with non overlapping semantics and compatible indices. (Kay, 1996; Carroll and Oepen, 2005; Gardent and Kow, 2007) propose various techniques to restrict the combinatorics induced by intersective modifiers all applying to the same structure. And (Koller and Striegnitz, 2002; Gardent and Kow, 2007) describe two alternative techniques for reducing the initial search space.

In this paper, we focus on the optimisation mechanisms of two TAG based surface realisers namely, GENI (Gardent and Kow, 2007) and the algorithm we present in this paper namely, RTGEN (Perez-Beltrachini, 2009). GENI's optimisation includes both a filtering process whose aim is to reduce the initial search space and a two step, "substitution before adjunction", tree combination phase whose effect is to delay modifier adjunction thereby reducing the number of intermediate structures being built. In RTGEN on the other hand, the initial FTAG is converted to a Regular Tree Grammar (RTG) describing its derivation trees and an Earley algorithm, including sharing and packing, is used to optimise tree combination.

We compare GENI with several variants of the proposed RTGEN algorithm using an automatically produced testsuite of 2 679 input formulae and relate the RTGEN approach to existing work on surface realisation optimisation.

The paper is structured as follows. We first present the grammar used by both GENI and RTGEN, namely SEMXTAG (Section 2). We then describe the two surface realisation algorithms (Section 3). In Section 4, we describe the empirical evaluation carried out and present the results. Finally, Section 5 situates RTGEN with respect to related work on surface realisation optimisation.

2 SemXTag

The grammar (SEMXTAG) used by GENI and RTGEN is a Feature-Based Lexicalised Tree Adjoining Grammar (FTAG) augmented with a unification-based semantics as described in (Gardent and Kallmeyer, 2003). We briefly introduce each of these components and describe the grammar coverage. We then show how this FTAG can be converted to an RTG describing its derivation trees.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 367–375,
Beijing, August 2010

2.1 FTAG.

A Feature-based TAG (Vijay-Shanker and Joshi, 1988) consists of a set of (auxiliary or initial) elementary trees and of two tree-composition operations: substitution and adjunction. Initial trees are trees whose leaves are labeled with substitution nodes (marked with a downarrow) or terminal categories. Auxiliary trees are distinguished by a foot node (marked with a star) whose category must be the same as that of the root node. Substitution inserts a tree onto a substitution node of some other tree while adjunction inserts an auxiliary tree into a tree. In an FTAG, the tree nodes are furthermore decorated with two feature structures (called **top** and **bottom**) which are unified during derivation as follows. On substitution, the top of the substitution node is unified with the top of the root node of the tree being substituted in. On adjunction, the top of the root of the auxiliary tree is unified with the top of the node where adjunction takes place; and the bottom features of the foot node are unified with the bottom features of this node. At the end of a derivation, the top and bottom of all nodes in the derived tree are unified. Finally, each sentence derivation in an FTAG is associated with both a **derived tree** representing the phrase structure of the sentence and a **derivation tree** recording how the corresponding elementary trees were combined to form the derived tree. Nodes in a derivation tree are labelled with the name of a TAG elementary tree. Edges are labelled with a description of the operation used to combine the TAG trees whose names label the edge vertices.

2.2 FTAG with semantics.

To associate semantic representations with natural language expressions, the FTAG is modified as proposed in (Gardent and Kallmeyer, 2003).

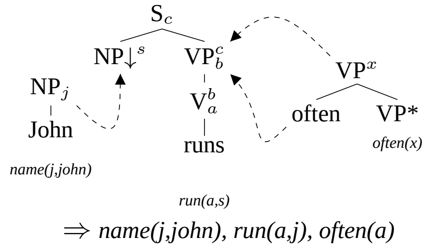


Figure 1: Flat Semantics for “John often runs”

Each elementary tree is associated with a flat semantic representation. For instance, in Figure 1,[1] the trees for *John*, *runs* and *often* are associated with the semantics *name(j,john)*, *run(a,s)* and *often(x)* respectively. Importantly, the arguments of a semantic functor are represented by unification variables which occur both in the semantic representation of this functor and on some nodes of the associated syntactic tree. For instance in Figure 1, the semantic index s occurring in the semantic representation of *runs* also occurs on the subject substitution node of the associated elementary tree. The value of semantic arguments is determined by the unifications resulting from adjunction and substitution. For instance, the semantic index s in the tree for *runs* is unified during substitution with the semantic index labelling the root node of the tree for *John*. As a result, the semantics of *John often runs* is {`name(j,john),run(a,j),often(a)`}.

2.3 SemXTAG.

SemXTag is an FTAG for English augmented with a unification based compositional semantics of the type described above. Its syntactic coverage approaches that of XTAG, the FTAG developed for English by the XTAG group (The XTAG Research Group, 2001). Like this grammar, it contains around 1300 elementary trees and covers auxiliaries, copula, raising and small clause constructions, topicalization, relative clauses, infinitives, gerunds, passives, adjuncts, ditransitives and datives, ergatives, it-clefts, wh-clefts, PRO constructions, noun-noun modification, extraposition, sentential adjuncts, imperatives and resultatives.

2.4 Converting SemXTAG to RTG

As shown in (Schmitz and Le Roux, 2008), an FTAG can be converted to a Regular Tree Grammar describing its derivation tree. In this section, we briefly sketch this conversion process. For a more precise description of this FTAG to RTG conversion, the reader is referred to (Schmitz and Le Roux, 2008).

[1] C^x/C_x abbreviate a node with category C and a top/bottom feature structure including the feature-value pair { **index :** x}.

In the FTAG-to-RTG conversion, each SemXTag elementary tree is converted to a rule that models its contribution to a TAG derivation tree. A TAG derivation involves the selection of an initial tree, which has some nodes requiring substitution and some permitting adjunction. Let us think of the potential adjunction sites as requiring, rather than permitting, adjunction, but such that the requirement can be satisfied by ‘null’ adjunction. Inserting another tree into this initial tree satisfies one of the substitution or adjunction requirements, but introduces some new requirements into the resulting tree, in the form of its own substitution nodes and adjunction sites.

Thus, intuitively, the RTG representation of a SemXTag elementary tree is a rule that rewrites the satisfied requirement as a local tree whose root is a unique identifier of the tree and whose leaves are the introduced requirements. A requirement of a substitution or adjunction of a tree of root category X is written as X_S or X_A, respectively. Here, for example, is the translation to RTG of the FTAG tree (minus semantics) for *run* in Figure 1, using the word anchoring the tree as its identifier (the upperscripts abbreviates features structures: b/t refers to the bottom/top feature structure and the upper case letters to the semantic index value, $[idx : X]$ is abbreviated to X):

$$S_S^{[t:T]} \rightarrow runs(S_A^{[t:T,b:C]}\ NP_S^{[t:S]}\ VP_A^{[t:C,b:B]}\ V_A^{[t:B,b:A]})$$

The semantics of the SemXTAG tree are carried over as-is to the corresponding RTG rule. Further, the feature structures labelling the nodes of the SemXTAG tree are converted into the RTG rules so as to correctly interact with substitution and adjunction (see (Schmitz and Le Roux, 2008) for more details on this part of the conversion process).

To account for the optionality of adjunction, there are additional rules allowing any adjunction requirement to be rewritten as the symbol ϵ, a terminal symbol of the RTG.

The terminal symbols of the RTG are thus the tree identifiers and the symbol ϵ, and its non-terminals are X_S and X_A for each terminal or non-terminal X of SemXTAG.

3 TAG-based surface realisation

We now present RTGen and describe GenI, and compare the optimisations they propose to deal with the task complexity.

GenI and RTGen are similar on several points. They use the same grammar, namely SemXTag (cf. Section 2). Further, they both pipeline three main steps. First, **lexical selection** selects from the grammar those elementary trees whose semantics subsumes part of the input semantics. Second, the **tree combining** phase systematically tries to combine trees using substitution and adjunction. Third, the **retrieval phase** extracts the yields of the complete derived trees, thereby producing the generated sentence(s).

GenI and RTGen differ however with respect to the trees they are working with (derived trees in GenI *vs* derivation trees in RTGen). They also differ in how tree combination is handled. We now describe these differences in more detail and explain how each approach address the complexity issue.

3.1 GenI

The tree combining phase in GenI falls into two main steps namely, filtering and tree combining.

Filtering. The so-called polarity filtering step aims to reduce the initial search space. It eliminates from the initial search space all those sets of TAG elementary trees which cover the input semantics but cannot possibly lead to a valid derived tree. In specific, this filtering removes all tree sets covering the input semantics such that either the category of a substitution node cannot be canceled out by that of the root node of a different tree; or a root node fails to have a matching substitution site. Importantly, this filtering relies solely on categorial information – feature information is not used. Furthermore, auxiliary trees have no impact on filtering since they provide and require the same category thereby being “polarity neutral elements”.

Tree combining. The tree combining algorithm used after filtering has taken place, is a bottom-up tabular algorithm (Kay, 1996) optimised for TAGs. This step, unlike the first, uses all the features present in the grammar. To handle intersective modifiers, the delayed modifiers insertion strategy from (Carroll et al., 1999) is adapted to TAG as follows. First, all possible derived trees are obtained using only substitution. Next, adjunction is applied. Although the number of intermediate structures generated is still 2^n for n modifiers, this strategy has the effect of blocking these 2^n structures from multiplying out with other structures in the chart.

3.2 RTGen

RTGen synthesises different techniques that have been observed in the past to improve surface realisation runtimes. We first describe these techniques i.e., the main features of RTGEN. We then present three alternative ways of implementing RTGEN which will be compared in the evaluation.

3.2.1 RTGen's main features

A main feature of RTGEN is that it focuses on building derivation rather than derived trees. More specifically, the first two steps of the surface realisation process (lexical selection, tree combining) manipulate RTG rules describing the contribution of the SEMXTAG elementary trees to the derivation tree rather than the elementary tree themselves. The derived trees needed to produce actual sentences are only produced in the last phase i.e., the retrieval phase.

This strategy is inspired from a similar approach described in (Koller and Striegnitz, 2002) which was shown to be competitive with state of the art realisers on a small sample of example input chosen for their inherent complexity. (Koller and Striegnitz, 2002)'s approach combines trees using a constraint based dependency parser rather than an Earley algorithm so that it is difficult to assess how much of the efficiency is due to the parser and how much to the grammar conversion. Intuitively however, the motivation underlying the construction of a derivation rather than a derived tree is that efficiency might be increased because the context free derivation trees (i) are simpler than the mildly context sensitive trees generated by an FTAG and (ii) permit drawing on efficient parsing and surface realisation algorithms designed for such grammars.

Second, RTGEN makes use of the now standard semantic criteria proposed in (Kay, 1996; Carroll et al., 1999) to reduce the number of combinations tried out by the realiser. On the one hand, two constituents are combined by the algorithm's inference rules only if they cover disjoint parts of the input semantics. On the other hand, the semantic indices present in both the input formula and the lexically retrieved RTG trees are used to prevent the generation of intermediate structures that are not compatible with the input semantics. For instance, given the input formula for "John likes Mary", semantic indices will block the generation of "likes John" because this constituent requires that the constituent for "John" fills the patient slot of "likes" whereas the input semantics requires that it fills the agent slot. In addition, chart items in RTGEN are indexed by semantic indices to efficiently select chart items for combination.

Third, RTGEN implements a standard Earley algorithm complete with sharing and packing. Sharing allows for intermediate structures that are common to several derivations to be represented only once – in addition to not being recomputed each time. Packing means that partial derivation trees with identical semantic coverage and similar combinatorics (same number and type of substitution and adjunction requirements) are grouped together and that only one representative of such groups is stored in the chart. In this way, intermediate structures covering the same set of intersective modifiers in a different order are only represented once and the negative impact of intersective modifiers is lessened (cf. (Brew, 1992)). . As (Carroll and Oepen, 2005) have shown, packing and sharing are important factors in improving efficiency. In particular, they show that an algorithm with packing and sharing clearly outtperforms the same algorithm without packing and sharing giving an up to 50 times speed-up for inputs with large numbers of realizations.

3.2.2 Three ways to implement RTGen

Depending on how much linguistic information (i.e. feature constraints from the feature structures) is preserved in the RTG rules, several RTGEN configurations can be tried out which each reflect a different division of labour between constraint solving and structure building. To experiment with these several configurations, we exploit the fact that the FTAG-to-RTG conversion procedure developed by Sylvain Schmitz permits specifying which features should be preserved by the conversion.

RTGen-all. In this configuration, all the feature structure information present in the SEMXTAG elementary trees is carried over to the RTG rules. As a result, tree combining and constraint solving proceed simultaneously and the generated parse forest contains the derivation trees of all the output sentences.

RTGen-level0. In the RTGen-level0 configuration, only the syntactic category and the semantic features are preserved by the conversion. As a result, the grammar information used by the (derivation) tree building phase is comparable to that used by GENI filtering step. In both cases, the aim is to detect those sets of elementary trees which cover the input semantics and such that all syntactic requirements are satisfied while no syntactic resource is left out. A further step is additionally needed to produce only those trees which can be built from these tree sets when applying the constraints imposed by other features. In GENI, this additional step is carried out by the tree combining phase, in RTGEN, it is realised by the extraction phase i.e., the phase that constructs the derived trees from the derivation trees produced by the tree combining phase.

RTGen-selective. Contrary to parsing, surface realisation only accesses the morphological lexicon last i.e., after sentence trees are built. Because throughout the tree combining phase, lemmas are handled rather than forms, much of the morpho-syntactic feature information which is necessary to block the construction of ill-formed constituents is simply not available. It is therefore meaningful to only include in the tree combining phase those features whose value is available at tree combining time. In a third experiment, we automatically identified those features from the observed feature structure unification failures during runs of the realisation algorithm. We then use only these features (in combination with the semantic features and with categorial information) during tree combining.

4 Evaluation

To evaluate the impact of the different optimisation techniques discussed in the previous section, we use two benchmarks generated automatically from SEMXTAG (Gottesman, 2009).

The first benchmark (MODIFIERS) was designed to test the realisers on cases involving intersective modifiers. It includes 1 789 input formulae with a varying number (from 0 to 4 modifications), type (N and VP modifications) and distribution of intersective modifiers (n modifiers distributed differently over the predicate argument structures). For instance, the formula in (1) involves 2 N and 1 VP modification. Further, it combines lexical ambiguity with modification complexities, i.e. for the *snore* modifier the grammar provides 10 trees.

(1) $l_1 : \exists(x_1, h_r, h_s), h_r \geq l_2, h_s \geq l_3, l_2 : man(x_1), l_2 : snoring(e_1, x_1), l_2 : big(x_1), l_3 : sleep(e_2, x_1), l_4 : soundly(e_2)$
(*A snoring big man sleeps soundly*)

The second benchmark (COMPLEXITY) was designed to test overall performance on cases of differing complexity (input formulae of increasing length, involving verbs with a various number and types of arguments and with a varying number of and types of modifiers). It contains 890 distinct cases. A sample formula extracted from this benchmark is shown in (2), which includes one modification and to different verb types.

(2) $h_1 \geq l_4, l_0 : want(e, h_1), l_1 : \exists(x_1, h_r, h_s), h_r \geq l_1, h_s \geq l_0, l_1 : man(x_1), l_1 : snoring(e_1, x_1), l_3 : \exists(x_2, h_p, h_w, h_u), h_p \geq l_3, h_w \geq l_4, h_u \geq l_5, l_3 : monkey(x_2), l_4 : eat(e_2, x_2, e_3), l_5 : sleep(e_3, x_2)$
(*The snoring man wants the monkey to sleep*)

To evaluate GENI and the various configurations of RTGEN (RTGEN-all, RTGEN-level0, RTGEN-selective), we ran the 4 algorithms in batch mode on the two benchmarks and collected the following data for each test case:

- Packed chart size : the number of chart items built. This feature is only aplicable to RTGen as GENI does not implement packing.

- Unpacked chart size : the number of intermediate and final structures available after unpacking (or at the end of the tree combining process in the case of GENI).
- Initial Search Space (ISS) : the number of all possible combinations of elementary trees to be explored given the result of lexical selection on the input semantics. That is, the product of the number of FTAG elementary trees selected by each literal in the input semantics.
- Generation forest (GF) : the number of derivation trees covering the input semantics.

The graph in Figure 2 shows the differences between the different strategies with respect to the unpacked chart size metric.

A first observation is that RTGEN-all outperforms GENI in terms of intermediate structures built . In other words, the Earley sharing and packing strategy is more effective in reducing the number of constituents built than the filtering and substitution-before-adjunction optimisations used by GENI. In fact, even when no feature information is used at all (RTGEN-level0 plot), for more complex test cases, packing and sharing is more effective in reducing the chart size than filtering and operation ordering.

Another interesting observation is that RTGEN-all and RTGEN-selective have the same impact on chart size (their plots coincide). This is unsurprising since the features used by RTGEN-selective have been selected based on their ability to block constituent combination. The features used in RTGEN-selective mode are `wh`, `xp`, `assign-comp`, `mode`, `definite`, `inv`, `assign-case`, `rel-clause`, `extracted` and `phon`, in addition to the categorial and semantic information. In other words, using all 42 SEMXTAG grammar features has the same impact on search space pruning as using only a small subset of them. As explained in the previous section, this is probably due to the fact that contrary to parsing, surface realisation only accesses the morphological lexicon after tree combining takes place. Another possibility is that the grammar is under constrained and that feature values are missing thereby inducing overgeneration.

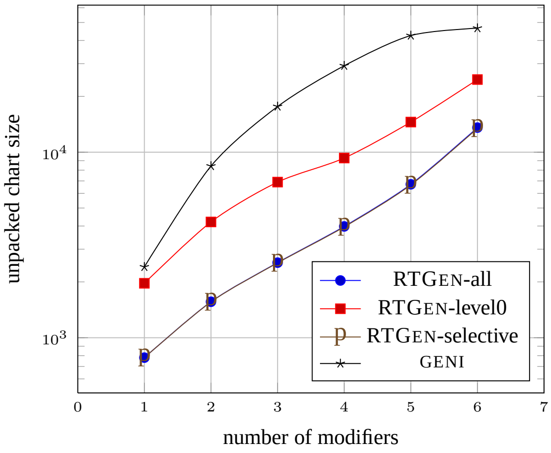


Figure 2: Performance of realisation approaches on the MODIFIERS benchmark, average unpacked chart size as a function of the number of modifiers.

Zooming in on cases involving three modifiers, we show in Table 1 the average results for various efficiency metrics [2]. This provides a more detail view of the performance of the differences among the three RTGEN variants.

strategy	GF	chart	unpacked-chart	seconds
RTGen-all	15.05	918.31	2,538.98	0.99
RTGen-level0	1,118.06	2,018	6,898.28	1.41
RTGen-selective	27.08	910.34	2,531.23	0.44

Table 1: Average results on 610 test cases from the MODIFIERS benchmark. Each test case has 3 modifications, distributed in various ways between adjectival and adverbial modifications. The second column, Generation Forest (GF), is the number of derivation trees present in the generated parse forest. The third and fourth columns show the chart and unpacked chart sizes, respectively. The last column shows the runtime in seconds.

This data shows that running RTGEN with no feature information leads not only to an increased chart size but also to runtimes that are higher in average than for full surface realisation i.e., realisation using the full grammar complete with con-

[2] The two realisers being implemented in different programming languages (RTGEN uses Prolog and GENI Haskell), runtimes comparisons are not necessarily very meaningful. Additionally, GENI does not provide time statistics. After adding this functionality to GENI, we found that overall GENI is faster on simple cases but slower on more complex ones. We are currently working on optimising RTGEN prolog implementation before carrying out a full scale runtime comparison.

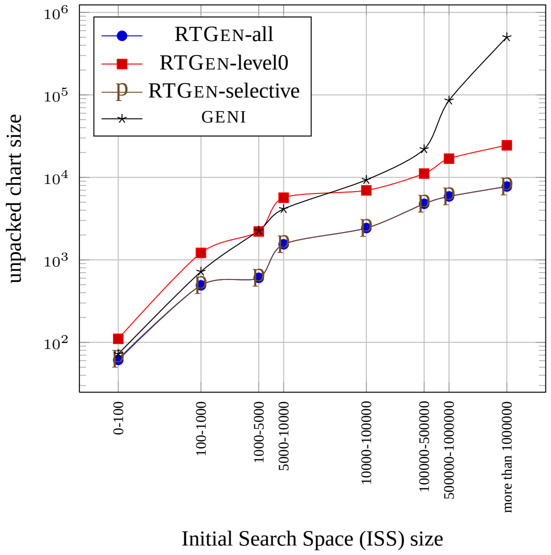


Figure 3: Performance of realisation approaches on the COMPLEXITY benchmark, average unpacked chart size as a function of the ISS complexity.

straints.

Interestingly, it also shows that the selective mode (RTGEN-selective) permits improving runtimes while achieving almost perfect disambiguation in that the average number of derivation trees (GF) produced is close to that produced when using all features. The differences between the two generation forests stems from packing. Using only a subset of features favors packing, thereby reducing the number of chart items built, but introduces over- generation.

Graph 3 and Table 2 confirm the results obtained using the MODIFIERS benchmark on a testset (COMPLEXITY) where input complexity varies not only with respect to modification but also with respect to the length of the input and to the degree of lexical ambiguity. Typically, in a TAG, one word or one semantic literal may be associated either with one tree or with up to several hundred trees (e.g., ditransitive verbs and verbs with several subcategorisation types). By varying the type and the number of verbs selected by the semantic literals contained in the input semantics, the COMPLEXITY benchmark provides a more extensive way to test performance on cases of varying complexity.

strategy	GF	chart	unpacked-chart	seconds
RTGen-all	14.77	693.39	2,427.82	0.81
RTGen-level0	162.02	2,114.16	6,954.84	1.09
RTGen-selective	15.31	692.9	2,427.2	0.36

Table 2: Average results on 335 cases with $10000 < ISS \leq 100000$, from the COMPLEXITY benchmark. The columns show the same performance metrics as in Table 1.

5 Related work

Much work has already been done on optimising surface realisation. Because surface realisation often draws on parsing techniques, work on parsing optimisation is also relevant. In this section, we briefly relate our proposal to another grammar converting approach (Koller and Striegnitz, 2002); to another chart based approach (Carroll and Oepen, 2005); and to approaches based on statistical pruning (White, 2004; Bangalore and Rambow, 2000).

5.1 Optimising surface realisation

Encoding into another grammatical formalism. As already mentioned, the RTGEN approach is closely related to the work of (Koller and Striegnitz, 2002) where the XTAG grammar is converted to a dependency grammar capturing its derivation trees. This conversion enables the use of a constraint based dependency parser, a parser which was specifically developed for the efficient parsing of free word order languages and is shown to support an efficient handling of both lexical and modifier attachment ambiguity.

Our proposal differs from this approach in three main ways. First, contrary to XTAG, SEMXTAG integrates a full-fledged, unification based compositional semantics thereby allowing for a principled coupling between semantic representations and natural language expressions. Second, the grammar conversion and the feature-based RTGs used by RTGEN accurately translates the full range of unification mechanisms employed in FTAG wheras the conversion described by (Koller and Striegnitz, 2002) does not take into account feature structure information. Third, the RTGEN approach was extensively tested on a large benchmark using 3 different configurations whilst (Koller and Striegnitz, 2002) results are restricted to a few hand constructed example inputs.

Chart generation algorithm optimisations. (Carroll and Oepen, 2005) provides an extensive and detailed study of how various techniques used to optimise parsing and surface realisation impact the efficiency of a surface realiser based on a large coverage Head-Driven Phrase Structure grammar.

Because they use different grammars, grammar formalisms and different benchmarks, it is difficult to compare the RTGEN and the HPSG approach. However, one point is put forward by (Carroll and Oepen, 2005) which it would be interesting to integrate in RTGEN(Carroll and Oepen, 2005) show that for packing to be efficient, it is important that equivalence be checked through subsumption, not through equality. RTGEN also implements a packing mechanism with subsumption check, i.e. different ways of covering the same subset of the input semantics are grouped together and represented in the chart by the most general one. One difference however it that RTGEN will pack analyses together as long as the new ones are more specific cases. It will not go backwards to recalculate the packing made so far if a more general item is found (Stefan and John, 2000). In this case the algorithm will pack them under two different groups.

Statistical pruning. Various probabilistic techniques have been proposed in surface realisation to improve e.g., lexical selection, the handling of intersective modifiers or ranking. For instance, (Bangalore and Rambow, 2000) uses a tree model to produce a single most probable lexical selection while in White's system, the best paraphrase is determined on the basis of n-gram scores. Further, to address the fact that there are $n!$ ways to combine any n modifiers with a single constituent, (White, 2004) proposes to use a language model to prune the chart of identical edges representing different modifier permutations, e.g., to choose between *fierce black cat* and *black fierce cat*. Similarly, (Bangalore and Rambow, 2000) assumes a single derivation tree that encodes a word lattice (*a* {*fierce black, black fierce*} *cat*), and uses statistical knowledge to select the best linearisation. Our approach differs from these approaches in that lexical selection is not filtered, intersective modifiers are handled by the grammar (constraints on the respective order of adjectives) and the chart packing strategy (for optimisation), and ranking is not performed. We are currently exploring the use of Optimality Theory for ranking.

6 Conclusion

We presented RTGEN, a novel surface realiser for FTAG grammars which builds on the observation that an FTAG can be translated to a regular tree grammar describing its derivation trees. Using automatically constructed benchmarks, we compared the performance of this realiser with that of GENI, another state of the art realiser for FTAG. We showed that RTGEN outperforms GENI in terms of space i.e. that the Earley sharing and packing strategy is more effective in reducing the number of constituents built than the filtering and substitution-before-adjunction optimisations used by GENI. Moreover, we investigated three ways of interleaving phrase structure and feature structure constraints and showed that, given a naive constraint solving approach, the interleaving approach with selective features seems to provide the best space/runtimes compromise.

Future work will concentrate on further investigating the interplay in surface realisation between phrase structure and feature structure constraints. In particular, (Maxwell and Kaplan, 1994) shows that a more sophisticated approach to constraint solving and to its interaction with chart processing renders the non interleaved approach more effective than the interleaved one. We plan to examine whether this observation applies to SEMXTAG and RTGEN. Further, we intend to integrate Optimality Theory constraints in RTGEN so as support ranking of multiple outputs. Finally, we want to further optimise RTGEN on intersective modifiers using one the methods mentioned in Section 5.

References

Bangalore, S. and O. Rambow. 2000. Using TAGs, a tree model and a language model for generation. In *Proceedings of TAG+5*, Paris, France.

Brew, Chris. 1992. Letting the cat out of the bag: generation for shake-and-bake mt. In *Proceedings* *of the 14th conference on Computational linguistics*, pages 610–616, Morristown, NJ, USA. Association for Computational Linguistics.

Carroll, J. and S. Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. *2nd IJCNLP*.

Carroll, J., A. Copestake, D. Flickinger, and V. Paznański. 1999. An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of EWNLG '99*.

Gardent, C. and L. Kallmeyer. 2003. Semantic construction in FTAG. In *10th EACL*, Budapest, Hungary.

Gardent, C. and E. Kow. 2007. Spotting overgeneration suspect. In *11th European Workshop on Natural Language Generation (ENLG)*.

Gottesman, B. 2009. Generating examples. Master's thesis, Erasmus Mundus Master Language and Communication Technology, Saarbrucken/Nancy.

Kay, Martin. 1996. Chart generation. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 200–204, Morristown, NJ, USA. Association for Computational Linguistics.

Koller, A. and K. Striegnitz. 2002. Generation as dependency parsing. In *Proceedings of the 40th ACL*, Philadelphia.

Maxwell, J. and R. Kaplan. 1994. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4).

Perez-Beltrachini, L. 2009. Using regular tree grammars to reduce the search space in surface realisation. Master's thesis, Erasmus Mundus Master Language and Communication Technology, Nancy/Bolzano.

Schmitz, S. and J. Le Roux. 2008. Feature unification in TAG derivation trees. In Gardent, C. and A. Sarkar, editors, *Proceedings of the 9th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+'08)*, pages 141–148, Tübingen, Germany.

Stefan, Oepen and Carroll John. 2000. Parser engineering and performance profiling. *Journal of Natural Language Engineering*, 6(1):81–98.

The XTAG Research Group. 2001. A lexicalised tree adjoining grammar for english. Technical report, Institute for Research in Cognitive Science, University of Pennsylvannia.

Vijay-Shanker, K. and AK Joshi. 1988. Feature Structures Based Tree Adjoining Grammars. *Proceedings of the 12th conference on Computational linguistics*, 55:v2.

White, M. 2004. Reining in CCG chart realization. In *INLG*, pages 182–191.

Automatically Learning Source-side Reordering Rules for Large Scale Machine Translation

Dmitriy Genzel
Google, Inc.
dmitriy@google.com

Abstract

We describe an approach to automatically learn reordering rules to be applied as a preprocessing step in phrase-based machine translation. We learn rules for 8 different language pairs, showing BLEU improvements for all of them, and demonstrate that many important order transformations (SVO to SOV or VSO, head-modifier, verb movement) can be captured by this approach.

1 Introduction

One of the major problems of modern statistical machine translation relates to its difficulties in producing the correct word order on the target side of the translation where the source side order is not the same as the target side. In many cases where the translation is spectacularly bad, if one only enters the source sentence in the word order of the target language the translation becomes near-perfect (largely because the language model can now make sense of it). The word order problems are especially extensive for languages that have major differences, such as SOV vs. SVO languages, but also cause insidious, but entirely avoidable errors for the language pairs where the word order is almost right, but not quite[1]. For practical reasons all phrase-based decoders limit the amount of reordering allowed and thus are completely unable to produce correct translations when the necessary movement is over a large distance. Furthermore, where the actual systematic reordering for the two languages is within the decoder's search space, it is penalized just as any other kind of reordering, whereas doing anything other than this systematic reordering should in fact be penalized.

It has been argued that this is a fundamental flaw in phrase-based decoding systems and hierarchical and syntax-based systems have been proposed to solve this problem. These systems can in principle resolve a part of this problem, but at a significant time cost during training, and even worse, during translation, making it less practical for realtime systems. Instead we propose a system for learning pre-ordering rules automatically from data and demonstrate that it can capture many different kinds of reordering phenomena and do so at no additional online cost.

2 Related Work

Many solutions to the reordering problem have been proposed, e.g. syntax-based models (Chiang, 2005), lexicalized reordering (Och et al., 2004), and tree-to-string methods (Zhang et al., 2006). All these methods try to solve the reordering problem in different ways, but have the following problems in common: word alignment is not affected by them and they tend to introduce significant additional work to be done at translation time. Most state of the art systems use HMM or IBM Model 4 word alignment, both of which have a penalty term associated with long distance jumps, and tend to misalign words which move far from their expected positions.

We are going to focus on the approaches where reordering is done as a preprocessing step (sometimes called pre-ordering). These approaches have the advantage that they are independent of the actual MT system used, are often fast to apply, and tend to decrease (due to improved quality of heuristic estimates) rather than dramatically increase the time spent in actual decoding, unlike

[1] For example of the latter kind, verb movement for English-German and similar language pairs often causes verbs to be aligned to nothing and to be altogether dropped in translation.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 376–384,
Beijing, August 2010

some of the previously mentioned approaches. The downside of these methods is that the reordering is fixed, and if it is wrong it can hurt the quality of translations. We will discuss solutions for this problem later.

Even in the relatively limited space of preprocessing-based reordering solutions, there has been a large amount of previous work, as far back as Brown et al. (1992). Most approaches focus on utilizing manually written rules for different languages. A common language pair for which rules were proposed is German-English (Nießen and Ney, 2001; Collins et al., 2005). There is similar work for Chinese-English (Wang et al., 2007) and quite a few other languages. Clearly, such methods work quite well, but require linguistic expertise to produce. Our goal, however, is to learn reordering from parallel data that is already available to an MT system in an entirely unsupervised manner.

We are not the first to attempt this task. In particular, Xia and McCord (2004) proposed a way to automatically learn reordering patterns for French-English. Their system parses parallel data both on the source and target side and then uses a variety of heuristics to extract reordering rules which are then applied during training. More recently, Li et al. (2007) use a maximum entropy system to learn reordering rules for binary trees (i.e., whether to keep or reorder for each node). An approach most similar to ours is that of Rottmann and Vogel (2007) where they learn reordering rules based on sequences of part-of-speech tags (but do not use parse trees). All of these approaches show improvements in translation quality, but are applied on a single language pair. Our goal is to find a method that works well for many language pairs, regardless of the word order transformations needed, and without language-specific tuning. Unlike our predecessors, we use a systematic search through the space of possible permutation rules to minimize a specific metric, related to the monotonicity of resulting alignments.

3 Our Approach

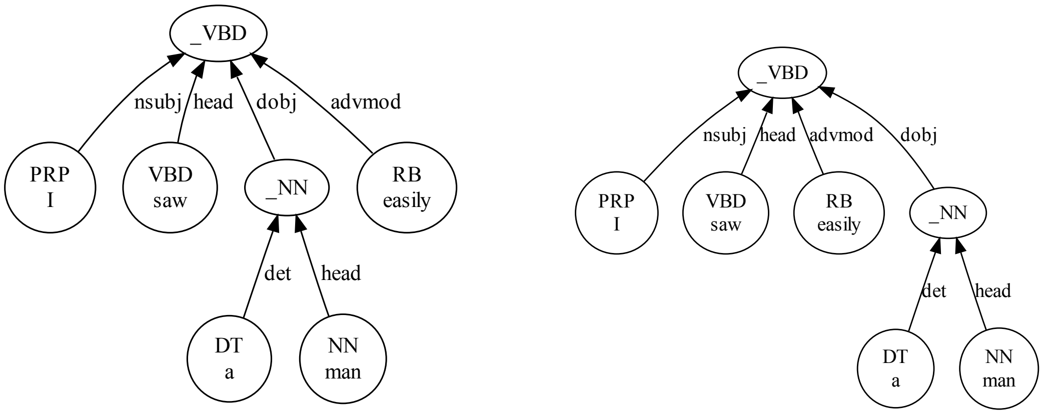
We limit ourselves to reorderings of the source side of training and test data. To constrain our reorderings, we first produce a parse tree, using a dependency parser similar to that of Nivre and Scholz (2004). The above parser is much faster than the time spent in translating the same sentence and thus creates almost no overhead. In our experiments where the source language is English the training data for the parser is the Penn Treebank (Marcus et al., 1993). For German, we use TIGER treebank (Brants et al., 2002). We then convert the dependency tree to a shallow constituent tree. The trees are annotated by both Penn Treebank part of speech tags and by Stanford dependency types (de Marneffe et al., 2006; de Marneffe and Manning, 2008). For an example, see Figure 1a.

Our reorderings are constrained by reordering of nodes in a parse tree of the source sentence. Thus, the full space of reorderings we consider consists of all reorderings that would produce a parse tree with the same set of child-parent relationships. For an example of a valid reordering, see Figure 1b.

Each reordering is described by a series of rules and we learn one such series for each language pair automatically. Each source sentence is parsed, and the tree is transformed sequentially, one rule at a time applying to the entire tree, top down. The reordered sentence is read off the leaves of the tree and training and evaluation proceeds as normal. We are using a state-of-the-art phrase-based statistical machine translation system to perform the actual translation. The system is itself capable of further local reordering during translation limited by the maximum distance of 4 words.

3.1 Rule Space

Each rule consists of two parts: conditioning context and action. For every internal node in the parse tree, traversed top-down, the node is matched against the conditioning context, and if a match is found, the associated action applies. All actions are limited to reordering children of the matching node. Furthermore, if a rule applies at a node, its descendants are not traversed for the purpose of matching to avoid modifying the same part of the sentence twice by the same rule. A different rule may apply on this node or its descendants



(a) A sample parse tree

(b) After reordering (moving RB over _NN)

Figure 1: Parse tree of a sentence and its reordering

Feature	Description
nT	POS tag of this node
nL	Syntactic label of this node
pT	POS tag of the parent of this node
pL	Syntactic label of the parent
1T	POS tag of the first child
1L	Label of the first child
2T	POS tag of the second child
2L	Label of the second child
...	...

Table 1: Set of features used as conditioning variables

later in the sequence.

A conditioning context is a conjunction of conditions. Each condition is a (feature, value) pair. List of features is given in table 1. In practice, we limit ourselves to no more than 4 conditions in a given context to avoid combinatorial explosion and sparsity as well as contexts that fail to generalize. However, we may exhaustively generate every possible conjunction of up to 5 conditions from this list that covers up to 4 children that we actually observe in training.

For example, the following contexts would be valid for transformation in Fig. 1:

- nT = _VBD
- 1T = PRP
- 1L = nsubj
- 3T = dobj
- etc.

or any conjunction of these. The action performed in this example is swapping children 3 and 4 of the _VBD node, and can be denoted as the permutation (1,2,4,3).

When processing a rule sequence, once a rule applies, the action is performed, and that rule is no longer applied on the same node or its descendants (but can be further applied elsewhere in the tree). Another rule (even an identical one) starts from the top and can apply to nodes modified by previous rules.

3.2 Reordering metrics

To evaluate the quality of a given reordering rule, we need to have reliable metrics that, for each sentence pair, can evaluate whether an improvement in monotonicity has been made.

The easiest metric to use is the number of crossing alignment links for a given aligned sentence pair. For instance, in Figure 2, there are 2 crossing links. This metric is trivial to compute and has some nice properties. For instance, moving a single word one position out of place causes one link

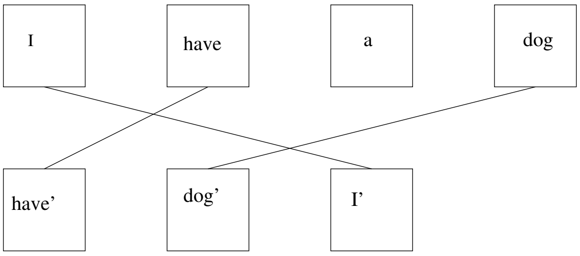


Figure 2: Counting crossing alignment links

to cross, moving it farther away from its correct position would cause more links to cross. We will refer to this metric as *crossing score*.

An ideal metric would be the actual BLEU score that the system would obtain under this reordering rule on the development set. However, since each rule affects word alignment, phrase extraction, optimal feature weights, and the actual translation, it would be necessary to retrain the entire phrase-based system for each possible rule, which is impractical. It is, however, practical, to retranslate the development set, keeping the phrase table and feature weights constant. Normally, however, phrase tables contain multi-word phrases, such as "a b" which may no longer match after the reordering, and this biases the system toward the original word order. To avoid this, for this computation only, we use a phrase table that only contains single words and is therefore independent of the source sentence word order. This lets us test whether a given reordering improves the search space for the phrase-based decoder at the relatively small computational cost of translating the development set. We obtain a difference of the BLEU scores with and without a given rule, which we hope to be a reasonable estimate of the true gain in BLEU score that one would obtain, by retraining the full system, including word alignment, full-length phrase extraction, and tuning the feature weights. We refer to this score as *estimated BLEU gain*.

Note that these two scores are used to obtain an estimate of utility of any given rule, and are not used for evaluation of the entire system. Those metrics are discussed in detail in the evaluation section.

3.3 Algorithm

We propose a straightforward algorithm to automatically learn reordering rules. The input data for all algorithms is word-aligned sentence pairs. We have found that sophisticated alignment models introduce a bias toward alignment between certain kinds of nodes (usually ones that are close), and this has undesirable effects. In practical terms this means that neither HMM nor Model 4 alignments are useful (even though they are better as alignments), but Model 1 alignments are. However, to compensate for poor quality of the alignments, we simply delete those alignment links that have posterior probabilities under 0.5[2] and remove sentence pairs which have very few alignments left. The crossing score works quite well even when only a portion of the words in a sentence are aligned.

The algorithm's outline is given as Alg. 1.

The algorithm proceeds by considering all rules after the best sequence of rules so far, and appends the best new rule (according to the metric) to the sequence. In practice, some changes are needed, and we describe some variations. Each of these variations produces a different sequence of rules, but they are interchangeable, and we can simply pick one that performs best on the development set, or to combine them through multi-source translation or consensus.

In all variations, we are unable to generate all possible rules for every sentence, as the number can easily be 10^4-10^6 per sentence. It is sufficient, however, to take a random sample of the input, extract top candidates, and reevaluate those on the entire set.

We also limit the kinds of rules we are allowed to generate. The number of possible actions on a node with n children is $n! - 1$ and our trees are quite shallow, often containing 5, 6, or even more children per node. To avoid dealing with explosion of rules and the resulting sparsity of the rule space, we modify the process slightly, so that instead of matching a node, we match a node and a consecutive subsequence of its children of a given size, as a sliding window. For example, in Figure 1a, node _VBD has 4 children. If we limit our-

[2] This guarantees only one alignment per word

Algorithm 1 Optimizing alignment links

```
input: A set of aligned sentence pairs
base = <empty sequence>;
for several iterations do
    candidate_rules = GenerateAllCandidateRules(input, base);
    base.append(MinCost(candidate_rules))
end for
```

selves to 3 children at a time we would attempt to match this node twice: with its children 1,2,3 and 2,3,4. In other words, we pretend to consider two nodes, one with the first set of children, and one with the second, proceeding left to right. If either one matches, we apply the action to the subset of children in the window and stop processing the node further.

It is also useful to produce more than one rule per iteration, although this can be problematic, since the rules may interfere with each other.

3.3.1 Variant 1: Optimizing crossing score

We start with the initially empty base sequence. As described above, we generate every possible rule from a subset of sentences, and evaluate them on the entire input, with the base sequence always applied first. We use crossing score as a metric. However, instead of extracting only one best-scoring rule, we extract K best. Now we need to obtain a decorrelated set: for every pair of rules, we count the number of sentences where they both apply. For every rule we consider all rules that are ranked higher, and if the percentage of matches between these two rules is high, the rules may interfere with each other, and the current rule is dropped. We thus obtain a small ordered set of rules that tend to apply on different sentences, and should not interfere with each other. From this ordered set we produce all candidate rule subsequences and evaluate them, to ensure there really is no interference. The one with the best score is then appended to the base sequence. The process is then repeated with a new base sequence.

3.3.2 Variant 2: Optimizing Estimated BLEU gain

We proceed as in the previous variant, but final evaluation of potential sequences to be appended is done differently. Instead of using a crossing score, we reorder the development set with each candidate rule sequence and score it using a translation system with a fixed phrase table with single word phrases only (to avoid bias for a specific word order). The sequence with the highest BLEU is then appended to base sequence, and the process is repeated.

3.3.3 Variant 3: Optimizing Estimated BLEU gain in sequence

In this variant, once we obtain a set of decorrelated candidate rules $\{a_1, a_2, \ldots a_n\}$ ordered by crossing score, we evaluate the following rule sequences (where b is base sequence): $(b), (b, a_1), (b, a_1, a_2) \ldots (b, a_1, \ldots a_n)$ using estimated BLEU gain, as above. If we find that for some k, $score(b, a_1, \ldots a_{k-1}) > score(b, a_1, \ldots a_{k-1}, a_k)$, that means that a_k interferes with preceding rules. We remove all such a_k, and retranslate/rescore until the score sequence is monotonically non-decreasing. At this point, we append all surviving rules to the base sequence, and repeat the process.

4 Evaluation

As described above, our base system is a phrase-based statistical MT system, similar to that of Och and Ney (2004). The baseline decoder is capable of local reordering of up to 4 words. Our training data is extracted by mining from the Web, as well as from other published sources. We train systems from English to 7 other languages, as well as German-English. We chose them as follows: SOV languages (Japanese, Korean, Hindi), VSO language (Welsh), long distance verb movement (German), noun-modifier issues (Russian and Czech). The amount of training data varies from 28 million words (for Hindi) to 260 million (for German). The baseline system is a production-quality system used by a large number of users.

For the first set of experiments for German-English and English-German we use WMT-09 data sets for development and testing (Callison-Burch et al., 2009). We report BLEU scores for each of the algorithms along with the best score from the WMT-09 workshop for reference in Table 2.

Unfortunately, there is no standard data set for most of the languages we would like to experiment with. For the second set of experiments, we use an unpublished data set, containing data in English and 7 languages mentioned above. Our test data comes from two sources: news articles from WikiNews[3] (996 sentences) and a set of random sentences from the web (9000 sentences). From these, we create 3 sets: *dev1*: 3000 sentences from *web* and 486 sentences from *wiki*; *dev2*: 1000 sentences from *web*; and *test*: the remainder of *web* (5000 sentences) and *wiki* (510 sentences). The *dev1* set is used for tuning the system, both *dev1* and *dev2* for tuning consensus, and the *test* set for evaluation. These sets are the same for all 7 languages.

Discriminative minimum error rate training (Macherey et al., 2008) was applied to optimize the feature weights for each system.

We evaluate the three variants of the algorithm mentioned above. Each algorithm outputs a reordering rule sequence (40-50 rules long) which is applied to all the training and test data, and a complete system is trained from scratch.

There is no need for us to pick a single algorithm for all language pairs, since each algorithm produces rules that are compatible with each other. We are able to pick the algorithm that works best on the development set for each language pair.

In addition, we can use a decoder that is capable of performing a multi-input translation which is given the unreordered input as well as the three reordered inputs produced by the above algorithm. This decoder is able to learn separate feature weights for each feature/algorithm combination.

Finally, we can use consensus translation (Macherey and Och, 2007) to produce the best possible translation for each sentence.

Table 4: Manual vs. automatic reordering. *Automatic* score is the combined score from Table 3.

Language	Base	Manual	Automatic	Diff
Hindi	16.85	19.25	19.36	0.11
Japanese	25.91	28.78	29.12	0.34
Korean	23.61	27.99	27.91	-0.08

Results using BLEU score (character-level for Japanese and Korean, word-level for other languages) for English to X systems are given in Table 3, along with the score of Google Translate as of Feb 15, 2010, for expected quality reference. All gains in the combined and consensus columns are statistically significant using a bootstrap resampling test (Noreen, 1989).

We should also note that the parsing and reordering overhead was an average of 10msec per sentence, and had no appreciable impact on the speed of the system.

4.1 Comparison with manual reordering

We also compared our automatic method with a manually written reordering rule set for SOV languages (Xu et al., 2009) (rules initially written for Korean) for comparison with our approach. The results are given in Table 4. The results are mostly comparable, with automatic rules being better for two of the three languages.

4.2 Turning off decoder reordering

All of the above experiments allowed the decoder to further reorder the sentence as needed. Reordering in the decoder creates an exponential increase in the search space, and for a typical decoding strategy can lead to increase in decoding time, search errors, or both. Since we already preorder the sentence, it should be possible to avoid reordering in the decoder altogether.

Results for the combined decoder are given in Table 5. It contains the gain of the combined decoder against the baseline from Table 3, and the gain when decoder reordering is turned off against the same baseline (which has decoder reordering on). For many languages it is indeed now possi-

[3] http://en.wikinews.org

Table 2: Results for 3 algorithms on WMT-09 data with best individual system score from the workshop: for EN to DE, Edinburgh, for DE to EN, Google

Language	Base	Var. 1	Var. 2	Var. 3	Best workshop
EN to DE	16.09	16.30	16.35	**16.40**	*14.76*
DE to EN	21.00	**22.45**	22.13	22.05	*20.23*

Table 3: Results on internal test set for 3 systems (Variant 1,2,3), the variant which performed best on the development set, the combined system, and the consensus run, along with Google Translate scores (Feb 15, 2010) for reference

Language	Google	Base	Var. 1	Var. 2	Var. 3	Best on dev	Combined	Consensus
	%BLEU	%BLEU	gain	gain	gain	gain	gain	gain
Czech	16.68	15.35	-0.08	0.13	**0.19**	0.19	0.21	0.21
German	20.34	18.65	**0.47**	0.30	0.39	0.39	0.72	0.73
Hindi	19.15	16.85	**2.25**	2.08	0.15	2.08	2.51	2.47
Japanese	30.74	25.91	**3.05**	2.60	**3.05**	3.05	3.21	3.03
Korean	27.99	23.61	3.34	3.77	**4.16**	4.16	4.30	4.30
Russian	16.80	15.33	0.08	**0.10**	**0.10**	0.08	0.14	0.23
Welsh	27.38	25.48	1.25	0.77	**1.43**	1.43	1.34	1.63

Table 5: Disallowing decoder reordering: difference against baseline in %BLEU gain

Language	Decoder reordering	No decoder reordering
Czech	0.21	0.08
German	0.72	0.55
Hindi	2.51	2.27
Japanese	3.21	3.21
Korean	4.30	4.15
Russian	0.14	-0.10
Welsh	1.34	0.98

ble to avoid decoder reordering altogether which leads to a significant speedup.

5 Analysis

We looked at the rules being learned as well as at the differences in the output to see if the gains in BLEU are in fact due to the reordering phenomena being resolved. The top rules for each language are given in Table 6.

One can observe that the top rules for German and Slavic languages are as expected: verb movement and noun modifier reordering. Other top rules for German cover other specific cases of verb movement, other rules for Czech include, for example, movement of the subject of the passive sentence to the right and movement of the possessive (which is similar to the noun compound case).

The rules for Welsh include movement of the adjective modifier over its head (given in the table above) and other rules moving noun modifiers, moving a modal verb left over its subject, moving determiners to the right of nouns, etc.

For Japanese and Korean, there are many rules with dramatic impact, such as a rule moving all heads to the right, reversing a sequence of three nodes starting with a modal (e.g. *can do something* to *something do can*), moving numerical modifiers to the right of their heads, and many others.

Hindi is also an SOV language, but its grammar is not as similar to Japanese or Korean as they are to each other. Still, Hindi also has some similar rules, but there are many more involving verb movement, such as a rule directly moving the verb to the final position.

By looking at the sentences produced by the system we can see that the differences are dramatic for SOV and VSO languages, as expected,

Table 6: Examples of top rules and their application

Languages	Context	Order	Example
Hindi	1L:head 3L:none	2,1,3	*I see him → I him see*
Japanese, Korean	2L:prep	2,1	*eat with a spoon → eat a spoon with*
German	1T:VBN 2L:prep	2,1	*struck with a ball → with a ball struck*
Russian, Czech	1L:nn 2L:head	2,1	*a building entrance → a entrance building*
Welsh	1L:amod 2L:head	2,1	*blue ball → ball blue*

but more interestingly, most German sentences now have a verb where the baseline had none. Another profound effect can be observed for Russian: the baseline almost invariably translated noun compounds incorrectly: e.g. *group leaders* may be translated as *group of-leaders* since this requires no reordering and no preposition insertion. This is especially problematic, since the user of the translation system often cannot detect this: the resulting sentence is not ungrammatical and can even make sense. Our algorithm learns a rule that prevents this from happening. Now the decoder must pay a cost to keep the order the same as in English.

6 Discussion and Future Work

We have demonstrated a general technique which requires only access to a parser for the source language (in addition to parallel data which already exists for an MT system) and is capable of reducing reordering problems endemic in a phrase-based system. No linguists or even native speakers of any of these languages were needed to write the rules. The algorithm is quite robust and performs well on noisy web data, much of it being ungrammatical.

All variants turned out to perform well, although variants 1 and 3 were better most of the time. We consider all variants to be useful, since they find different local maxima under different objective functions, and in practice use all of them and pick a rule sequence that performs best on the development set for any specific language pair.

We plan to explore this research area further in several ways. First, it would be interesting to experiment with applying rules learned for one language to a related language, e.g. Portuguese for Spanish or German for Dutch. This would let us use rules learned from a major language for a minor one with less available training data.

We have only used English and German as source languages. There is training data for parsers in other languages, and this approach should work well for most source languages. Where a source language parser is not available, we can still improve quality, by learning rules from the target side and applying them only for the purpose of improving word alignment. Improving word alignment alone would not help as much as also using the reordering in the decoder, but it will probably help in extracting better phrases. We also plan to use parser projection to induce a reasonable quality parser for other languages.

References

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *In Proceedings of the Workshop on Treebanks and Linguistic Theories*, pages 24–41.

Peter F. Brown, Stephen A. Della, Pietra Vincent, J. Della Pietra, John D. Lafferty Robert, and L. Mercer. 1992. Analysis, statistical transfer, and synthesis in machine translation. In *Proceedings of the Fourth International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 83–100.

Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. 2009. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece, March. Association for Computational Linguistics.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the ACL'05*, pages 263–270, Ann Arbor, Michigan, June.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the ACL'05*, pages 531–540, Ann Arbor, Michigan, June.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representations. In *COLING'08 Workshop on Cross-framework and Cross-domain Parser Evaluation*, Manchester, England, August.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure trees. In *LREC*.

Chi-Ho Li, Minghui Li, Dongdong Zhang, Mu Li, Ming Zhou, and Yi Guan. 2007. A probabilistic approach to syntax-based reordering for statistical machine translation. In *Proceedings of the ACL-07*, pages 720–727, Prague, Czech Republic, June.

Wolfgang Macherey and Franz J. Och. 2007. An empirical study on computing consensus translations from multiple machine translation systems. In *Proceedings of the EMNLP-CoNLL'07*, pages 986–995, Prague, Czech Republic, June.

Wolfgang Macherey, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the EMNLP-2008*, pages 725–734, Honolulu, Hawaii, October.

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

Sonja Nießen and Hermann Ney. 2001. Morpho-syntactic analysis for reordering in statistical machine translation. In *Machine Translation Summit*, pages 247–252, Santiago de Compostela, Spain, September.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of Coling 2004*, pages 64–70, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley & Sons, Canada.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Franz Josef Och, Daniel Gildea, Sanjeev Khudanpur, Anoop Sarkar, Kenji Yamada, Alex Fraser, Shankar Kumar, Libin Shen, David Smith, Katherine Eng, Viren Jain, Zhen Jin, and Dragomir Radev. 2004. A smorgasbord of features for statistical machine translation. In *HLT-NAACL 2004: Main Proceedings*, pages 161–168, Boston, Massachusetts, USA, May 2 - May 7.

Kay Rottmann and Stephan Vogel. 2007. Word reordering in statistical machine translation with a pos-based distortion model. In *Proceedings of TMI*, Skovde, Sweden.

Chao Wang, Michael Collins, and Philipp Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proceedings of the EMNLP-CoNLL'2007*, pages 737–745, Prague, Czech Republic, June.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of Coling 2004*, pages 508–514, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve SMT for subject-object-verb languages. In *Proceedings of NAACL-HLT'09*, Boulder, Colorado.

Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 256–263, New York City, USA, June. Association for Computational Linguistics.

Comparing Language Similarity across Genetic and Typologically-Based Groupings

Ryan Georgi
University of Washington
rgeorgi@uw.edu

Fei Xia
University of Washington
fxia@uw.edu

William Lewis
Microsoft Research
wilewis@microsoft.com

Abstract

Recent studies have shown the potential benefits of leveraging resources for resource-rich languages to build tools for similar, but resource-poor languages. We examine what constitutes "similarity" by comparing traditional phylogenetic language groups, which are motivated largely by genetic relationships, with language groupings formed by clustering methods using typological features only. Using data from the World Atlas of Language Structures (WALS), our preliminary experiments show that typologically-based clusters look quite different from genetic groups, but perform as good or better when used to predict feature values of member languages.

1 Introduction

While there are more than six thousand languages in the world, only a small portion of these languages have received substantial attention in the field of NLP. With the increase in use of data-driven methods, languages with few or no electronic resources have been difficult to process with current methods. The morphological tagging of Russian using Czech resources as done by (Hana et al., 2004) shows the potential benefit for using the resources of resource-rich languages to bootstrap NLP tools for related languages. Projecting syntactic structures across languages (Yarowsky and Ngai, 2001; Xia and Lewis, 2007) is another possible way to harness existing tools, though such projection is more reliable among languages with similar syntax.

Studies such as these show the possible benefits of working with similar languages. A crucial question is how we should define similarity between languages. While genetically related languages tend to have similar typological features as they could inherit the features from their common ancestor, they could also differ a lot due to language change over time. On the other hand, languages with no common ancestor could share many features due to language contact and other factors.

It is worth noting that the goals of historical linguistics differ from those of language typology in that while historical linguistics focuses primarily on diachronic language change, typology is more focused on a synchronic survey of features found in the world's languages: what typological features exist, where they are found, and why a language has a feature.

These differences between the concepts of genetic relatedness and language similarities lead us to the following questions:

Q1. If we cluster languages based only on their typological features, how do the induced clusters compare to phylogenetic groupings?

Q2. How well do induced clusters and genetic families perform in predicting values for typological features?

Q3. What typological features tend to stay the same within language families, and what features are likely to differ?

These questions are the focus of this study, and for the experiments, we use information from World Atlas of Language Structures (Haspelmath et al., 2005), or WALS.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 385–393,
Beijing, August 2010

ID#	Feature Name	Category	Feature Values
1	Consonant Inventories	Phonology (19)	{1:Large, 2:Small, 3:Moderately Small, 4:Moderately Large, 5:Average}
23	Locus of Marking in the Clause	Morphology (10)	{1:Head, 2:None, 3:Dependent, 4:Double, 5:Other}
30	Number of Genders	Nominal Categories (28)	{1:Three, 2:None, 3:Two, 4:Four, 5:Five or More}
58	Obligatory Possessive Inflection	Nominal Syntax (7)	{1:Absent, 2:Exists}
66	The Perfect	Verbal Categories (16)	{1:None, 2:Other, 3:From 'finish' or 'already', 4:From Possessive}
81	Order of Subject, Object and Verb	Word Order (17)	{1:SVO, 2:SOV, 3:No Dominant Order, 4:VSO, 5:VOS, 6:OVS, 7:OSV}
121	Comparative Constructions	Simple Clauses (24)	{1:Conjoined, 2:Locational, 3:Particle, 4:Exceed}
125	Purpose Clauses	Complex Sentences (7)	{1:Balanced/deranked, 2:Deranked, 3:Balanced}
138	Tea	Lexicon (10)	{1:Other, 2:Derived from Sinitic 'cha', 3:Derived from Chinese 'te'}
140	Question Particles in Sign Languages	Sign Languages (2)	{1:None, 2:One, 3:More than one}
142	Para-Linguistic Usages of Clicks	Other (2)	{1:Logical meanings, 2:Affective meanings, 3:Other or none}

Table 1: Sample features and their values used in the WALS database. There are eleven feature categories in WALS, one feature from each is given here. The numbers in parentheses in the 'Category' column are the total number of features in that category. Feature values are given with both the integers that represent them in the database and their description in the form {#:description}.

2 WALS

The WALS project consists of a database that catalogs linguistic features for over 2,556 languages in 208 language families, using 142 features in 11 different categories.[1] Table 1 shows a small sample of features, one feature from each category in WALS. Listed are the ID number for each example, the feature category, and the possible values for that feature.

WALS as a resource, however, is primarily designed for surveying the distribution of particular typological features worldwide, not comparing languages. The authors of WALS compiled their data from a wide array of primary sources, but these sources do not always cover the same sets of features or languages.

If we conceive of the WALS database as a two-dimensional matrix with languages along one dimension and features along the other, then only 16% of the cells in that matrix are filled. An empty cell in the matrix means the feature value for the (language, feature) pair is *not-specified (NS)*. Even well-studied languages could have many empty cells in WALS, and this kind of data sparsity presents serious problems to clustering algorithms that cannot handle unknown values. To address the data sparsity problem, we experiment with different pruning criteria to create a new matrix that is reasonably dense for our study.

2.1 Pruning Methods

Answering questions Q1–Q3 is difficult if there are too many empty cells in the data. Pruning the data to produce a smaller but denser subset can be done by one or more of the following methods.

Prune Languages by Minimum Features

Perhaps the most straightforward method of pruning is to eliminate languages that fail to contain some minimum number of features. Following Daumé (2009), we require languages to have a minimum of 25 features for the whole-world set, or 10 features for comparing across subfamilies. This eliminates many languages that simply do not have enough features to be adequately represented.

Prune Features by Minimum Coverage

The values for some features, such as those specific to sign languages, are provided only for a very small number of languages. Taking this into account, in addition to removing languages with a small number of features, it is also helpful to remove features that only cover a small portion of languages. Again we choose the thresholds selected by Daumé (2009) for pruning features that do not cover more than 10% of the selected languages in the whole-world set, and 25% in comparisons across subfamilies.

Use a Dense Language Family

Finally, using a well-studied family with a number of subfamilies can produce data sets with less sparsity. When clustering methods are used with this data, the groups correspond to subfamilies

[1] Our copy of the database was downloaded from `http://wals.info` in June of 2009 and appears to differ slightly from the statistics given on the website at the time of writing. Currently, the WALS website reports 2,650 languages, with 141 features in use.

Data Set	Min Features	Min Coverage	Grouped By	# Langs	# Groups	# Features	Density
Unpruned	0	0%	Family	2556	208	142	16.0%
Whole-World	25	10%	Family	735	121	139	39.7%
Indo-European	10	25%	Subfamily	87	10	64	44.9%
Sino-Tibetan	10	25%	Subfamily	96	14	64	38.6%

Table 2: Data sets and pruning options used for this paper. Density = $\frac{|Filled\ Cells|}{|Total\ Cells|} \cdot 100$

rather than families. In this study, we choose two families: Indo-European and Sino-Tibetan.

The resulting data sets after various methods of pruning can be seen in Table 2.

2.2 Features and Feature Values

Besides dealing with the sparsity of the features, the actual representation of the features in WALS needs to be taken into account. As can be seen in Table 1, features are represented with a range of discrete integer values. Some features, such as #58–Obligatory Possessive Inflection–are essentially binary features with values "`Absent`" or "`Exists`". Others, such as #1–Consonant Inventories–appear to be indices along some dimension related to size, ranging from small to large. Features such as these might conceivably be viewed as on a continuum where closer distances between values suggests closer relationship between languages.

Still other features, such as #81–Order of Subject, Object, and Verb–have multiple values but cannot be clearly be treated using distance measures. It's unclear how such a distance would vary between an SOV language and either VSO or VOS languages.

Binarization

Clustering algorithms use similarity functions, and some functions may simply check whether two languages have the same value for a feature. In these cases, no feature binarization is needed. If a clustering algorithm requires each data point (a language in this case) to be presented as a feature vector, features with more than two categorical values should be binarized. We simply treat a feature with k possible values as k binary features. There are other ways to binarize features. For instance, Daumé (2009) chose one feature value as the "canonical" value and grouped the other values into the second value (personal communication). We did not use this approach as it is not clear to us which values should be selected as the "canonical" ones.

3 Experimental Setup

To get a picture of how clustering methods compare to genetic groupings, we looked at three elements: cluster similarity, prediction capability, and feature selection.

3.1 Clustering

Our first experiment is designed to address question Q1: how do induced clusters compare to phylogenetic groupings?

Clustering Methods

For clustering, two clustering packages were used. First, we implemented the k-medoids algorithm, a partitional algorithm similar to k-means, but using median instead of mean distance for cluster centers (Estivill-Castro and Yang, 2000).

Second, we used a variety of methods from the CLUTO (Steinbach et al., 2000) clustering toolkit: repeated-bisection (`rb`), a k-means implementation (`direct`), an agglomerative algorithm (`agglo`) using UPGMA to produce hierarchical clusters, and `bagglo`, a variant of `agglo`, which biases the agglomerative algorithm using partitional clusters.

Similarity Measures

For similarity measures, we used CLUTO's default cosine similarity measure (`cos`), but also implemented another similarity measure `shared_overlap` designed to handle empty cells. Given two languages A and B, $shared\_overlap(A, B)$ is defined to be $\frac{\text{\# Of Features with Same Values}}{\text{\# Features Both Filled Out in WALS}}$. This measure can handle language pairs with many empty cells in WALS as it uses only features with cells

a is the number of language pairs found in the same set in both clusterings.
b is the number of language pairs found in different sets in C_1, and different sets in C_2.
c is the number of language pairs found in the same set in C_1, but in different sets in C_2.
d is the number of language pairs found in different sets in C_1, but the same set in C_2.

(a) Variables Used In Calculations

$$R(C_1, C_2) = \frac{a+b}{a+b+c+d}$$

(b) Rand Index

$$Precision(C_1, C_2) = \frac{a}{a+c}$$

(c) Cluster precision

$$Recall(C_1, C_2) = \frac{a}{a+d}$$

(d) Cluster recall

$$Fscore(C_1, C_2) = \frac{2 \cdot (Precision \cdot Recall)}{Precision + Recall}$$

(e) Cluster f-score

Figure 1: Formulas for calculating the Rand Index, cluster precision, recall, and f-score of two clusterings C_1 and C_2. C_1 is the system output, C_2 is the gold standard.

filled out for both languages, and calculates the percentage of features with the same values.

3.2 Clustering Performance Metrics

To measure clustering performance, we treat the genetic families specified in WALS as the gold standard, although we are not strictly aiming to recreate them.

Rand Index

The Rand Index (Rand, 1971) is one of the standard metrics for evaluating clustering results. It compares pairwise assignments of data points across two clusterings. For every pair of points there are four possibilities, as given in Figure 1. The Rand index is calculated by dividing the number of matching pairs ($a + b$) by the number of all pairs. This results in a number between 0 and 1 where 1 represents an identical clustering. Unfortunately, as noted by (Daumé and Marcu, 2005), the Rand Index tends to give disproportionately greater scores to clusterings with a greater number of clusters. For example, the Rand Index will always be 1.0 when each data point belongs to its own cluster. As a result, we have chosen to calculate metrics other than the Rand index: cluster precision, recall, and f-score.

Cluster Precision, Recall, and F-Score

Extending the notation in Figure 1, precision is defined as the proportion of same-set pairs in the target cluster C_1 that are correctly identified as being in the same set in the gold cluster C_2, while recall is the proportion of all same-set pairs in the gold cluster C_2 that are identified in the target cluster C_1. F-score is calculated as the usual harmonic mean of precision and recall. As it gives a more accurate representation of cluster similarity across varying amounts of clusters, we will report cluster similarity using cluster F-score.

3.3 Prediction Accuracy

Our second experiment was to answer the question posed in Q2: how do induced clusters and genetic families compare in predicting the values of features for languages in the same group?

To answer this question, we measure the accuracy of the prediction when both types of groups are used to predict the values of “empty” cells. We used 90% of the filled cells to build clusters, and then predicted the values of the remaining 10% of filled cells. The missing cells are filled with the value that occurs the most times among languages in the same group. If there are no other languages in the cluster, or the other languages have no values for this feature, then the cell is filled with the most common values for that feature across all languages in the dataset. Finally, the accuracy is calculated by comparing these predicted values with the actual values in the gold standard. We run 10-fold cross validation and report the average accuracy.

In addition to the prediction accuracy for each method of producing groupings, we calculate the baseline result where an empty cell is filled with the most frequent value for that feature across all the languages in the training data.

3.4 Determining Feature Stability

Finally, we look to answer Q3: what typological features tend to stay the same within related families? To find an answer, we look again to prediction accuracy. While prediction accuracy can be averaged across all features, it can also be broken down feature-by-feature to rank features according to how accurately they can be predicted by language families. Features that can be predicted with high accuracy implies that these features are more likely to remain *stable* within a language family than others.

Using prediction accuracies based on the genetic families, we rank features according to their accuracy and then perform clustering using the top features to determine if the cluster similarity to the genetic groups increases when using only the stable features.

4 Results & Analysis

4.1 Cluster Similarity

The graph in Figure 2(a) shows f-scores of clustering methods with the whole-world set. None achieve an f-score greater than 0.15, and most perform even worse when the number of clusters matches the number of genetic families or subfamilies. This indicates that the induced clusters based on typological features are very different from genetic groupings.

The question of similarity between these induced clusters and the genetic families is however a separate one from how those clusters perform in predicting typological feature values.

4.2 Prediction Accuracy

To determine the amount of similarity between languages within clusters, we instead look at prediction accuracy across clustering methods and the genetic groups. These scores are similar to those given in Daumé (2009), though not directly comparable due to small discrepancies in the size of the data set. As can be seen by the numbers in Table 3 and the graph in 2(b), despite the lack of similarity between clustering methods and the genetic groups, the clustering methods produce as good or better prediction accuracies. Furthermore, the `agglo` and `bagglo` hierarchical clustering methods which are favored for producing phylogenetically motivated clusters do indeed result in higher f-score similarity to the genetic clusters than the partitional `rb` and `direct` methods, but produce poorer prediction-accuracy results.

In fact, it is not surprising that some induced clusters outperform the genetic groupings in prediction accuracy, considering that clustering algorithms often want to maximize the similarity between languages in the same clusters. Now that we know similarity between languages does not necessarily mirror language family membership, the next question is what features tend to stay the same among languages in the same language families.

4.3 Feature Selection

Our final experiment was to examine the features in WALS themselves, and look for features that appear to vary the least within families, and act as better predictors of family membership.

In order to do this, we again looked at prediction accuracy information on a feature-by-feature basis. The results from this experiment are shown in Table 4, which gives a breakdown of how features rank both individually and by category.

Since this table is built upon genetic relationships, it is not surprising that the category for “Lexicon” appears to be the most reliably stable category. As noted in (McMahon, 1994), lexical cognates are often used as good evidence for determining a shared ancestry. We also find that word order is rather stable within a family.

We ran one further experiment where, using the `agglo` clustering method that provided clusters most similar to the genetic families previously, only features that showed accuracies above 50%. This eliminated 28 features, leaving 111 higher-scoring features for the whole-world set. Pruning the features to use only these selected for their stability within the genetic groupings yielded a very small increase in f-score similarity, as can be seen in Figure 3. Although this increase is small, it suggests that more advanced feature selection methods may be able to reveal language features that are more resistant to language contact and language change.

5 Error Analysis

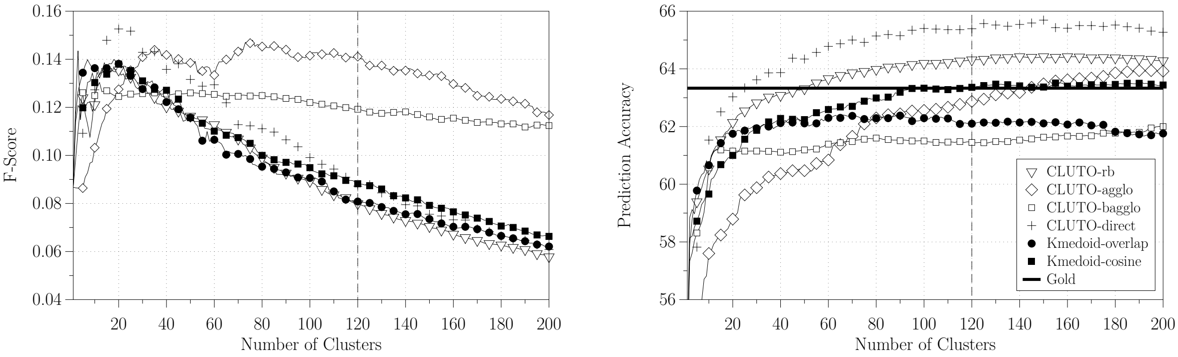
There are two main reasons for the differences between induced clusters and genetic groupings.

5.1 Language Similarity vs. Genetic Relatedness

As mentioned before, language similarity and genetic relatedness are two different concepts. Simi-

	baseline	gold	rb	agglo	bagglo	direct	k-medoids with similarity overlap	k-medoids with cosine similarity
Whole-World-Set (121 Clusters)								
F-Score	0.087	–	0.080	**0.140**	0.119	0.089	0.081	0.088
Acc (%)	53.72	63.43	64.33	62.86	61.44	**65.47**	62.11	63.36
Indo-European Subset (10 Clusters)								
F-Score	0.319	–	0.365	0.377	**0.391**	0.355	0.352	0.331
Acc (%)	64.27	74.1	71.12	72.26	70.62	**74.13**	73.36	72.12
Sino-Tibetan Subset (14 Clusters)								
F-Score	0.305	–	0.224	**0.340**	0.333	0.220	0.285	0.251
Acc (%)	58.08	61.71	63.93	63.74	63.06	**65.31**	64.55	63.94

Table 3: Comparison of clustering algorithms when the number of clusters is set to the same number of genetic groupings. The highest number in each row is in boldface.



(a) F-scores of clustering results

(b) Prediction accuracy

Figure 2: Comparison of the performances of different clustering methods using the whole-world data set. The number of groups in the gold standard (i.e., genetic grouping) is shown as a vertical dashed line in 2(a) and 2(b), and the prediction accuracy of the gold standard as a horizontal solid line in 2(b).

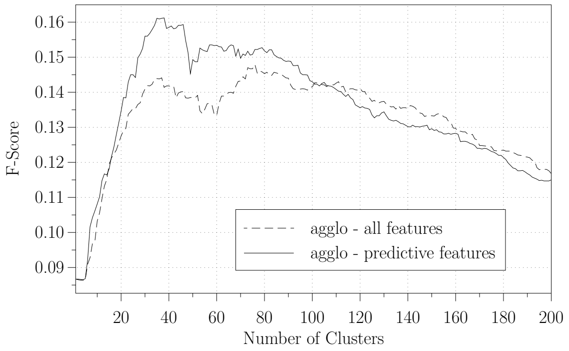


Figure 3: F-scores of the agglo clustering method when using all the features vs. only features whose prediction accuracy by the genetic grouping is higher than 50%.

lar languages might not be genetically related and dissimilar languages might be genetically related. An example is given in Table 5. Persian and English are both Indo-European languages, but look very different typologically; in contrast, Finnish and English are not genetically related but they look more similar typologically. While English and Persian are related, they have been diverging in geographically distant areas for thousands of years. Thus, the fact that English appears to share more features with a geographically closer Finnish is expected.

5.2 WALS as the Dataset

Perhaps the biggest challenge we encounter in this project has been the dataset itself. WALS has certain properties that complicate the task.

Data Sparsity and Shared Features

While the previous example shows unrelated languages can be quite similar typologically, our clustering methods put two closely related languages, Eastern and Western Armenian, into dif-

Breakdown by Feature Category		**Breakdown By Feature: Top 10**				**Breakdown by Feature: Bottom 10**			
Category	Accuracy	Feature	Acc	C	V	Feature	Acc	C	V
Whole-World Set									
Lexicon	75.0%	(136) M-T Pronouns	94.0%	230	3	(1) Consonant Inventories	32.6%	561	5
Word Order	68.6%	(18) Absence of Common Consonants	93.7%	565	6	(133) Number of Basic Color Categories	33.3%	119	7
Phonology	65.9%	(11) Front Rounded Vowels	91.1%	560	4	(23) Locus of Marking in the Clause	33.9%	236	5
Complex Sentences	64.0%	(73) The Optative	89.6%	319	2	(71) The Prohibitive	34.6%	495	4
Nominal Syntax	63.2%	(137) N-M Pronouns	87.9%	230	3	(22) Inflectional Synthesis of the Verb	35.1%	145	7
Verbal Categories	61.9%	(6) Uvular Consonants	85.0%	565	4	(56) Conjunctions and Universal Quantifiers	38.2%	116	3
Simple Clauses	60.5%	(130) Finger and Hand	84.4%	591	2	(117) Predicative Possession	39.4%	240	5
Nominal Categories	59.1%	(115) Negative Indefinite Pronouns	84.2%	206	4	(92) Position of Polar Question Particles	40.0%	775	6
Morphology	53.9%	(19) Presence of Uncommon Consonants	83.0%	565	7	(38) Indefinite Articles	40.4%	473	5
Other	41.3%	(58) Obligatory Possessive Inflection	81.4%	244	2	(50) Asymmetrical Case-Marking	40.7%	261	6
Indo-European Subset									
Lexicon	86.4%	(130) Finger and Hand	100.0%	35	2	(3) Consonant-Vowel Ratio	30.6%	31	5
Morphology	83.1%	(118) Predicative Adjectives	100.0%	29	3	(92) Position of Polar Question Particles	34.6%	47	6
Word Order	79.6%	(18) Absence of Common Consonants	100.0%	31	6	(78) Coding of Evidentiality	36.0%	23	6
Simple Clauses	76.6%	(107) Passive Constructions	100.0%	19	2	(1) Consonant Inventories	42.4%	31	5
Nominal Categories	70.4%	(88) Order of Demonstrative and Noun	97.2%	66	6	(2) Vowel Quality Inventories	44.4%	31	3
Phonology	66.7%	(89) Order of Numeral and Noun	95.7%	64	4	(84) Order of Object, Oblique, and Verb	47.8%	20	6
Verbal Categories	62.1%	(27) Reduplication	95.2%	20	3	(16) Weight Factors in Weight-Sensitive Stress Systems	51.1%	53	7
		(7) Glottalized Consonants	93.9%	31	8	(70) The Morphological Imperative	55.3%	53	5
		(93) Position of Interrogative Phrases in Content Questions	93.9%	44	3	(44) Gender Distinctions in Independent Personal Pronouns	56.5%	19	6
		(5) Voicing and Gaps in Plosive Systems	93.8%	31	5	(37) Definite Articles	59.2%	46	5
Sino-Tibetan Subset									
Lexicon	100.0%	(130) Finger and Hand	100.0%	8	2	(77) Semantic Distinctions of Evidentiality	9.1%	18	3
Word Order	67.7%	(82) Order of Subject and Verb	100.0%	99	3	(78) Coding of Evidentiality	17.7%	18	6
Morphology	63.8%	(119) Nominal and Locational Predication	100.0%	13	2	(4) Voicing in Plosives and Fricatives	20.7%	26	4
Simple Clauses	60.9%	(86) Order of Genitive and Noun	100.0%	73	3	(1) Consonant Inventories	22.2%	26	5
Verbal Categories	60.7%	(129) Hand and Arm	100.0%	8	2	(14) Fixed Stress Locations	25.0%	4	7
Nominal Categories	55.8%	(18) Absence of Common Consonants	100.0%	26	6	(15) Weight-Sensitive Stress	25.0%	4	8
Phonology	50.7%	(93) Pos. of Interr. Phrases in Content Q's	100.0%	79	3	(38) Indefinite Articles	31.7%	36	5
		(85) Order of Adposition and Noun Phrase	97.5%	79	5	(120) Zero Copula for Predicate Nominals	37.5%	13	2
		(95) Relationship b/t Object and Verb and Adposition and Noun Phrase	96.3%	76	5	(2) Vowel Quality Inventories	42.9%	26	3
		(48) Person Marking on Adpositions	93.3%	14	4	(3) Consonant-Vowel Ratio	46.7%	26	5

Table 4: Prediction accuracy figures derived from genetic groupings for each dataset and broken down by WALS feature category and feature. Ordering is by descending accuracy for the top 10 features, and by increasing accuracy for the bottom 10 features. The 'C' and 'V' columns give the number of languages in the set that a feature appears in, and the number of possible values for that feature, respectively.

ferent clusters. A quick review shows that the reason for this mistake is due to a lack of shared features in WALS. Table 6 shows that very few features are specified for both languages. The data sparsity problem and the distribution of empty cells adversely affect clustering results.

Notice that in this example, the features whose values are filled for both languages actually have identical feature values. While using shared overlap as a similarity measure can capture the similarity between these two languages, this measure biases clustering toward features with fewer cells filled out. The only way out of errors like this, it seems, is to obtain more data.

There are a few other typological databases that might be drawn upon to define a more complete set of data: PHOIBLE, (Moran and Wright, 2009), ODIN (Lewis, 2006), and the AUTOTYP database (Nichols and Bickel, 2009). Using these databases to fill in the gaps in data may be the only way to fully address these issues.

The Feature Set in WALS

The features in WALS are not systematically chosen for full typological coverage; rather, the contributors to WALS decide what features they want to work on based on their expertise. Also, some features in WALS overlap; for example, one WALS feature looks at the order between subject, verb, and object, and another feature checks the order between verb and object. As a result, the feature set in WALS might not be a good representative of the properties of the languages covered in the database.

6 Conclusion & Further Work

By comparing clusters derived from typological features to genetic groups in the world's languages, we have found two interesting results. First, the induced clusters look very different from genetic grouping and this is partly due to the design of WALS. Second, despite the differences, induced clusters show similar, or even greater levels

ID: Feature Name	English	Finnish	Persian
2: Vowel Quality Inventories	**Large (7-14)**	**Large (7-14)**	Average (5-6)
6: Uvular Consonants	**None**	**None**	Uvular stops only
11: Front Rounded Vowels	**None**	High and Mid	**None**
27: Reduplication	**No productive reduplication**	**No productive reduplication**	Productive full and partial reduplication
37: Definite Articles	Definite word distinct from demonstrative	No definite or indefinite article	No definite, but indefinite article
53: Ordinal Numerals	**First, second, three-th**	**First, second, three-th**	First/one-th, two-th, three-th
81: Order of Subject, Object and Verb	**SVO**	**SVO**	SOV
85: Order of Adposition and Noun Phrase	**Prepositions**	Postpositions	**Prepositions**
87: Order of Adjective and Noun	**Adjective-Noun**	**Adjective-Noun**	Noun-Adjective
124: 'Want' Complement Subjects	**Subject left implicit**	**Subject left implicit**	Subject expressed overtly
Number of Features	139	135	128
Cosine Similarity to Eng	1.00	0.56	0.42
Shared Overlap with Eng	1.00	0.56	0.44

Table 5: A selection of ten features from English, Finnish, and Persian. Same feature values in each row are in boldface. Despite the genetic relation between English and Persian, similarity metrics place English closer to Finnish than Persian.

ID#	Feature Name	Armenian (Eastern)	Armenian (Western)
1	Consonant Inventories	Small	–
27	Reduplication	**Full Reduplication Only**	**Full Reduplication Only**
33	Coding of Nominal Plurality	–	Plural suffix
48	Person Marking on Adj.	None	–
81	Order of Subj. Obj., and V	–	SOV
86	Order of Adposition and Noun Phrase	**Postpositions**	**Postpositions**
100	Alignment of Verbal Person Marking	Accusative	–
129	Hand and Arm	–	Identical
	Number of Features	85	33
	Cosine Similarity	0.22	
	Shared Overlap	1.00	

Table 6: Comparison of features between Eastern and Western Armenian. Same feature values in each row are in boldface. Empty cells are shown as '–'.

of typological similarity than genetic grouping as indicated by the prediction accuracy.

While these initial findings are interesting, using WALS as a dataset for this purpose leaves a lot to be desired. Subsequent work that supplements the typological data in WALS with the databases mentioned in §5.2 would help alleviate the data sparsity and feature selection problems.

Another useful follow-up would be to perform application-oriented evaluations. For instance, evaluating the performance of syntactic projection methods between languages determined to have similar syntactic patterns, or using similar morphological induction techniques on morphologically similar languages. With the development of large typological databases such as WALS, we hope to see more studies that take advantage of resources for resource-rich languages when developing tools for typologically similar, but resource-poor languages.

Acknowledgment This work is supported by the National Science Foundation Grant BCS-0748919. We would also like to thank Emily Bender, Tim Baldwin, and three anonymous reviewers for helpful comments.

References

Daumé, III, Hal and Daniel Marcu. 2005. A Bayesian Model for Supervised Clustering with the Dirichlet Process Prior. *Journal of Machine Learning Research*, 6:1551–1577.

Daumé, III, Hal. 2009. Non-Parametric Bayesian Areal Linguistics. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL)*, pages 593–601, Boulder, Colorado, June.

Estivill-Castro, Vladimir and Jianhua Yang. 2000. A fast and robust general purpose clustering algorithm. In *Proc. of Pacific Rim International Conference on Artificial Intelligence*, pages 208–218. Springer.

Hana, Jiri, Anna Feldman, and Chris Brew. 2004. A Resource-light Approach to Russian Morphology: Tagging Russian using Czech resources. In *Proceedings of EMNLP 2004*, Barcelona, Spain.

Haspelmath, Martin, Matthew S. Dryer, David Gil, and Bernard Comrie. 2005. *The World Atlas of Language Structures*. Oxford University Press, Oxford, England.

Lewis, William D. 2006. ODIN: A Model for Adapting and Enriching Legacy Infrastructure. In *Proceedings of the e-Humanities Workshop, held in cooperation with e-Science 2006: 2nd IEEE International Conference on e-Science and Grid Computing*, Amsterdam.

McMahon, April M. S. 1994. *Understanding language change*. Cambridge University Press, Cambridge; New York, NY, USA.

Moran, Steven and Richard Wright. 2009. Phonetics Information Base and Lexicon (PHOIBLE). Online: `http://phoible.org`.

Nichols, Johanna and Balthasar Bickel. 2009. The AUTOTYP genealogy and geography database: 2009 release. `http://www.uni-leipzig.de/~autotyp`.

Rand, William M. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Steinbach, Michael, George Karypis, and Vipin Kumar. 2000. A comparison of document clustering techniques. In *Proceedings of Workshop at KDD 2000 on Text Mining*.

Xia, Fei and William D. Lewis. 2007. Multilingual structural projection across interlinear text. In *Proc. of the Conference on Human Language Technologies (HLT/NAACL 2007)*, pages 452–459, Rochester, New York.

Yarowsky, David and Grace Ngai. 2001. Inducing multilingual pos taggers and np bracketers via robust projection across aligned corpora. In *Proc. of the Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies (NAACL-2001)*, pages 1–8, Morristown, NJ, USA.

Better Arabic Parsing: Baselines, Evaluations, and Analysis

Spence Green and Christopher D. Manning
Computer Science Department, Stanford University
{spenceg,manning}@stanford.edu

Abstract

In this paper, we offer broad insight into the underperformance of Arabic constituency parsing by analyzing the interplay of linguistic phenomena, annotation choices, and model design. First, we identify sources of syntactic ambiguity understudied in the existing parsing literature. Second, we show that although the Penn Arabic Treebank is similar to other treebanks in gross statistical terms, annotation consistency remains problematic. Third, we develop a human interpretable grammar that is competitive with a latent variable PCFG. Fourth, we show how to build better models for three different parsers. Finally, we show that in application settings, the absence of gold segmentation lowers parsing performance by 2–5% F1.

1 Introduction

It is well-known that constituency parsing models designed for English often do not generalize easily to other languages and treebanks.[1] Explanations for this phenomenon have included the relative informativeness of lexicalization (Dubey and Keller, 2003; Arun and Keller, 2005), insensitivity to morphology (Cowan and Collins, 2005; Tsarfaty and Sima'an, 2008), and the effect of variable word order (Collins et al., 1999). Certainly these linguistic factors increase the difficulty of syntactic disambiguation. Less frequently studied is the interplay among language, annotation choices, and parsing model design (Levy and Manning, 2003; Kübler, 2005).

To investigate the influence of these factors, we analyze Modern Standard Arabic (henceforth MSA, or simply "Arabic") because of the unusual opportunity it presents for comparison to English parsing results. The Penn Arabic Treebank (ATB) syntactic guidelines (Maamouri et al., 2004) were purposefully borrowed without major modification from English (Marcus et al., 1993). Further, Maamouri and Bies (2004) argued that the English guidelines generalize well to other languages. But Arabic contains a variety of linguistic phenomena unseen in English. Crucially, the conventional orthographic form of MSA text is *unvocalized*, a property that results in a deficient graphical representation. For humans, this characteristic can impede the acquisition of literacy. How do additional ambiguities caused by devocalization affect statistical learning? How should the absence of vowels and syntactic markers influence annotation choices and grammar development? Motivated by these questions, we significantly raise baselines for three existing parsing models through better grammar engineering.

Our analysis begins with a description of syntactic ambiguity in unvocalized MSA text (§2). Next we show that the ATB is similar to other treebanks in gross statistical terms, but that annotation consistency remains low relative to English (§3). We then use linguistic and annotation insights to develop a manually annotated grammar for Arabic (§4). To facilitate comparison with previous work, we exhaustively evaluate this grammar and two other parsing models when gold segmentation is assumed (§5). Finally, we provide a realistic evaluation in which segmentation is performed both in a pipeline and jointly with parsing (§6). We quantify error categories in both evaluation settings. To our knowledge, ours is the first analysis of this kind for Arabic parsing.

[1] The apparent difficulty of adapting constituency models to non-configurational languages has been one motivation for dependency representations (Hajič and Zemánek, 2004; Habash and Roth, 2009).

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 394–402,
Beijing, August 2010

2 Syntactic Ambiguity in Arabic

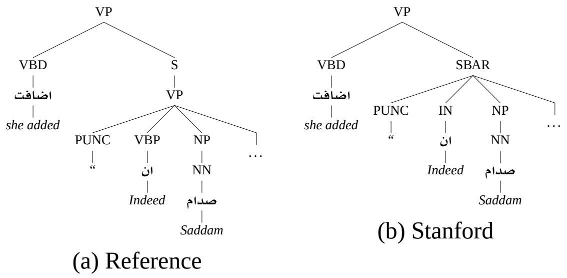
Arabic is a morphologically rich language with a root-and-pattern system similar to other Semitic languages. The basic word order is VSO, but SVO, VOS, and VO configurations are also possible.[2] Nouns and verbs are created by selecting a consonantal root (usually triliteral or quadriliteral), which bears the semantic core, and adding affixes and diacritics. Particles are uninflected. Diacritics can also be used to specify grammatical relations such as case and gender. But diacritics are not present in unvocalized text, which is the standard form of, e.g., news media documents.[3]

Let us consider an example of ambiguity caused by devocalization. Table 1 shows four words whose unvocalized surface forms ان *an* are indistinguishable. Whereas Arabic linguistic theory assigns (1) and (2) to the class of pseudo verbs إن وأخواتها *inna and her sisters* since they can be inflected, the ATB conventions treat (2) as a complementizer, which means that it must be the head of SBAR. Because these two words have identical complements, syntax rules are typically unhelpful for distinguishing between them. This is especially true in the case of quotations—which are common in the ATB—where (1) will follow a verb like (2) (Figure 1).

Even with vocalization, there are linguistic categories that are difficult to identify without semantic clues. Two common cases are the attributive adjective and the process nominal المصدار *maSdar*, which can have a verbal reading.[4] Attributive adjectives are hard because they are orthographically identical to nominals; they are inflected for gender, number, case, and definiteness. Moreover, they are used as substantives much more frequently than is done in English.

	Word	Head Of	Complement	POS
1	إنّ *inna* “Indeed, truly”	VP	Noun	VBP
2	أنّ *anna* “That”	SBAR	Noun	IN
3	إن *in* “If”	SBAR	Verb	IN
4	أن *an* “to”	SBAR	Verb	IN

Table 1: Diacritized particles and pseudo-verbs that, after orthographic normalization, have the equivalent surface form ان *an*. The distinctions in the ATB are linguistically justified, but complicate parsing. Table 8a shows that the best model recovers SBAR at only 71.0% F1.



(a) Reference

(b) Stanford

Figure 1: The Stanford parser (Klein and Manning, 2002) is unable to recover the verbal reading of the unvocalized surface form ان *an* (Table 1).

Process nominals name the action of the transitive or ditransitive verb from which they derive. The verbal reading arises when the *maSdar* has an NP argument which, in vocalized text, is marked in the accusative case. When the *maSdar* lacks a determiner, the constituent as a whole resembles the ubiquitous annexation construct الإضافة *iDafa*. Gabbard and Kulick (2008) show that there is significant attachment ambiguity associated with *iDafa*, which occurs in 84.3% of the trees in our development set. Figure 4 shows a constituent headed by a process nominal with an embedded adjective phrase. All three models evaluated in this paper incorrectly analyze the constituent as *iDafa*; none of the models attach the attributive adjectives properly.

For parsing, the most challenging form of ambiguity occurs at the discourse level. A defining characteristic of MSA is the prevalence of *discourse markers* to connect and subordinate words and phrases (Ryding, 2005). Instead of offsetting new topics with punctuation, writers of MSA insert connectives such as و *wa* and ف *fa* to link new elements to both preceding clauses and the text as a whole. As a result, Arabic sentences are usually long relative to English, especially after

[2] Unlike machine translation, constituency parsing is not significantly affected by variable word order. However, when grammatical relations like subject and object are evaluated, parsing performance drops considerably (Green et al., 2009). In particular, the decision to represent arguments in verb-initial clauses as VP internal makes VSO and VOS configurations difficult to distinguish. Topicalization of NP subjects in SVO configurations causes confusion with VO (pro-drop).

[3] Techniques for automatic vocalization have been studied (Zitouni et al., 2006; Habash and Rambow, 2007). However, the data sparsity induced by vocalization makes it difficult to train statistical models on corpora of the size of the ATB, so vocalizing and then parsing may well not help performance.

[4] Traditional Arabic linguistic theory treats both of these types as subcategories of noun الاسم.

Length	English (WSJ)	Arabic (ATB)
≤ 20	41.9%	33.7%
≤ 40	**92.4%**	73.2%
≤ 63	99.7%	**92.6%**
≤ 70	99.9%	94.9%

Table 2: Frequency distribution for sentence lengths in the WSJ (sections 2–23) and the ATB (p1–3). English parsing evaluations usually report results on sentences up to length 40. Arabic sentences of up to length 63 would need to be evaluated to account for the same fraction of the data. We propose a limit of 70 words for Arabic parsing evaluations.

	Part of Speech	Tag	Freq.
و *wa* "and"	conjunction	CC	4256
	preposition	IN	6
	abbreviation	NN	6
ف *fa* "so, then"	conjunction	CC	160
	connective particle	RP	67
	abbreviation	NN	22
	response conditioning particle	RP	11
	subordinating conjunction	IN	3

Table 3: Dev set frequencies for the two most significant discourse markers in Arabic are skewed toward analysis as a conjunction.

	ATB	CTB6	Negra	WSJ
Trees	23449	28278	20602	**43948**
Word Typess	40972	45245	51272	**46348**
Tokens	738654	782541	355096	**1046829**
Tags	32	34	**499**	45
Phrasal Cats	22	26	**325**	27
Test OOV	16.8%	22.2%	**30.5%**	13.2%
Per Sentence				
Depth (μ / σ^2)	3.87 / 0.74	**5.01 / 1.44**	3.58 / 0.89	4.18 / 0.74
Breadth (μ / σ^2)	**14.6 / 7.31**	10.2 / 4.44	7.50 / 4.56	12.1 / 4.65
Length (μ / σ^2)	**31.5 / 22.0**	27.7 / 18.9	17.2 / 10.9	23.8 / 11.2
Constituents (μ)	**32.8**	32.5	8.29	19.6
μ Const. / μ Length	1.04	**1.18**	0.482	0.820

Table 4: Gross statistics for several different treebanks. Test set OOV rate is computed using the following splits: ATB (Chiang et al., 2006); CTB6 (Huang and Harper, 2009); Negra (Dubey and Keller, 2003); English, sections 2-21 (train) and section 23 (test).

segmentation (Table 2). The ATB gives several different analyses to these words to indicate different types of coordination. But it conflates the coordinating and discourse separator functions of *wa* (واو العطف) into one analysis: conjunction (Table 3). A better approach would be to distinguish between these cases, possibly by drawing on the vast linguistic work on Arabic connectives (Al-Batal, 1990). We show that noun-noun vs. discourse-level coordination ambiguity in Arabic is a significant source of parsing errors (Table 8c).

3 Treebank Comparison

3.1 Gross Statistics

Linguistic intuitions like those in the previous section inform language-specific annotation choices. The resulting structural differences between treebanks can account for relative differences in parsing performance. We compared the ATB[5] to treebanks for Chinese (CTB6), German (Negra), and English (WSJ) (Table 4). The ATB is disadvantaged by having fewer trees with longer average yields.[6] But to its great advantage, it has a high ratio of non-terminals/terminals (μ Constituents / μ Length). Evalb, the standard parsing metric, is biased *toward* such corpora (Sampson and Babarczy, 2003). Also surprising is the low test set OOV rate given the possibility of morphological variation in Arabic. In general, several gross corpus statistics favor the ATB, so other factors must contribute to parsing underperformance.

3.2 Inter-annotator Agreement

Annotation consistency is important in any supervised learning task. In the initial release of the ATB, inter-annotator agreement was inferior to other LDC treebanks (Maamouri et al., 2008). To improve agreement during the revision process, a dual-blind evaluation was performed in which 10% of the data was annotated by independent teams. Maamouri et al. (2008) reported agreement between the teams (measured with Evalb) at 93.8% F1, the level of the CTB. But Rehbein and van Genabith (2007) showed that Evalb should not be used as an indication of real difference—or similarity—between treebanks.

Instead, we extend the *variation n-gram* method of Dickinson (2005) to compare annotation error rates in the WSJ and ATB. For a corpus C, let M be the set of tuples $\langle n, l \rangle$, where n is an n-gram with bracketing label l. If any n appears

[5] LDC A-E catalog numbers: LDC2008E61 (ATBp1v4), LDC2008E62 (ATBp2v3), and LDC2008E22 (ATBp3v3.1). We map the ATB morphological analyses to the shortened "Bies" tags for all experiments.

[6] Generative parsing performance is known to deteriorate with sentence length. As a result, Habash et al. (2006) developed a technique for splitting and chunking long sentences. In application settings, this may be a profitable strategy.

	Corpus		Sample	Error %	
	Trees	Nuclei	n-grams	Type	n-gram
WSJ 2–23	43948	25041	746	12.0%	**2.10%**
ATB	23449	20292	2100	**37.0%**	1.76%

Table 5: Evaluation of 100 randomly sampled variation nuclei types. The samples from each corpus were independently evaluated. The ATB has a much higher fraction of nuclei per tree, and a higher type-level error rate.

in a corpus position without a bracketing label, then we also add $\langle n, \mathtt{NIL} \rangle$ to M. We call the set of unique n-grams with multiple labels in M the *variation nuclei* of C.

Bracketing variation can result from either annotation errors or linguistic ambiguity. Human evaluation is one way to distinguish between the two cases. Following Dickinson (2005), we randomly sampled 100 variation nuclei from each corpus and evaluated each sample for the presence of an annotation error. The human evaluators were a non-native, fluent Arabic speaker (the first author) for the ATB and a native English speaker for the WSJ.[7]

Table 5 shows type- and token-level error rates for each corpus. The 95% confidence intervals for type-level errors are (5580, 9440) for the ATB and (1400, 4610) for the WSJ. The results clearly indicate increased variation in the ATB relative to the WSJ, but care should be taken in assessing the magnitude of the difference. On the one hand, the type-level error rate is not calibrated for the number of n-grams in the sample. At the same time, the n-gram error rate is sensitive to samples with extreme n-gram counts. For example, one of the ATB samples was the determiner ذلك *dhalik* "that." The sample occurred in 1507 corpus positions, and we found that the annotations were consistent. If we remove this sample from the evaluation, then the ATB type-level error rises to only 37.4% while the n-gram error rate increases to 6.24%. The number of ATB n-grams also falls below the WSJ sample size as the largest WSJ sample appeared in only 162 corpus positions.

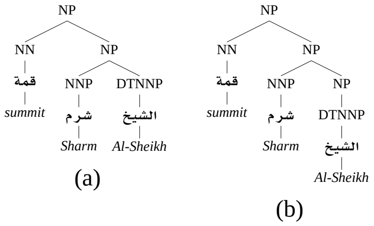


Figure 2: An ATB sample from the human evaluation. The ATB annotation guidelines specify that proper nouns should be specified with a flat NP (a). But the city name *Sharm Al-Sheikh* is also *iDafa*, hence the possibility for the incorrect annotation in (b).

4 Grammar Development

We can use the preceding linguistic and annotation insights to build a manually annotated Arabic grammar in the manner of Klein and Manning (2003). Manual annotation results in human interpretable grammars that can inform future treebank annotation decisions. A simple lexicalized PCFG with second order Markovization gives relatively poor performance: 75.95% F1 on the test set.[8] But this figure is surprisingly competitive with a recent state-of-the-art baseline (Table 7).

In our grammar, features are realized as annotations to basic category labels. We start with noun features since written Arabic contains a very high proportion of NPs. **genitiveMark** indicates recursive NPs with a indefinite nominal left daughter and an NP right daughter. This is the form of recursive levels in *iDafa* constructs. We also add an annotation for one-level *iDafa* (**oneLevelIdafa**) constructs since they make up more than 75% of the *iDafa* NPs in the ATB (Gabbard and Kulick, 2008). For all other recursive NPs, we add a common annotation to the POS tag of the head (**recursiveNPHead**).

Base NPs are the other significant category of nominal phrases. **markBaseNP** indicates these non-recursive nominal phrases. This feature includes named entities, which the ATB marks with a flat NP node dominating an arbitrary number of NNP pre-terminal daughters (Figure 2).

For verbs we add two features. First we mark any node that dominates (at any level) a verb

[7] Unlike Dickinson (2005), we strip traces and only consider POS tags when pre-terminals are the only intervening nodes between the nucleus and its bracketing (e.g., unaries, base NPs). Since our objective is to compare distributions of bracketing discrepancies, we do not use heuristics to prune the set of nuclei.

[8] We use head-finding rules specified by a native speaker of Arabic. This PCFG is incorporated into the Stanford Parser, a factored model that chooses a 1-best parse from the product of constituency and dependency parses.

Feature	States	Tags	F1	Indiv. ΔF1
—	3208	33	76.86	—
recursiveNPHead	3287	38	77.46	+0.60
genitiveMark	3471	38	77.88	+0.42
splitPUNC	4221	47	77.98	+0.10
markContainsVerb	5766	47	79.16	+1.18
markBaseNP	6586	47	79.5	+0.34
markOneLevelIdafa	7202	47	79.83	+0.33
splitIN	7595	94	80.48	+0.65
containsSVO	9188	94	80.66	+0.18
splitCC	9492	124	80.87	+0.21
markFem	10049	141	80.95	+0.08

Table 6: Incremental dev set results for the manually annotated grammar (sentences of length $\leq$ 70).

phrase (**markContainsVerb**). This feature has a linguistic justification. Historically, Arabic grammar has identified two sentences types: those that begin with a nominal (الجملة الاسمية), and those that begin with a verb (الجملة الفعلية). But foreign learners are often surprised by the verbless predications that are frequently used in Arabic. Although these are technically nominal, they have become known as "equational" sentences. **markContainsVerb** is especially effective for distinguishing root S nodes of equational sentences. We also mark all nodes that dominate an SVO configuration (**containsSVO**). In MSA, SVO usually appears in non-matrix clauses.

Lexicalizing several POS tags improves performance. **splitIN** captures the verb/preposition idioms that are widespread in Arabic. Although this feature helps, we encounter one consequence of variable word order. Unlike the WSJ corpus which has a high frequency of rules like VP → VB PP, Arabic verb phrases usually have lexicalized intervening nodes (e.g., NP subjects and direct objects). For example, we might have VP → VB NP PP, where the NP is the subject. This annotation choice weakens **splitIN**.

The ATB gives all punctuation a single tag. For parsing, this is a mistake, especially in the case of interrogatives. **splitPUNC** restores the convention of the WSJ. We also mark all tags that dominate a word with the feminine ending ة *taa marbuuTa* (**markFeminine**).

To differentiate between the coordinating and discourse separator functions of conjunctions (Table 3), we mark each CC with the label of its right sister (**splitCC**). The intuition here is that the role of a discourse marker can usually be determined by the category of the word that follows it. Because conjunctions are elevated in the parse trees when they separate recursive constituents, we choose the right sister instead of the category of the next word. We create equivalence classes for verb, noun, and adjective POS categories.

5 Standard Parsing Experiments

We compare the manually annotated grammar, which we incorporate into the Stanford parser, to both the Berkeley (Petrov et al., 2006) and Bikel (Bikel, 2004) parsers. All experiments use ATB parts 1–3 divided according to the canonical split suggested by Chiang et al. (2006). Preprocessing the raw trees improves parsing performance considerably.[9] We first discard all trees dominated by X, which indicates errors and non-linguistic text. At the phrasal level, we remove all function tags and traces. We also collapse unary chains with identical basic categories like NP → NP. The preterminal morphological analyses are mapped to the shortened "Bies" tags provided with the treebank. Finally, we add "DT" to the tags for definite nouns and adjectives (Kulick et al., 2006).

The orthographic normalization strategy we use is simple.[10] In addition to removing all diacritics, we strip instances of *taTweel* تطويل, collapse variants of *alif* ا to bare *alif*,[11] and map Arabic punctuation characters to their Latin equivalents. We retain segmentation markers—which are consistent only in the vocalized section of the treebank—to differentiate between e.g. هم "they" and هم+ "their." Because we use the vocalized section, we must remove null pronoun markers.

In Table 7 we give results for several evaluation metrics. Evalb is a Java re-implementation of the standard labeled precision/recall metric.[12]

[9] Both the corpus split and pre-processing code are available at http://nlp.stanford.edu/projects/arabic.shtml.

[10] Other orthographic normalization schemes have been suggested for Arabic (Habash and Sadat, 2006), but we observe negligible parsing performance differences between these and the simple scheme used in this evaluation.

[11] *taTweel* (ـ) is an elongation character used in Arabic script to justify text. It has no syntactic function. Variants of *alif* are inconsistently used in Arabic texts. For *alif* with *hamza*, normalization can be seen as another level of devocalization.

[12] For English, our Evalb implementation is identical to the most recent reference (EVALB20080701). For Arabic we

Model	System	Length	Leaf Ancestor			Evalb			Tag
			Corpus	Sent	Exact	LP	LR	F1	%
Stanford (v1.6.3)	Baseline	70	0.791	0.825	**358**	80.37	79.36	79.86	95.58
		all	0.773	0.818	358	78.92	77.72	78.32	95.49
	GoldPOS	70	0.802	0.836	452	81.07	80.27	80.67	99.95
Bikel (v1.2)	Baseline (Self-tag)	70	0.770	0.801	278	77.92	76.00	76.95	94.64
		all	0.752	0.794	278	76.96	75.01	75.97	94.63
	Baseline (Pre-tag)	70	0.771	0.804	295	78.35	76.72	77.52	**95.68**
		all	0.752	0.796	295	77.31	75.64	76.47	95.68
	GoldPOS	70	0.775	0.808	309	78.83	77.18	77.99	96.60
Berkeley (Sep. 09)	(Petrov, 2009)	all	—	—	—	76.40	75.30	75.85	—
	Baseline	70	**0.809**	**0.839**	335	**82.32**	**81.63**	**81.97**	95.07
		all	0.796	0.834	336	81.43	80.73	81.08	95.02
	GoldPOS	70	0.831	0.859	496	84.37	84.21	84.29	99.87

Table 7: Test set results. Maamouri et al. (2009b) evaluated the Bikel parser using the same ATB split, but only reported dev set results with gold POS tags for sentences of length ≤ 40. The Bikel GoldPOS configuration only supplies the gold POS tags; it does not force the parser to use them. We are unaware of prior results for the Stanford parser.

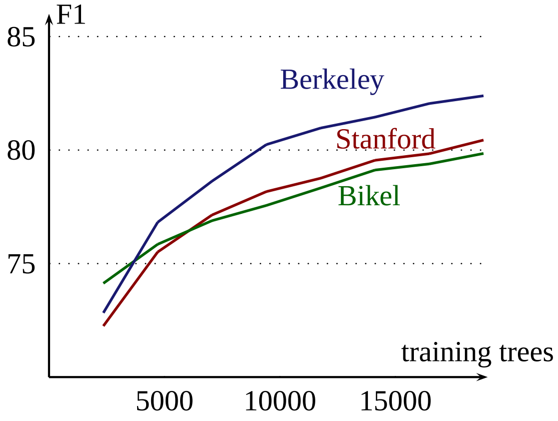


Figure 3: Dev set learning curves for sentence lengths ≤ 70. All three curves remain steep at the maximum training set size of 18818 trees.

The Leaf Ancestor metric measures the cost of transforming guess trees to the reference (Sampson and Babarczy, 2003). It was developed in response to the non-terminal/terminal bias of Evalb, but Clegg and Shepherd (2005) showed that it is also a valuable diagnostic tool for trees with complex deep structures such as those found in the ATB. For each terminal, the Leaf Ancestor metric extracts the shortest path to the root. It then computes a normalized Levenshtein edit distance between the extracted chain and the reference. The range of the score is between 0 and 1 (higher is better). We report micro-averaged (whole corpus) and macro-averaged (per sentence) scores along with the number of exactly matching guess trees.

5.1 Parsing Models

The Stanford parser includes both the manually annotated grammar (§4) and an Arabic unknown word model with the following lexical features:

1. Presence of the determiner ال *Al*
2. Contains digits
3. Ends with the feminine affix ة *p*
4. Various verbal (e.g., وا, ت) and adjectival suffixes (e.g., ية)

Other notable parameters are second order vertical Markovization and marking of unary rules.

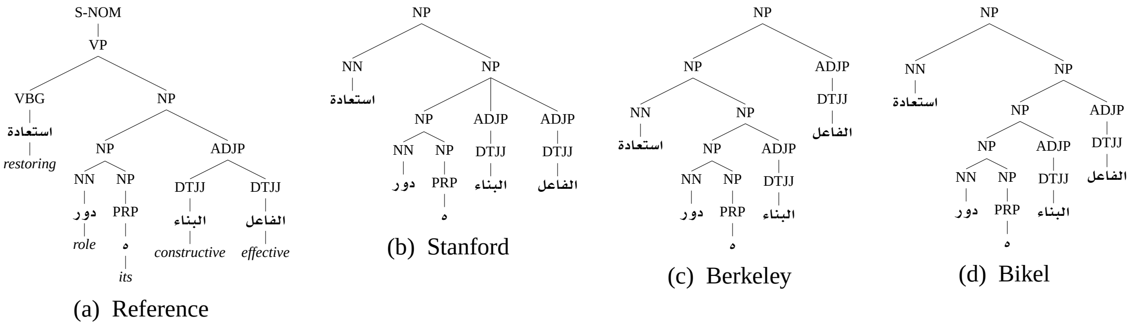
Modifying the Berkeley parser for Arabic is straightforward. After adding a ROOT node to all trees, we train a grammar using six split-and-merge cycles and no Markovization. We use the default inference parameters.

Because the Bikel parser has been parameterized for Arabic by the LDC, we do not change the default model settings. However, when we pre-tag the input—as is recommended for English—we notice a 0.57% F1 improvement. We use the log-linear tagger of Toutanova et al. (2003), which gives 96.8% accuracy on the test set.

5.2 Discussion

The Berkeley parser gives state-of-the-art performance for all metrics. Our baseline for all sentence lengths is 5.23% F1 higher than the best previous result. The difference is due to more careful

add a constraint on the removal of punctuation, which has a single tag (PUNC) in the ATB. Tokens tagged as PUNC are not discarded unless they consist entirely of punctuation.



(a) Reference

(b) Stanford

(c) Berkeley

(d) Bikel

Figure 4: The constituent *Restoring of its constructive and effective role* parsed by the three different models (gold segmentation). The ATB annotation distinguishes between verbal and nominal readings of *maSdar* process nominals. Like verbs, *maSdar* takes arguments and assigns case to its objects, whereas it also demonstrates nominal characteristics by, e.g., taking determiners and heading *iDafa* (Fassi Fehri, 1993). In the ATB, استعادة *asta'adah* is tagged 48 times as a noun and 9 times as verbal noun. Consequently, all three parsers prefer the nominal reading. Table 8b shows that verbal nouns are the hardest pre-terminal categories to identify. None of the models attach the attributive adjectives correctly.

pre-processing. However, the learning curves in Figure 3 show that the Berkeley parser does not exceed our manual grammar by as wide a margin as has been shown for other languages (Petrov, 2009). Moreover, the Stanford parser achieves the most exact Leaf Ancestor matches and tagging accuracy that is only 0.1% below the Bikel model, which uses pre-tagged input.

In Figure 4 we show an example of variation between the parsing models. We include a list of per-category results for selected phrasal labels, POS tags, and dependencies in Table 8. The errors shown are from the Berkeley parser output, but they are representative of the other two parsing models.

6 Joint Segmentation and Parsing

Although the segmentation requirements for Arabic are not as extreme as those for Chinese, Arabic is written with certain cliticized prepositions, pronouns, and connectives connected to adjacent words. Since these are distinct syntactic units, they are typically segmented. The ATB segmentation scheme is one of many alternatives. Until now, all evaluations of Arabic parsing—including the experiments in the previous section—have assumed gold segmentation. But gold segmentation is not available in application settings, so a segmenter and parser are arranged in a pipeline. Segmentation errors cascade into the parsing phase, placing an artificial limit on parsing performance.

Lattice parsing (Chappelier et al., 1999) is an alternative to a pipeline that prevents cascading errors by placing all segmentation options into the parse chart. Recently, lattices have been used successfully in the parsing of Hebrew (Tsarfaty, 2006; Cohen and Smith, 2007), a Semitic language with similar properties to Arabic. We extend the Stanford parser to accept pre-generated lattices, where each word is represented as a finite state automaton. To combat the proliferation of parsing edges, we prune the lattices according to a hand-constructed lexicon of 31 clitics listed in the ATB annotation guidelines (Maamouri et al., 2009a). Formally, for a lexicon L and segments $I \in L$, $O \notin L$, each word automaton accepts the language $I^*(O+I)I^*$. Aside from adding a simple rule to correct *alif* deletion caused by the preposition ل, no other language-specific processing is performed.

Our evaluation includes both weighted and unweighted lattices. We weight edges using a unigram language model estimated with Good-Turing smoothing. Despite their simplicity, unigram weights have been shown as an effective feature in segmentation models (Dyer, 2009).[13] The joint parser/segmenter is compared to a pipeline that uses MADA (v3.0), a state-of-the-art Arabic segmenter, configured to replicate ATB segmentation (Habash and Rambow, 2005). MADA uses an ensemble of SVMs to first re-rank the output of a deterministic morphological analyzer. For each

[13] Of course, this weighting makes the PCFG an improper distribution. However, in practice, unknown word models also make the distribution improper.

Label	# gold	F1
ADJP	**1216**	**59.45**
SBAR	**2918**	**69.81**
FRAG	254	72.87
VP	5507	78.83
S	6579	78.91
PP	7516	80.93
NP	34025	84.95
ADVP	1093	90.64
WHNP	787	96.00

(a) Major phrasal categories

Tag	# gold	%	Tag	# gold	%
VBG	**182**	**48.84**	**JJR**	**134**	**92.83**
VN	**163**	**60.37**	DTNNS	1069	94.29
VBN	352	72.42	**DTJJ**	**3361**	**95.07**
DTNNP	932	83.48	NNP	4152	95.09
JJ	**1516**	**86.09**	NN	10336	95.23
ADJ_NUM	**277**	**88.93**	DTNN	6736	95.78
VBP	2139	89.94	NOUN_QUANT	352	98.16
RP	818	91.23	PRP	1366	98.24
NNS	907	91.75	CC	4076	98.92
DTJJR	**78**	**92.41**	IN	8676	99.07
VBD	2580	92.42	DT	525	99.81

(b) Major POS categories

Parent	Head	Modifer	Dir	# gold	F1
NP	NP	TAG	R	946	0.54
S	**S**	**S**	**R**	**708**	**0.57**
NP	**NP**	**ADJP**	**R**	**803**	**0.64**
NP	**NP**	**NP**	**R**	**2907**	**0.66**
NP	NP	SBAR	R	1035	0.67
NP	**NP**	**PP**	**R**	**2713**	**0.67**
VP	TAG	PP	R	3230	0.80
NP	NP	TAG	L	805	0.85
VP	TAG	SBAR	R	772	0.86
S	VP	NP	L	961	0.87

(c) Ten lowest scoring (Collins, 2003)-style dependencies occurring more than 700 times

Table 8: Per category performance of the Berkeley parser on sentence lengths ≤ 70 (dev set, gold segmentation). **(a)** Of the high frequency phrasal categories, ADJP and SBAR are the hardest to parse. We showed in §2 that lexical ambiguity explains the underperformance of these categories. **(b)** POS tagging accuracy is lowest for *maSdar* verbal nouns (VBG,VN) and adjectives (e.g., JJ). Richer tag sets have been suggested for modeling morphologically complex distinctions (Diab, 2007), but we find that linguistically rich tag sets do not help parsing. **(c)** Coordination ambiguity is shown in dependency scores by e.g., ⟨S S S R⟩ and ⟨NP NP NP R⟩. ⟨NP NP PP R⟩ and ⟨NP NP ADJP R⟩ are both *iDafa* attachment.

input token, the segmentation is then performed deterministically given the 1-best analysis.

Since guess and gold trees may now have different yields, the question of evaluation is complex. Cohen and Smith (2007) chose a metric like SParseval (Roark et al., 2006) that first aligns the trees and then penalizes segmentation errors with an edit-distance metric. But we follow the more direct adaptation of Evalb suggested by Tsarfaty (2006), who viewed exact segmentation as the ultimate goal. Therefore, we only score guess/gold pairs with identical *character* yields, a condition that allows us to measure parsing, tagging, and segmentation accuracy by ignoring whitespace.

Table 9 shows that MADA produces a high quality segmentation, and that the effect of cascading segmentation errors on parsing is only 1.92% F1. However, MADA is language-specific and relies on manually constructed dictionaries. Conversely, the lattice parser requires no linguistic resources and produces segmentations of comparable quality. Nonetheless, parse quality is much lower in the joint model because a lattice is effectively a long sentence. A cell in the bottom row of the parse chart is required for each potential whitespace boundary. As we have said, parse quality decreases with sentence length. Finally, we note that simple weighting gives nearly a 2% F1 improvement, whereas Goldberg and Tsarfaty (2008) found that unweighted lattices were more effective for Hebrew.

	LP	LR	F1	Seg F1	Tag F1	Coverage
STANFORD (Gold)	81.64	80.55	81.09	100.0	95.81	100.0%
MADA	—	—	—	97.67	—	96.42%
MADA+STANFORD	**79.44**	**78.90**	**79.17**	**97.67**	**94.27**	**96.42%**
STANFORDJOINT	76.13	72.61	74.33	94.12	90.13	94.73%
STANFORDJOINT+UNI	77.09	74.97	76.01	96.26	92.23	95.87%

Table 9: Dev set results for sentences of length ≤ 70. Coverage indicates the fraction of hypotheses in which the character yield exactly matched the reference. Each model was able to produce hypotheses for all input sentences. In these experiments, the input lacks segmentation markers, hence the slightly different dev set baseline than in Table 6.

7 Conclusion

By establishing significantly higher parsing baselines, we have shown that Arabic parsing performance is not as poor as previously thought, but remains much lower than English. We have described grammar state splits that significantly improve parsing performance, catalogued parsing errors, and quantified the effect of segmentation errors. With a human evaluation we also showed that ATB inter-annotator agreement remains low relative to the WSJ corpus. Our results suggest that current parsing models would benefit from better annotation consistency and enriched annotation in certain syntactic configurations.

Acknowledgments We thank Steven Bethard, Evan Rosen, and Karen Shiells for material contributions to this work. We are also grateful to Markus Dickinson, Ali Farghaly, Nizar Habash, Seth Kulick, David McCloskey, Claude Reichard, Ryan Roth, and Reut Tsarfaty for constructive discussions. The first author is supported by a National Defense Science and Engineering Graduate (NDSEG) fellowship. This paper is based on work supported in part by DARPA through IBM. The content does not necessarily reflect the views of the U.S. Government, and no official endorsement should be inferred.

References

Al-Batal, M. 1990. Connectives as cohesive elements in a modern expository Arabic text. In Eid, Mushira and John McCarthy, editors, *Perspectives on Arabic Linguistics II*. John Benjamins.

Arun, A and F Keller. 2005. Lexicalization in crosslinguistic probabilistic parsing: The case of French. In *ACL*.

Bikel, D M. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30:479–511.

Chappelier, J-C, M Rajman, R Arages, and A Rozenknop. 1999. Lattice parsing for speech recognition. In *TALN*.

Chiang, D, M Diab, N Habash, O Rambow, and S Shareef. 2006. Parsing Arabic dialects. In *EACL*.

Clegg, A and A Shepherd. 2005. Evaluating and integrating treebank parsers on a biomedical corpus. In *ACL Workshop on Software*.

Cohen, S and N A Smith. 2007. Joint morphological and syntactic disambiguation. In *EMNLP*.

Collins, M, J Hajic, L Ramshaw, and C Tillmann. 1999. A statistical parser for Czech. In *ACL*.

Collins, M. 2003. Head-Driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Cowan, B and M Collins. 2005. Morphology and reranking for the statistical parsing of Spanish. In *NAACL*.

Diab, M. 2007. Towards an optimal POS tag set for Modern Standard Arabic processing. In *RANLP*.

Dickinson, M. 2005. *Error Detection and Correction in Annotated Corpora*. Ph.D. thesis, The Ohio State University.

Dubey, A and F Keller. 2003. Probabilistic parsing for German using sister-head dependencies. In *ACL*.

Dyer, C. 2009. Using a maximum entropy model to build segmentation lattices for MT. In *NAACL*.

Fassi Fehri, A. 1993. *Issues in the structure of Arabic clauses and words*. Kluwer Academic Publishers.

Gabbard, R and S Kulick. 2008. Construct state modification in the Arabic treebank. In *ACL*.

Goldberg, Y and R Tsarfaty. 2008. A single generative model for joint morphological segmentation and syntactic parsing. In *ACL*.

Green, S, C Sathi, and C D Manning. 2009. NP subject detection in verb-initial Arabic clauses. In *Proc. of the Third Workshop on Computational Approaches to Arabic Script-based Languages (CAASL3)*.

Habash, N and O Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *ACL*.

Habash, N and O Rambow. 2007. Arabic diacritization through full morphological tagging. In *NAACL*.

Habash, N and R Roth. 2009. CATiB: The Columbia Arabic Treebank. In *ACL, Short Papers*.

Habash, N and F Sadat. 2006. Arabic preprocessing schemes for statistical machine translation. In *NAACL*.

Habash, N, B Dorr, and C Monz. 2006. Challenges in building an Arabic-English GHMT system with SMT components. In *EAMT*.

Hajič, J and P Zemánek. 2004. Prague Arabic dependency treebank: Development in data and tools. In *NEMLAR*.

Huang, Z and M Harper. 2009. Self-training PCFG grammars with latent annotations across languages. In *EMNLP*.

Klein, D and C D Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *NIPS*.

Klein, D and C D Manning. 2003. Accurate unlexicalized parsing. In *ACL*.

Kulick, S, R Gabbard, and M Marcus. 2006. Parsing the Arabic Treebank: Analysis and improvements. In *TLT*.

Kübler, S. 2005. How do treebank annotation schemes influence parsing results? Or how not to compare apples and oranges. In *RANLP*.

Levy, R and C D Manning. 2003. Is it harder to parse Chinese, or the Chinese treebank? In *ACL*.

Maamouri, M and A Bies. 2004. Developing an Arabic Treebank: Methods, guidelines, procedures, and tools. In *Proc. of the Workshop on Computational Approaches to Arabic Script-based Languages (CAASL1)*.

Maamouri, M, A Bies, T Buckwalter, and W Mekki. 2004. The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus. In *NEMLAR*.

Maamouri, M, A Bies, and S Kulick. 2008. Enhancing the Arabic Treebank: A collaborative effort toward new annotation guidelines. In *LREC*.

Maamouri, M, A Bies, S Krouna, F Gaddeche, and B Bouziri. 2009a. Penn Arabic Treebank guidelines v4.92. Technical report, Linguistic Data Consortium, University of Pennsylvania, August 5.

Maamouri, M, A Bies, and S Kulick. 2009b. Creating a methodology for large-scale correction of treebank annotation: The case of the Arabic Treebank. In *MEDAR*.

Marcus, M, M A Marcinkiewicz, and B Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

Petrov, S, L Barrett, R Thibaux, and D Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL*.

Petrov, S. 2009. *Coarse-to-Fine Natural Language Processing*. Ph.D. thesis, University of California-Berkeley.

Rehbein, I and J van Genabith. 2007. Treebank annotation schemes and parser evaluation for German. In *EMNLP-CoNLL*.

Roark, B, M Harper, E Charniak, B Dorr, M Johnson, J G Kahne, Y Liuf, Mari Ostendorf, J Hale, A Krasnyanskaya, M Lease, I Shafran, M Snover, R Stewart, and L Yung. 2006. SParseval: Evaluation metrics for parsing speech. In *LREC*.

Ryding, K. 2005. *A Reference Grammar of Modern Standard Arabic*. Cambridge University Press.

Sampson, G and A Babarczy. 2003. A test of the leaf-ancestor metric for parse accuracy. *Natural Language Engineering*, 9:365–380.

Toutanova, K, D Klein, C D Manning, and Y Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL*.

Tsarfaty, R and K Sima'an. 2008. Relational-realizational parsing. In *COLING*.

Tsarfaty, R. 2006. Integrated morphological and syntactic disambiguation for Modern Hebrew. In *ACL*.

Zitouni, I, J S Sorensen, and R Sarikaya. 2006. Maximum entropy based restoration of Arabic diacritics. In *ACL*.

Paraphrase Alignment for Synonym Evidence Discovery

Gintarė Grigonytė
Saarland University
g.grigonyte@hmf.vdu.lt

João Cordeiro, Gaël Dias, Rumen Moraliyski
HULTIG
University of Beira Interior
{jpaulo,ddg,rumen}@di.ubi.pt

Pavel Brazdil
LIAAD/FEP
University of Porto
pbrazdil@liaad.up.pt

Abstract

We describe a new unsupervised approach for synonymy discovery by aligning paraphrases in monolingual domain corpora. For that purpose, we identify phrasal terms that convey most of the concepts within domains and adapt a methodology for the automatic extraction and alignment of paraphrases to identify *paraphrase casts* from which valid synonyms are discovered. Results performed on two different domain corpora show that general synonyms as well as synonymic expressions can be identified with a 67.27% precision.

1 Introduction

Synonymy is a specific type of a semantic relationship. According to (Sowa and Siekmann, 1994), a synonym is a word (or concept) that means the same or nearly the same as another word (or concept). It has been observed that words are similar if their contexts are similar (Freitag et al., 2005) and so synonymy detection has received a lot of attention during the last decades. However, words used in the same context are not necessarily synonyms and can embody different semantic relationships such as hyponyms, meronyms or co-hyponyms (Heylen et al., 2008). In this paper, we introduce a new unsupervised methodology for synonym detection by extracting and aligning paraphrases on normalized domain corpora[1]. In particular, we study a specific structure within aligned paraphrases, *paraphrase casts*, from which valid synonyms are discovered. In fact, we propose a new approach based on the idea that synonyms are substitutable words within a given domain corpus. Results performed on two different domain corpora, the Corpus of Computer Security (COCS) and the Corpus of Cancer Research (COCR), show that general synonyms as well as synonymic expressions can be identified with a 67.27% precision performance.

2 Related Work

Automatic synonymy detection has been tackled in a variety of ways which we explain as follows.

2.1 Pattern-based Approaches

This approach to information extraction is based on a technique called *selective concept extraction* as defined by (Riloff, 1993). Selective concept extraction is a form of text skimming that selectively processes relevant text while effectively ignoring surrounding text that is thought to be irrelevant to the domain. The pioneer of pattern-based approaches (Hearst, 1992) has introduced lexico-syntactic patterns to automatically acquire given word semantic relationships. Specific patterns like ”*X and other Y*” or ”*X such as Y*” were used for hypernym-hyponym detection. Later, the idea was extended and adapted for synonymy by other researchers such as (Roark and Charniak, 1998), (Caraballo, 1999) and (Maynard and Peters, 2009). In general, manual pattern definition is time consuming and requires linguistic skills. Usually, systems based on lexico-syntactic patterns perform with very high precision, but low recall due to the fact that these patterns are rare. However, recent work by (Ohshima and Tanaka,

[1] By normalized, we intend that phrasal terms have been previously identified.

2009) on Web data reported high recall figures. To avoid manual encoding of patterns, many supervised approaches have been proposed as summarized in (Stevenson and Greenwood, 2006).

2.2 Distributional Similarity

Distributional similarity for capturing semantic relatedness is relying on the hypothesis that semantically similar words share similar contexts. These methods vary in the level of supervision from unsupervised to semi-supervised or to supervised. The first type of methods includes the work of (Hindle, 1990), (Lin, 1998) and (Heylen et al., 2008) who used unsupervised methods for detecting word similarities based on shallow-parsed corpora. Others have proposed unsupervised methodologies to solve TOEFL-like tests, instead of discovering synonyms (Turney, 2001), (Terra and Clarke, 2003) and (Freitag et al., 2005). Other researchers, such as (Girju et al., 2004), (Muller et al., 2006), (Wu and Zhou, 2003) and (Wei et al., 2009), have used language or knowledge resources to enhance the representation of the vector space model. Unlike the pattern-based approach, the distributional similarity-based approach shows low precision compared to high recall.

One obvious way to verify all the possible connections between words of the vocabulary employs an exhaustive search. However, comparison based on word usage can only highlight those terms that are highly similar in meaning. This method of representation is usually unable to distinguish between middle strength and weak semantic relations, or detect the relationship between hapax-legomena.

2.3 Hybrid Approaches

More recently, approaches combining patterns and distributional similarity appeared to bring the best of the two methodologies. (Hagiwara et al., 2009) describe experiments that involve training various synonym classifiers. (Giovannetti et al., 2008) use syntactically parsed text and manually composed patterns together with distributional similarity for detecting semantically related words. Finally, (Turney, 2008) proposes a supervised machine learning approach for discovering synonyms, antonyms, analogies and associations. For that purpose, feature vectors are based on frequencies of patterns and classified by a SVM.

2.4 Our Approach

(Van der Plas and Tiedemann, 2006) state that ”*People use multiple ways to express the same idea. These alternative ways of conveying the same information in different ways are referred to by the term* paraphrase *and in the case of single words or phrasal terms sharing the same meaning, we speak of synonyms*”. Based on this, we propose that in order to discover pairs of semantically related words (in the best case synonyms) that may be used in figurative or rare sense, and as consequence impossible to be identified by the distributional similarity approach, we need to have them highlighted by their **local specific** environment. Here we differ from the pattern-based approach that use **local general** environment. We propose to align paraphrases from domain corpora and discover words that are possibly substitutable for one another in a given context (*paraphrase casts*), and as such are synonyms or near-synonyms. Comparatively to existing approaches, we propose an unsupervised and language-independent methodology which does not depend on linguistic processing[2], nor manual definition of patterns or training sets and leads to higher precision when compared to distributional similarity-based approaches.

3 Normalization of the Corpora

The main goal of our research is to build knowledge resources in different domains that can effectively be used in different NLP applications. As such, precision takes an important part in the overall process of our methodology. For that purpose, we first identify the phrasal terms (or multi-word units) present in the corpora. Indeed, it has been shown in many works that phrasal terms convey most of the specific contents of a given domain. Our approach to term extraction is based on linguistic pattern matching and Inverse Document Frequency (IDF) measurements for term

[2] We will see in the next section that we will use linguistic resources to normalize our corpora, but the methodology can be applied to any raw text.

quality assurance as explained in (Avizienis et al., 2009). For that purpose, we present a domain independent hybrid term extraction framework that includes the following steps. First, the text is morphologically annotated with the MPRO system (Maas et al., 2009). Then grammar rules for morphological disambiguation, syntactic parsing and noun phrase detection are applied based on finite-state automata technology, KURD (Carl and Schmidt-Wigger, 1998). Following this, a variant and non-basic term form detection is applied, as well as stop words removal. Then, combining rich morphological and shallow syntactical analysis with pattern matching techniques allows us to extract a wide span of candidate terms which are finally filtered with the well-known IDF measure.

4 Paraphrase Identification

A few unsupervised metrics have been applied to automatic paraphrase identification and extraction (Barzilay and McKeown, 2001) and (Dolan et al., 2004). However, these unsupervised methodologies show a major drawback by extracting quasi-exact or even exact match pairs of sentences as they rely on classical string similarity measures. Such pairs are useless for our research purpose. More recently, (Cordeiro et al., 2007a) proposed the *sumo* metric specially designed for asymmetrical entailed pair identification in corpora which obtained better performance than previously established metrics, even in corpora with exclusively symmetrical entailed paraphrases as in the Microsoft Paraphrase Research Corpus (Dolan et al., 2004). This function states that for a given sentence pair $\langle S_a, S_b \rangle$, having m and n words in each sentence and λ lexical exclusive links (word overlaps, see figure 1) between them, its lexical connection strength is computed as defined in Equations 1 and 2.

$$Sumo(S_a, S_b) = \begin{cases} S(m,n,\lambda) & if \;\; S(m,n,\lambda) < 1 \\ 0 & if \;\; \lambda = 0 \\ e^{-kS(m,n,\lambda)} & otherwise \end{cases} \tag{1}$$

where

$$\begin{aligned} & S(m,n,\lambda) = \alpha \log_2(\tfrac{m}{\lambda}) + \beta \log_2(\tfrac{n}{\lambda}) \\ & \alpha, \beta \in [0,1], \alpha + \beta = 1 \end{aligned} \tag{2}$$

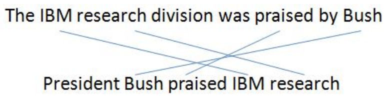

The IBM research division was praised by Bush
President Bush praised IBM research

Figure 1: 4 exclusive links between S_a and S_b.

To obtain a paraphrase corpus, we compute all sentence pairs similarities $Sumo(S_a, S_b)$ and select only those pairs exceeding a given threshold, in our case $threshold = 0.85$, which is quite restrictive, ensuring the selection of pairs strongly connected[3].

However, to take into account the normalization of the corpus, little adjustments had to be integrated in the methodology proposed in (Cordeiro et al., 2007a). Indeed, the original $Sumo(.,.)$ function was under-weighting links that occurred between phrasal terms such as "*molecular laboratory*" or "*renal cancer*". So, instead of counting the number of lexical links among sentences, each link weights differently according to the word length in the connection, hence connections of longer words will result in a larger value. For example, in figure 1, instead of having $\lambda = 4$, we count $\lambda = 3 + 8 + 7 + 4 = 22$. This adjustment is important as multi-word units are treated as longer words in the corpus. This modification has also, as a side effect, under-evaluation of functional words which usually follow the Zipf's Law and give more importance to meaningful words in the paraphrase extraction process.

5 Paraphrase Alignment

In order to usefully explore the evidence synonymy from paraphrases, sentence alignment techniques must be applied to paraphrases in order to identify *paraphrase casts*, i.e., substitutable word pairs as shown in figure 2. As we can see, the paraphrase cast includes three parts: the left segment (context), a middle segment and the right segment (context). In our figure the left and right segments (contexts) are identical.

In the context of DNA sequence alignment, two main algorithms have been proposed: (1) the Needleman-Wunsch algorithm (Needleman and

[3] Further details about the sumo metric are available in (Cordeiro et al., 2007a).

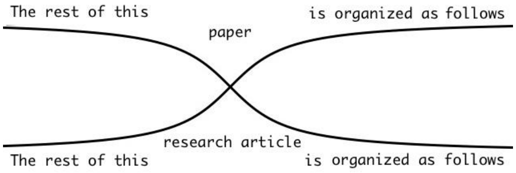


Figure 2: A paraphrase cast.

Wunsch, 1970) based on dynamic programming which outputs a unique global alignment and (2) the Smith-Waterman (SW) algorithm (Smith and Waterman, 1981) which is an adaptation of the previous algorithm and outputs local alignments. In the context of NLP, (Cordeiro et al., 2007a) proposed a combination of both algorithms depending on the structure of paraphrase. However, since any local alignment is a candidate for *paraphrase casts*, the SW algorithm revealed itself more appropriate and was always chosen. The SW alignment algorithm uses dynamic programming to compute the optimal local alignments between two sequences[4]. This process requires first the definition of an alignment matrix (function), which governs the likelihood of alignment of two symbols. Thus we first build a matrix H such that $H(i, 0) = 0$ and $H(0, j) = 0$, for $0 \leq i \leq m$, and $0 \leq j \leq n$, where m and n are the number of words in the paraphrase sentences. The rest of the H elements are recursively calculated as in Equation 3 where $gs(.,.)$ is the gap-scoring function and S_{a_i} (resp. S_{b_j}) represents the i^{th} (resp. j^{th}) word of sentence S_a (resp. S_b).

$$H(i,j) = max \begin{cases} 0 & \\ H(i-1, j-1) + gs(S_{a_i}, S_{b_j}) & \text{MMisatch} \\ H(i-1, j) + gs(S_{a_i}, \_) & \text{Deletion} \\ H(i, j-1) + gs(\_, S_{b_j}) & \text{Insertion} \end{cases} \tag{3}$$

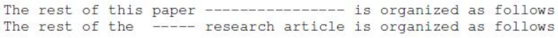
Obviously, this algorithm is based on an alignment function which exploits the alignment likelihood between two alphabet symbols. For DNA sequence alignments, this function is defined as a mutation matrix, scoring gene mutation and gap alignments. In our case, we define the gap-scoring function $gs(.,.)$ in Equations 4 and 5 which prioritize the alignment of specific domain key terms i.e., single match, or key expressions i.e., phrasal match, (reward 50), as well as lexically similar[5] words such as "*programme*" and "*programming*" for example. In particular, for these similar words an adaptation of the well known *Edit Distance* is used, the $c(.,.)$ function (5), which is explained in more detail in (Cordeiro et al., 2007b).

$$gs(S_{a_i}, S_{b_j}) = \begin{cases} -1 & \text{if } (S_{a_i} = -) \text{ and } (S_{b_j} \neq -) \\ -1 & \text{if } (S_{b_j} = -) \text{ and } (S_{a_i} \neq -) \\ 10 & \text{Single Match} \\ 50 & \text{Phrasal Match} \\ c(S_{a_i}, S_{b_j}) & \text{Mismatch} \end{cases} \tag{4}$$

where

$$c(S_{a_i}, S_{b_j}) = -\frac{edist(S_{a_i}, S_{b_j})}{\epsilon + maxseq(S_{a_i}, S_{b_j})} \tag{5}$$

To obtain local alignments, the SW algorithm is applied, using the alignment function defined with $H(.,.)$ in 3. The alignment of the paraphrase in figure 2 would give the result in figure 3.



```
The rest of this paper ---------------- is organized as follows
The rest of the  ----- research article is organized as follows
```

Figure 3: An alignment.

6 Paraphrase Casts

In order to discover synonyms, we are looking for special patterns from the aligned paraphrase sentences, which naturally give us more evidence for the existence of equivalent terms or expressions. Due to the topological aspect of such patterns, we decided to name them *paraphrase casts*, or just *casts* as shown in figure 2. As we have mentioned earlier, the paraphrase cast includes three parts: the left segment (*contextL*), a middle segment and the right segment (*contextR*). In the following example the left and right segments (contexts) are identical, but the middle segment includes **different** misaligned sequences of words, represented by *wordSa* and *wordSb*.

```
contextL wordSa ----- contextR
contextL ----- wordSb contextR
```

[4] In our case, the two sequences are the two sentences of a paraphrase

[5] This is why we have in equation 3 the label "Mismatch", where "mismatch" means different yet lexically near words.

We can attribute different levels of confidence to different paraphrase casts. Indeed, the larger the contexts and the smaller the misaligned sequences are, the more likely it is for single or phrasal terms to be synonyms or near-synonyms. Note that in the cast shown in figure 3, each context has a significant size, with four words on each side, and the misaligned segments are in fact equivalent expressions i.e. ”*paper*” is a synonym of ”*research article*”. In the analyzed domain these expressions are equivalent and interchangeable and appear to be interchangeable in other domains. For the purpose of this paper, we only take into account the casts where the misaligned sequences of words contain only one word or one multi-word unit in each sentence. That is, we have a one-to-one match. However, no constraints are imposed on the contexts[6]. So, all casts are computed and analyzed for synonym discovery and results are provided in the next section.

7 Experiments

To evaluate our methodology we have used two different corpora, both from scientific domains built from abstracts of publications (see Table 1). The corpus of computer security (COCS) is a collection of 4854 abstracts on computer security extracted from the IEEE (http://ieee.rkbexplorer.com/) repository[7]. The corpus of cancer research (COCR) contains 3334 domain specific abstracts of scientific publications extracted from the PubMed[8] on three types of cancer: (1) the mammary carcinoma register (COCR1) consisting of 1500 abstracts, (2) the nephroblastoma register (COCR2) consisting of 1500 abstracts, and (3) the rhabdoid tumor register (COCR3) consisting of 334 abstracts. From the paraphrase casts, we were able to automatically extract, without further processing, single synonymous word pairs, as well as synonymic multi-word units, as can be seen in Table 2. For that purpose we have used specific paraphrase casts, whose aim was to privilege precision to recall. In particular, we have removed all casts which contained numbers or special characters i.e. casts with filter in Table 1. However, no constraints were imposed on the frequency of the casts. Indeed, all casts were included even if their overall frequency was just one. Although most of the word relationships were concerned with synonymy, the other ones were not just errors, but lexically related words, namely examples of antonymy, hyperonymy/hyponymy and associations as shown in Table 3. In the evaluation, we have focused on the precision of the method. The evaluation of the extracted pairs was performed manually by two domain experts. In fact, four

Corpus	COCS	COCR1	COCR2	COCR3
Tokens	412.265	336.745	227.477	46.215
Sentences	18.974	15.195	10.575	2.321
Aligned Pairs	589	994	511	125
Casts without filter	320	10.217	2.520	48
Casts with filter	202	361	292	16

Table 1: Corpora

Synonym (COCS)	**Complementary**
frequency tuning	frequency control
attack consequences	attack impact
error-free operation	error free operation
pseudo code	pseudo algorithm
tolerance	resilience
package loss	message loss
adjustable algorithm	context-aware algorithm
helpful comment	valuable comment
Synonym (COCR)	**Complementary**
childhood renal tumor	childhood kidney tumor
hypertrophy	growth
doxorubicin	vincristine
carcinomas of the kidney	sarcoma of the kidney
metastasis	neoplasm
renal tumor	renal malignancy
neoplastic thrombus	tumor thrombus
vincristine	adriamycin

Table 2: Synonyms for COCS

Antonym	**Complementary**
positive sentinel nodes	negative sentinel nodes
higher bits	lower bits
older version	newer version
Hypernym	**Hyponym**
Multi-Tasking Virtual Machine	Java Virtual Machine
therapy	chemotherapy
hormone breast cancer	estrogen breast cancer
Association	**Complementary**
performance	reliability
statistical difference	significant difference
relationship	correlation

Table 3: Other Word Semantic Relationships.

[6] This issue will be discussed in the next section.

[7] An example of an abstract can be viewed at: http://ieee.rkbexplorer.com/description/publication-00534618

[8] http://www.ncbi.nlm.nih.gov/pubmed

different evaluations were carried out depending on whether the adapted $S(.,.)$ measure was used (or not) and whether the normalization of the corpora was used (or not). The best results were obtained in all cases for the adapted $S(.,.)$ measure with the multi-word units. Table 4 shows also the worst results for the COCS as a baseline (COCS (1)), i.e. non-adapted $S(.,.)$ and non-normalized corpus. For the rest of the experiments we always present the results with the adapted $S(.,.)$ measure and normalized corpus.

Corpus	**COCS (1)**	**COCS (2)**	
Precision	54.62%	71.29%	
Extracted Synonyms	130	144	
Errors	108	58	
Corpus	**COCR1**	**COCR2**	**COCR3**
Precision	69.80%	61.30%	75.00%
Extracted Synonyms	252	178	12
Errors	109	111	4

Table 4: Overall Results

7.1 Discussion

Many results have been published in the literature, especially tackling the TOEFL synonym detection problem which aims at identifying the correct synonym among a small set of alternatives (usually four). For that purpose, the best precision rate has been reached by (Turney et al., 2003) with 97.50% who have exploited many resources, both statistical and linguistic. However, our methodology tackles **a different problem**. Indeed, our goal is to automatically extract synonyms from texts. The published works toward this direction have not reached so good results. One of the latest studies was conducted by (Heylen et al., 2008) who used distributional similarity measures to extract synonyms from shallow parsed corpora with the help of unsupervised methods. They report that ”*the dependency-based model finds a tightly related neighbor for 50% of the target words and a true synonym for 14%*”. So, it means that by comparing all words in a corpus with all other words, one can expect to find a correct semantic relationship in 50% of the cases and a correct synonym in just 14%. In that perspective, our approach reaches higher results. Moreover, (Heylen et al., 2008) use a frequency cut-off which only selects features that occur at least five times together with the target word. In our case, no frequency threshold is imposed to enable extraction of synonyms with low frequency, such as *hapax legomena*. This situation is illustrated in figure 4. We note that most of the candidate pairs appear only once in the corpus.

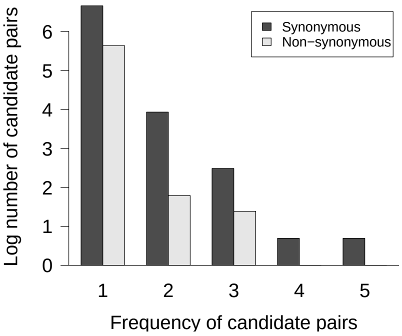


Figure 4: Synonyms Frequency Distribution.

In order to assess the quality of our results, we calculated the similarity between all extracted pairs of synonyms following the distributional analysis paradigm as in (Moraliyski and Dias, 2007) who build context[9] feature vectors for noun synonyms. In particular, we used the cosine similarity measure and the Loglike association measure (Dunning, 1993) as the weighting scheme of the context features. The distribution of the similarity measure for all noun synonyms (62 pairs) is shown in figure 5.

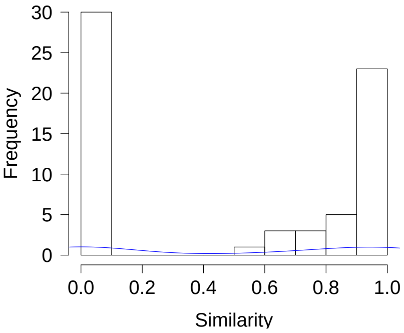


Figure 5: Synonym Pairs Similarity Distribution.

The results clearly show that all extracted synonyms are highly correlated in terms of context.

[9] In this case, the contexts are the surrounding nouns, verbs and adjectives in the closest chunks of a shallow parsed corpus.

Nearly half of the cases have similarities higher than 0.5. It is important to notice that a specific corpus[10] was built to calculate as sharply as possible the similarity measures as it is done in (Moraliyski and Dias, 2007). Indeed, based on the COCS and the COCR most statistics were insignificant leading to zero-valued features. This situation is well-known as it is one of the major drawbacks of the distributional analysis approach which needs huge quantities of texts to make secure decisions. So we note that applying the distributional analysis approach to such small corpora would have led to rather poor results. Even with an adapted corpus, figure 5 (left-most bar) shows that there are no sufficient statistics for 30 pairs of synonyms. Although the quality of the extracted pairs of synonyms is high, the precision remains relatively low with 67.27% precision on average. As we pointed out in the previous section, we did not make any restrictions to the left and right contexts of the casts. However, the longer these contexts are, compared to the misaligned sequence of words, the higher the chance is that we find a correct synonym. Table 5 shows the average lengths of both cast contexts for synonyms and erroneous pairings, in terms of words (WCL) and characters (CCL). We also provide the ratio (R) between the character lengths of the middle segment (i.e. misaligned character sequences) and the character lengths of the cast contexts (i.e. right and left sizes of equally aligned character sequences). It is clear that a more thorough study of the effects of the left and right contexts should be carried out to improve precision of our approach, although this may be to the detriment of recall. Based on the results of the ratio R[11], we note that the larger the cast context is compared to the cast content, the more likely it is that the misaligned words are synonyms.

Type	WCL	CCL	R
Synonyms	2.70	11.67	0.70
Errors	2.45	8.05	0.55

Table 5: Average Casts Contexts Lengths

8 Conclusions

In this paper we have introduced a new unsupervised methodology for synonym detection that involves extracting and aligning paraphrases on normalized domain corpora. In particular, we have studied a specific structure within aligned paraphrases, *paraphrase casts*, from which valid synonyms were discovered. The overall precision was 71.29% for the computer security domain and 66.06% for the cancer research domain. This approach proved to be promising for extracting synonymous words and synonymic multi-word units. Its strength is the ability to effectively work with small domain corpora, without supervised training, nor definition of specific language-dependent patterns. Moreover, it is capable to extract synonymous pairs with figurative or rare senses which would be impossible to identify using the distributional similarity approach. Finally, our approach is completely language-independent as it can be applied to any raw text, not obligatorily normalized corpora, although the results for domain terminology may be improved by the identification of phrasal terms.

However, further improvements of the method should be considered. A measure of quality of the *paraphrase casts* is necessary to provide a measure of confidence to the kind of extracted word semantic relationship. Indeed, the larger the contexts and the smaller the misaligned sequences are, the more likely it is for single or phrasal terms to be synonyms or near-synonyms. Further work should also be carried out to differentiate the acquired types of semantically related pairs. As it is shown in Table 3, some of the extracted pairs were not synonymic, but lexically related words such as antonyms, hypernyms/hyponyms and associations. A natural follow-up solution for discriminating between semantic types of extracted pairs could involve context-based classification of acquired *casts* pairs. In particular, (Turney, 2008) tackled the problem of classifying different lexical information such as synonymy, antonymy, hypernymy and association by using context words. In order to propose a completely unsupervised methodology, we could also follow the idea of (Dias et al., 2010) to automatically construct small

[10] This corpus contains 125.888.439 words.

[11] These results are statistically relevant with $p - value < 0.001$ using the Wilcoxon Rank-Sum Test.

TOEFL-like tests based on sets of *casts* which could be handled with the help of different distributional similarities.

Acknowledgments

We thank anonymous reviewers whose comments helped to improve this paper. We also thank IFOMIS institute (Saarbrucken) and the ReSIST project for allowing us to use the COCR and COCS corpora.

References

Avizienis, A., Grigonyte, G., Haller, J., von Henke, F., Liebig, T., and Noppens, O. 2009. *Organizing Knowledge as an Ontology of the Domain of Resilient Computing by Means of NLP - An Experience Report.* In Proc. of the 22th Int. FLAIRS Conf. AAAI Press, pp. 474-479.

Barzilay, R. and McKeown, K. R. 2001. *Extracting Paraphrases from a Parallel Corpus.* In Proc. of the 39th meeting of ACL, pp. 50-57.

Caraballo, S. A. 1999. *Automatic Construction of a Hypernym-labeled Noun Hierarchy from Text.* In Proc. of 37th meeting of ACL 1999, pp 120-126.

Carl, M., and Schmidt-Wigger, A. 1998. *Shallow Post Morphological Processing with KURD.* In Proc. of the Conf. on New Methods in Language Processing.

Cordeiro, J.P., Dias, G. and Brazdil, P. 2007. *Learning Paraphrases from WNS Corpora.* In Proc. of the 20th Int. FLAIRS Conf. AAAI Press, pp. 193-198.

Cordeiro, J.P., Dias, G. and Cleuziou, G. 2007. *Biology Based Alignments of Paraphrases for Sentence Compression.* In Proc. of the 20th meeting of ACL, workshop PASCAL, pp. 177-184.

Dias, G., Moraliyski, R., Cordeiro, J.P., Doucet, A. and Ahonen-Myka, H. 2010. *Automatic Discovery of Word Semantic Relations using Paraphrase Alignment and Distributional Lexical Semantics Analysis.* In Journal of Natural Language Engineering, Cambridge University Press. ISSN 1351-3249, pp. 1-26.

Dolan, B., Quirk, C. and Brockett, C. 2004. *Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources.* In Proc. of the 20th int. Conf. on Computational Linguistics.

Dunning T. D 1993. *Accurate Methods for the Statistics of Surprise and Coincidence.* In Computational Linguistics, 19(1), pp. 61-74.

Freitag, D., Blume, M., Byrnes, J., Chow, E., Kapadia, S., Rohwer, R. and Wang, Z. 2005. *New Experiments in Distributional Representations of Synonymy.* In Proc. of 9th conf. on Computational Natural Language Learning, pp. 25-32.

Giovannetti, E., Marchi, S. and Montemagni, S. 2008. *Combining Statistical Techniques and Lexico-Syntactic Patterns for Semantic Relations Extraction from Text.* In Proc. of the 5th Workshop on Semantic Web Applications and Perspectives.

Girju, R., Giuglea, A. M., Olteanu, M., Fortu, O., Bolohan, O. and Moldovan, D. 2004. *Support Vector Machines Applied to the Classification of Semantic Relations in Nominalized Noun Phrases.* In Proc. of the HLT-NAACL Workshop on Computational Lexical Semantics, pp. 68-75.

Hagiwara, M. O. Y. and Katsuhiko, T. 2009. *Supervised Synonym Acquisition using Distributional Features and Syntactic Patterns.* In Information and Media Technologies 4(2), pp. 558-582.

Hearst, M. A. 1992. *Automatic Acquisition of Hyponyms from Large Text Corpora.* In Proc. of the 14th conf. on Computational Linguistics, pp. 539-545.

Heylen, K., Peirsman, Y., Geeraerts, D. and Speelman, D. 2008. *Modelling Word Similarity: an Evaluation of Automatic Synonymy Extraction Algorithms.* In Proc. of the 6th LREC.

Hindle, D. 1990. *Noun Classification from Predicate-Argument Structures.* In Proc. of the 28th meeting of ACL, pp. 268-275.

Inkpen, D. 2007. *A Statistical Model for Near-Synonym choice.* In ACM Trans. Speech Lang. Process. 4(1), 1-17.

Kiefer, J. 1953. *Sequential Minimax Search for a Maximum.* In Proc. of the American Mathematical Society 4, pp. 502-506.

Lin, D. 1998. *Automatic Retrieval and Clustering of Similar Words.* In Proc. of the 17th Int. Conf. on Computational Linguistics, pp. 768-774.

Maas, D., Rosener, Ch., and Theofilidis, A. 2009. *Morphosyntactic and Semantic Analysis of Text: The MPRO Tagging Procedure.* Workshop on systems and frameworks for computational morphology. Springer., pp. 76-87.

Maynard, D. F. A. and Peters, W. 2009. *Using lexico-syntactic Ontology Design Patterns for Ontology Creation and Population.* In Proc. of the Workshop on Ontology Patterns.

Moraliyski, R., and Dias, G. 2007. *One Sense per Discourse for Synonym Detection*. In Proc. of the Int. Conf. On Recent Advances in NLP, Bulgaria, pp. 383-387.

Muller, P., Hathout, N. and Gaume, B. 2006. *Synonym Extraction Using a Semantic Distance on a Dictionary*. In Proc. of the 1st Workshop on Graph-Based Methods for NLP, pp. 65-72.

Needleman, S. B. and Wunsch, C. D. 1970. *A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of two Proteins*. In Journal of Molecular Biology 48(3), pp. 443-453.

Ohshima, H. and Tanaka, K. 2009. *Real Time Extraction of Related Terms by Bi-directional lexico-syntactic Patterns from the Web*. In Proc. of the 3rd Int. Conf. on Ubiquitous Information Management and Communication, pp. 441-449.

Riloff, E. 1993. *Automatically Constructing a Dictionary for Information Extraction Tasks*. In Proc. of the 11th Nat. Conf. on Artificial Intelligence, pp. 811-816.

Roark, B. and Charniak, E. 1998. *Noun-phrase Co-occurrence Statistics for Semiautomatic Semantic Lexicon Construction*. In Proc. of the 17th Int. Conf. on Computational Linguistics, pp. 1110-1116.

Smith, T. and Waterman, M. 1981. *Identification of Common Molecular Subsequences*. In Journal of Molecular Biology 147, pp. 195-197.

Sowa, J. F. and Siekmann, J. H. 1994. *Conceptual Structures: Current Practices*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Stevenson, M. and Greenwood, M. 2006. *Comparing Information Extraction Pattern Models*. In Proc. of the Workshop on Information Extraction Beyond the Document, ACL, pp. 29-35.

Terra, E. and Clarke, C. 2003. *Frequency Estimates for Statistical Word Similarity Measures*. In Proc. of HTL/NAACL 2003, pp. 165-172.

Turney, P. D. 2001. *Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL*. Lecture Notes in Computer Science, 2167, pp. 491-502.

Turney, P. D., Littman, M. L., Bigham, J. and Shnayder, V. 2003. *Combining Independent Modules in Lexical Multiple-choice Problems*. In Recent Advances in NLP III: Selected Papers, pp. 101-110.

Turney, P. D. 2008. *A Uniform Approach to Analogies, Synonyms, Antonyms and Associations*. In Proc .of the 22nd Int. Conf. on Computational Linguistics, pp. 905-912.

Van der Plas, L. and Tiedemann, J. 2006. *Finding Synonyms Using Automatic Word Alignment and Measures of Distributional Similarity*. In Proc. of the 21st Int. Conf. on Computational Linguistics, pp. 866-873.

Wei, X., Peng, F., Tseng, H., Lu, Y. and Dumoulin, B. 2009. *Context Sensitive Synonym Discovery for Web Search Queries*. In Proc. of the 18th ACM conference on Information and Knowledge Management, pp. 1585-1588.

Wu, H. and Zhou, M. 2003. *Optimizing Synonym Extraction Using Monolingual and Bilingual Resources*. In Proc. of the 2nd Int. Workshop on Paraphrasing, pp. 72-79.

Finding the Storyteller:
Automatic Spoiler Tagging using Linguistic Cues

Sheng Guo
Department of Computer Science
Virginia Tech
guos@cs.vt.edu

Naren Ramakrishnan
Department of Computer Science
Virginia Tech
naren@cs.vt.edu

Abstract

Given a movie comment, does it contain a spoiler? A spoiler is a comment that, when disclosed, would ruin a surprise or reveal an important plot detail. We study automatic methods to detect comments and reviews that contain spoilers and apply them to reviews from the IMDB (Internet Movie Database) website. We develop topic models, based on Latent Dirichlet Allocation (LDA), but using linguistic dependency information in place of simple features from bag of words (BOW) representations. Experimental results demonstrate the effectiveness of our technique over four movie-comment datasets of different scales.

1 Introduction

In everyday parlance, the notion of 'spoilers' refers to information, such as a movie plot, whose advance revelation destroys the enjoyment of the consumer. For instance, consider the movie *Derailed* which features Clive Owen and Jennifer Aniston. In the script, Owen is married and meets Aniston on a train during his daily commute to work. The two of them begin an affair. The adultery is noticed by some inscrupulous people who proceed to blackmail Owen and Aniston. To experience a spoiler, consider this comment from *imdb.com*:

> I can understand why Aniston wanted to do this role, since she gets to play majorly against type (as the supposedly 'nice' girl who's really - oh no! - part of the scam), but I'm at a loss to figure out what Clive Owen is doing in this sub-par, unoriginal, ugly and overly violent excuse for a thriller.

i.e., we learn that Aniston's character is actually a not-so-nice person who woos married men for later blackmail, and thus a very suspenseful piece of information is revealed. Automatic ways to detect spoilers are crucial in large sites that host reviews and opinions.

Arguably, what constitutes a spoiler is inherently a subjective assessment and, for movies/books with intricate storylines, some comments are likely to contain more spoilers than others. We therefore cast the spoiler detection problem as a ranking problem so that comments that are more likely to be spoilers are to be ranked higher than others. In particular, we rank user comments w.r.t. (i.e., given) the movie's synopsis which, according to *imdb*, is '[a detailed description of the movie, including spoilers, so that users who haven't seen a movie can read anything about the title]'.

Our contributions are three fold. (i) We formulate the novel task of spoiler detection in reviews and cast it as ranking user comments against a synopsis. We demonstrate how simple bag-of-words (BOW) representations need to be augmented with linguistic cues in order to satisfactorily detect spoilers. (ii) We showcase the ability of dependency parses to extract discriminatory linguistic cues that can distinguish spoilers from non-spoilers. We utilize an LDA-based model (Wei and Croft, 2006) to probabilistically rank spoilers. Our approach does not require manual tagging of positive and negative examples – an advantage that is crucial to large scale implementation. (iii) We conduct a detailed experimental evaluation with *imdb* to assess the effectiveness of our framework. Using manually tagged com-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 412–420,
Beijing, August 2010

ments for four diverse movies and suitably configured design choices, we evaluate a total of 12 ranking strategies.

2 LDA

Probabilistic topic modeling has attracted significant attention with techniques such as probabilistic latent semantic analysis (PLSA) (Hofmann, 1999) and LDA (Blei et al., 2003; Griffiths and Steyvers, 2004; Heinrich, 2008; Steyvers and Griffiths, 2007). We discuss LDA in detail due to its centrality to our proposed techniques. As a generative model, LDA describes how text could be generated from a latent set of variables denoting topics. Each document is modeled as a mixture of topics, and topics are modeled as multinomial distributions on words.

An unlabeled training corpus can be used to estimate an LDA model. Many inference methods have been proposed, e.g., variational methods (Blei et al., 2003), expectation propagation (Griffiths and Steyvers, 2004), Gibbs sampling (Griffiths and Steyvers, 2004), and a collapsed variational Bayesian inference method (Teh et al., 2007). Gibbs sampling, as a specific form of Markov chain Monte Carlo (MCMC), is a popular method for estimating LDA models. After an LDA model is estimated, it can be used in a very versatile manner: to analyze new documents, for inference tasks, or for retrieval/comparison functions. For instance, we can calculate the probability that a given word appears in a document conditioned on other words. Furthermore, two kinds of similarities can be assessed: between documents and between words (Steyvers and Griffiths, 2007). The similarity between two documents can also be used to retrieve documents relevant to a query document (Heinrich, 2008). Yet another application is to use LDA as a dimensionality reduction tool for text classification (Blei et al., 2003).

To improve LDA's expressiveness, we can relax the bag-of-words assumption and plug in more sophisticated topic models (Griffiths et al., 2005; Griffiths et al., 2007; Wallach, 2006; Wallach, 2008; Wang and Mccallum, 2005; Wang et al., 2007). sLDA (supervised LDA), as a statistical model of labeled collections, focuses on the prediction problem (Blei and Mcauliffe, 2007). The correlated topic model (CTM) (Blei and Lafferty, 2007) addresses plain LDA's inability to model topic correlation. The author-topic model (AT) (Steyvers et al., 2004) considers not only topics but also authors of the documents, and models documents as if they were generated by a two-stage stochastic process.

3 LDA-based spoiler ranking

3.1 Methods

Based on the fact that a spoiler should be topically close to the synopsis, we propose three methods to solve the spoiler ranking problem. The first two use LDA as a preprocessing stage, whereas the third requires positive training data.

Predictive perplexity: Our first method is motivated by the use of LDA-based predictive perplexity (PP) for collaborative filtering (Blei et al., 2003). Here, the PP metric is evaluated over a fixed test dataset in order to empirically compare LDA with other models (pLSI, mixture of unigrams). In our work, we view documents as analogous to users, and words inside documents as analogous to movies. Given a group of known words, we predict the other group of unkown words. We can either calculate the predictive perplexity value from each movie comment Com to the unique synopsis (PP1), or from the synopsis Syn to each comment (PP2).

$$PP1(Syn, \mathbf{w}_{com}) = exp\{-\frac{\sum_{d=1}^{M_{syn}} \log p(w_d|\mathbf{w}_{com})}{M_{syn}}\}$$

$$PP2(Com, \mathbf{w}_{syn}) = exp\{-\frac{\sum_{d=1}^{M_{com}} \log p(w_d|\mathbf{w}_{syn})}{M_{com}}\}$$

In the equations above, $p(w_d|\mathbf{w}_{com})$ and $p(w_d|\mathbf{w}_{syn})$ are the probabilities to generate the word (w_d) from a group of observed words $\mathbf{w}_{obs}$ (either a comment $\mathbf{w}_{com}$ or a synopsis $\mathbf{w}_{syn}$). $p(w|\mathbf{w}_{obs}) = \int \sum_z p(w|z)p(z|\theta)p(\theta|\mathbf{w}_{obs})d\theta$
M_{com} or M_{syn} is the length of a comment or a synopsis. Notice that $p(\theta|\mathbf{w}_{obs})$ can be easily calculated after estimating LDA model by Gibbs sampling. It is also discussed as "predictive likelihood ranking" in (Heinrich, 2008).

Symmetrized KL-divergence: Since documents are modeled as mixtures of topics in LDA, we can calculate the similarity between synopsis and comment by measuring their topic distributions' similarity. We adopt the widely-used symmetrized Kullback Leibler (KL) divergence (Heinrich, 2008; Steyvers and Griffiths, 2007) to measure the difference between the two documents' topic distributions, $sKL(Syn, Com) = \frac{1}{2}[D_{KL}(Syn\|Com) + D_{KL}(Com\|Syn)]$ where $D_{KL}(p\|q) = \sum_{j=1}^{T} p_j \log_2 \frac{p_j}{q_j}$

LPU: Viewing the spoiler ranking problem as a retrieval task given the (long) query synopsis, we also consider the LPU (Learning from Positive and Unlabeled Data) method (Liu et al., 2003). We apply LPU as if the comment collection was the unlabeled dataset, and the synopsis together with few obvious spoiler comments as the positive training data.

3.2 Dependency Parsing

LDA, as a topic model, is widely used as a clustering method and dimensionality reduction tool. It models text as a mixture of topics. However, topics extracted by LDA are not necessarily the same topics as judged by humans since the definition of topic is very subjective. For instance, when conducting sentimental polarity analysis, we hope that topics are clusters concerning one certain kind of subjective sentiment. But for other purposes, we may desire topics focusing on broad 'plots.' Since LDA merely processes a collection according to the statistical distribution of words, its results might not fit either of these two cases mentioned above.

In a basic topic model (section 3.1), neither the order of a sequence of words nor the semantic connections between two words affect the probabilistic modeling. Documents are generated only based on a BOW assumption. However, word order information is very important for most text-related tasks, and simply discarding the order information is inappropriate. Significant work has gone in to address this problem. Griffiths et al. use order information by incorporating collocations (Griffiths et al., 2005; Griffiths et al., 2007). They give an example of the collocation "*united kingdom*", which is ideally treated as a single chunk than two independent words. However, this model can only be used to capture collocations involving sequential terms. Their extended model (Griffiths et al., 2007) integrates topics and syntax, and identifies syntactic classes of words based on their distribution. More sophisticated models exist (Wallach, 2006; Wang and Mccallum, 2005; Wang et al., 2007; Wallach, 2008) but all of them are focused on solving linguistic analysis tasks using topic models. In this paper, however, our focus is on utilizing dependency information as a preprocessing step to help improve the accuracy of LDA models.

In more detail, we utilize dependency parsing to breakup sentences and treat parses as independent 'virtual words,' to be added to the original BOW-based LDA model. In our experiments we employ the Stanford typed dependency parser [1] (Marneffe et al., 2006) as our parsing tool. We use collapsed typed dependencies (a.k.a. grammatical relations) to form the virtual words. However, we do not incorporate all the dependencies. We only retain dependencies whose terms have the part-of-speech tags such as "*NN*", "*VB*", "*JJ*", "*PRP*" and "*RB*"[2], since these terms have strong plot meaning, and are close to the movie topic. Fig. 2 shows a typical parsing result from one sample sentence. This sentence is taken from a review of *Unbreakable*.

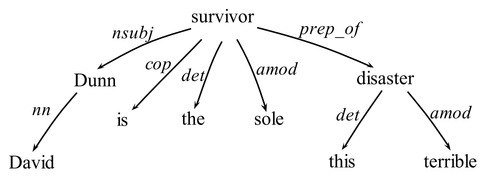
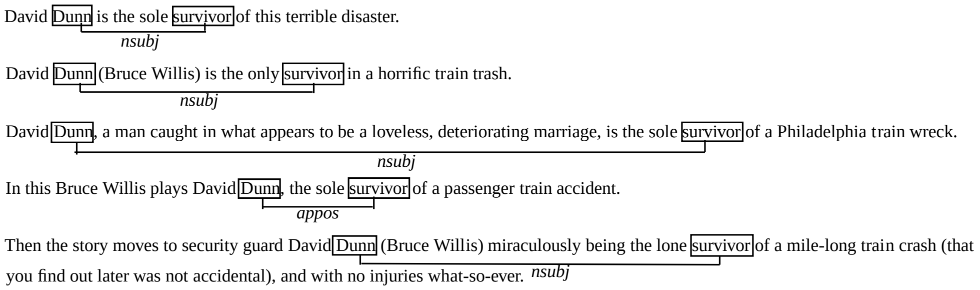


Figure 2: Dependency parse of "David Dunn is the sole survivor of this terrible disaster".

Consider Fig. 1, which depicts five sample sentences all containing two words: "*Dunn*" and "*survivor*". Although these sentences appear different, these two words above refer to the same individual. By treating dependencies as virtual words, we can easily integrate these plot-related relations into an LDA model. Notice that among these five sentences, the grammatical relations between these two words are different: in the fourth sentence, "*survivor*" serves as an appositional modifier of the term "*Dunn*"(appos), whereas in

[1] http://nlp.stanford.edu/software, V1.6

[2] In the implementation, we actually considered all the POS tags with these five tags as prefix, such as "*NNS*", "*VBN*", etc.


David Dunn is the sole survivor of this terrible disaster.

David Dunn (Bruce Willis) is the only survivor in a horrific train trash.

David Dunn, a man caught in what appears to be a loveless, deteriorating marriage, is the sole survivor of a Philadelphia train wreck.

In this Bruce Willis plays David Dunn, the sole survivor of a passenger train accident.

Then the story moves to security guard David Dunn (Bruce Willis) miraculously being the lone survivor of a mile-long train crash (that you find out later was not accidental), and with no injuries what-so-ever.

Figure 1: Four sentences with the same topical connection between "Dunn" and "survivor". We integrate this relation into LDA by treating it as a virtual word "Dunn-survivor."

other sentences, "*Dunn*" serves as the nominal subject of "*survivor*"(nsubj). What is important to note is that the surface distances between these given words in different sentences vary a lot. By utilizing dependency parsing, we can capture the semantic connection which is physically separated by even as much as 15 words, as in the third sentence.

We evaluate *topic drift* among the results from plain LDA. We mainly check whether plain LDA will assign the same topic to those terms that have specific linguistic dependency relations. We only consider the following four types of dependencies for evaluation[3]:

- Relations with two noun terms: <NN, NN>, such as "*appos*", "*nn*", "*abbrev*" etc.;
- Relations with one noun and one adjective: <NN, JJ>, like "*amod*";
- Relations with one noun and one verb: <NN, VB>, such as "*agent*", "*dobj*", etc.;
- Relations with only one noun: <NN, *>, which is the relaxed version of <NN, NN>;

We experimented with different pre-set topic numbers (500, 50, and 2) and conducted experiments on four different movie comment collections with LDA analysis. Table 1 shows that <NN, NN> dependency has the highest chance to be topic-matched[4] than other relations. However, all dependencies have very low percentage to be topic-matched, and with a topic number of 2, there remained a significant amount of unmatched <NN, NN> dependencies, demonstrating that simply doing plain LDA may not capture the plot "topic" as we desire.

Observing the results above, each method from section 3.1 (PP1, PP2, sKL and LPU) can be extended by: (1) using BOW-based words, (2) using only dependency-based words, or (3) using a mix of BOW and dependency (dependencies as virtual words). This induces 12 different ranking strategies.

Table 1: Topic match analysis for plain LDA (Each entry is the ratio of topic-matched dependencies to all dependencies)

topic number = 500				
Movie Name	<NN, NN>	<NN, JJ>	<NN, VB>	<NN, *>
Unbreakable	772/3024	412/4411	870/19498	5672/61251
Blood Diamond	441/1775	83/553	80/1012	609/3496
Shooter	242/1846	42/1098	114/2150	1237/15793
Role Models	409/2978	60/1396	76/2529	559/7276
topic number = 50				
Movie Name	<NN, NN>	<NN, JJ>	<NN, VB>	<NN, *>
Unbreakable	1326/3024	953/4411	3354/19498	14067/61251
Blood Diamond	806/1775	151/553	210/1012	1194/3496
Shooter	584/1846	204/1098	392/2150	3435/15793
Role Models	1156/2978	190/1396	309/2529	1702/7276
topic number = 2				
Movie Name	<NN, NN>	<NN, JJ>	<NN, VB>	<NN, *>
Unbreakable	2379/3024	3106/4411	13606/19498	43876/61251
Blood Diamond	1391/1775	404/553	761/1012	2668/3496
Shooter	1403/1846	768/1098	1485/2150	11008/15793
Role Models	2185/2978	908/1396	1573/2529	4920/7276

[3] Here we use <NN, JJ> to express relations having NN and JJ terms, but not necessarily in that order. Also, NN represents all tags related with nouns in the Penn Treebank Tagset, such as NNS. This applies to all the four expressions here.

[4] When both the left term and the right term of a dependency share the same topic, the relation is topic-matched.

Table 2: Some examples of incorrect spoiler tagging in IMDb (italicized sentences are spoilers).

No.	Tag by IMDb	Comment in IMDb
1	Spoiler	The whole film is somewhat slow and it would've been possible to add more action scenes. Even though I liked it very much (6.8/10) I think it is less impressive than "The Sixth Sense" (8.0/10). I would like to be more specific with each scene but it will turn this comment into a spoiler so I will leave it there. I recommend you to see the movie if you come from the basic Sci-Fi generation, otherwise you may feel uncomfortable with it. Anyway once upon a time you were a kid in wonderland and everything was possible. [tt0217869]
2	Spoiler	This is one of the rare masterpiece that never got the respect it deserved because people were expecting sixth sense part 2.Sixth sense was a great film but this is M.N. Shyamalan's best work till date. This is easily one of my top 10 films of all time. Excellent acting, direction, score, cinematography and mood. This movie will hold you in awe from start to finish and any student of cinema would tell what a piece of art this film is. The cast is phenomenal, right from bruce willis to sam jackson and penn , everyone is spectacular in their roles and they make u realise that you do not need loud dramatic moments to create an impact, going slow and subtle is the trick here. This is not a thriller, it's a realistic superhero film. [tt0217869]
3	Spoiler	I can't believe this movie gets a higher rating than the village. OK, after thinking about it, i get the story of unbreakable and i understand what it's trying to say. I do think the plot and the idea is captivating and interesting. Having said that, i don't think the director did anything to make this movie captivating nor interesting. It seemed to try too hard to make this movie a riddle for the audience to solve. The pace was slow at the beginning and ended just as it was getting faster. I remember going out of the cinema, feeling frustrated and confused. it's not until i thoroughly thought about it that i understood the plot. I believe a good movie should engaged the audience and be cleverly suspenseful without confusing the audience too much. Unbreakable tried to be that but failed miserably. 2 out of 10, see the village instead. [tt0217869]
4	Spoiler	This movie touched me in ways I have trouble expressing, and brings forth a message one truly need to take seriously! I was moved, and the ending brought a tear to my eye, as well as a constant two-minute shiver down my spine. It shows how our western way of life influence the lives of thousands of innocents, in a not-so-positive way. Conflict diamonds, as theme this movie debates, are just one of them. Think of Nike, oil, and so on. We continually exploit "lesser developed" nations for our own benefit, leaving a trail of destruction, sorrow, and broken backs in our trail. I, for one, will be more attentive as to what products I purchase in the future, that's for sure. [tt0450259]
5	Non-spoiler	... But the movie takes a while to get to the point. ***"Mr. Glass" has caused lots of mass tragedies in order to find the UNBREAKABLE person. Thus, he is both a mentor and a MONSTER***. ... [tt0217869]
6	Non-spoiler	... This film is about a sniper who loses his best friend while on a shooting mission. A few years later, he is now retired and living in a woodland with his do. Then he is visited by the military to plan an assassination of the president. The shot is fired. ***Unfortunately he is set up to being the shooter and is hunted by cops everywhere. He must find out why he has been set up and also try and stop the real killers***. ... [tt0822854]

4 Experimental Results

4.1 Data preparation

IMDb boasts a collection of more than 203,000 movies (from 1999 to 2009), and the number of comments and reviews for these movies number nearly 970,000. For those movies with synopsis provided by IMDb, the average length of their synopses is about 2422 characters[5]. Our experimental setup, for evaluation purposes, requires some amount of labeled data. We choose four movies from IMDb, together with 2148 comments. As we can see in Table 3, these four movies have different sizes of comment sets: the movie "Unbreakable" (2000) has more than 1000 comments, whereas the movie "Role Models" (2008) has only 123 comments.

Table 3: Evaluation dataset about four movies with different numbers of comments.

Movie Name	IMDB ID	#Comments	#Spoilers
Unbreakable	tt0217869	1219	205
Blood Diamond	tt0450259	538	147
Shooter	tt0822854	268	73
Role Models	tt0430922	123	39

We labeled all the 2148 comments for these four movies manually, and as Table 3 shows, about 20% of each movie's comments are spoilers. Our labeling result is a little different from the current labeling in IMDb: among the 2148 comments, although 1659 comments have the same labels with IMDb, the other 489 are different (205 are treated as spoilers by IMDb but non-spoilers by us; vice versa with 284) The current labeling system in IMDb is very coarse: as shown in Table 2, the first four rows of comments are labeled as spoilers by IMDb, but actually they are not. The last two rows of comments are ignored by IMDb; however, they do expose the plots about the twisting ends.

After crawling all the comments of these four movies, we performed sentence chunking using the LingPipe toolkit and obtained 356 sentences for the four movies' synopses, and 26964 sentences for all the comments of these four movies. These sentences were parsed to extract dependency information: we obtained 5655 dependencies for all synopsis sentences and 448170 dependencies for all comment sentences. From these, we only retain those dependencies that have at least one noun term in either left side or the right side. For measures which require the dependency information, the dependencies are re-organized and treated as a new term planted in the text.

[5] Those movies without synopsis are not included.

4.2 Experiments

4.2.1 Topic number analysis

One of the shortcomings of LDA-based methods is that they require setting a number of topics in advance. Numerous ways have been proposed to handle this problem (Blei et al., 2004; Blei et al., 2003; Griffiths and Steyvers, 2004; Griffiths et al., 2007; Heinrich, 2008; Steyvers and Griffiths, 2007; Teh et al., 2006). Perplexity, which is widely used in the language modeling community, is also used to predict the best number of topics. It is a measure of how well the model fits the unseen documents, and is calculated as average per-word held-out likelihood. The lower the perplexity is, the better the model is, and therefore, the number of topic is specified as the one leading to the best performance. Griffiths and Steyvers (Griffiths and Steyvers, 2004) also discuss the standard Bayesian method which computes the posterior probability of different models given the observed data. Another method from non-parametric Bayesian statistics automatically helps choose the appropriate number of topics, with flexibility to still choose hyper-parameters (Blei et al., 2004; Teh et al., 2006). Although the debate of choosing an appropriate number of topics continues (Boyd-Graber et al., 2009), we utilized the classic perplexity method in our work. Heinrich (Heinrich, 2008) demonstrated that perplexity can be calculated by: $P(\tilde{\mathcal{W}}|\mathcal{M}) = \prod_{m=1}^{M} p(\tilde{\vec{w}}_m|\mathcal{M})^{-\frac{1}{N}} = exp\{-\frac{\sum_{m=1}^{M} \log p(\tilde{\vec{w}}_m|\mathcal{M})}{\sum_{m=1}^{M} N_m}\}$ We chose different topic numbers and calculated the perplexity value for the 20% held-out comments. A good number of topics was found to be between 200 and 600 for both Bow-based strategy and Bow+Dependency strategy, and is also affected by the size of movie comment collections. (We used 0.1 as the document topic prior, and 0.01 as the topic word prior.)

4.2.2 LDA analysis process

As discussed earlier, our task is to rank all the comments according to their possibilities of being a spoiler. We primarily used four methods to do the ranking: PP1, PP2, sKL, and the LPU method. For each method, we tried the basic model using “bag-of-words”, and the model using dependency parse information (only), and also with both BOW and dependency information mixed. We utilize LingPipe LDA clustering component which uses Gibbs sampling.

Among the four methods studied here, PP1, PP2 and sKL are based on LDA preprocessing. After obtaining the topic-word distribution and the posterior distributions for topics in each document, the PP1 and PP2 metrics can be easily calculated. The symmetrized KL divergence between each pair of synopsis and comment is calculated by comparing their topic distributions. LPU method, as a text classifier, requires a set of positive training data. We selected those comments which contain terms or phrases as strong hint of spoiler (using a list of 20 phrases as the filter, such as “spoiler alert”, “spoiler ahead”, etc). These spoiler comments together with the synopsis, are treated as the positive training data. We then utilized LPU to label each comment with a real number for ranking.

4.3 Evaluation

To evaluate the ranking effects of the 12 strategies, we plot *n*-best precision and recall graphs, which are widely used for assessing collocation measures (Evert and Krenn, 2001; Pecina and Schlesinger, 2006). Fig. 3 visualizes the precision-recall graphs from 12 different measures for the four movie comment collections. The *x*-axis represents the proportion of the ranking list, while the *y*-axis depicts the corresponding precision or recall value. The upper part of the figure is the result for the movie which contains more than 1000 comments, while the bottom part demonstrates the result for the relatively small comment collection. The *n*-best evaluation shows that for all the four movie comment collections, PP1_mix and PP1 perform significantly better than the other methods, and the dependency information helps to increase the accuracy significantly, especially for the larger size collection. The LPU method, though using part of the positive training data, did not perform very well. The reason could be that although some of the users put the warning phrases (like “spoiler alert”) ahead of their comments, the comment might contain only indirect plot-revealing information. This also reflects that a spoiler tagging method by us-

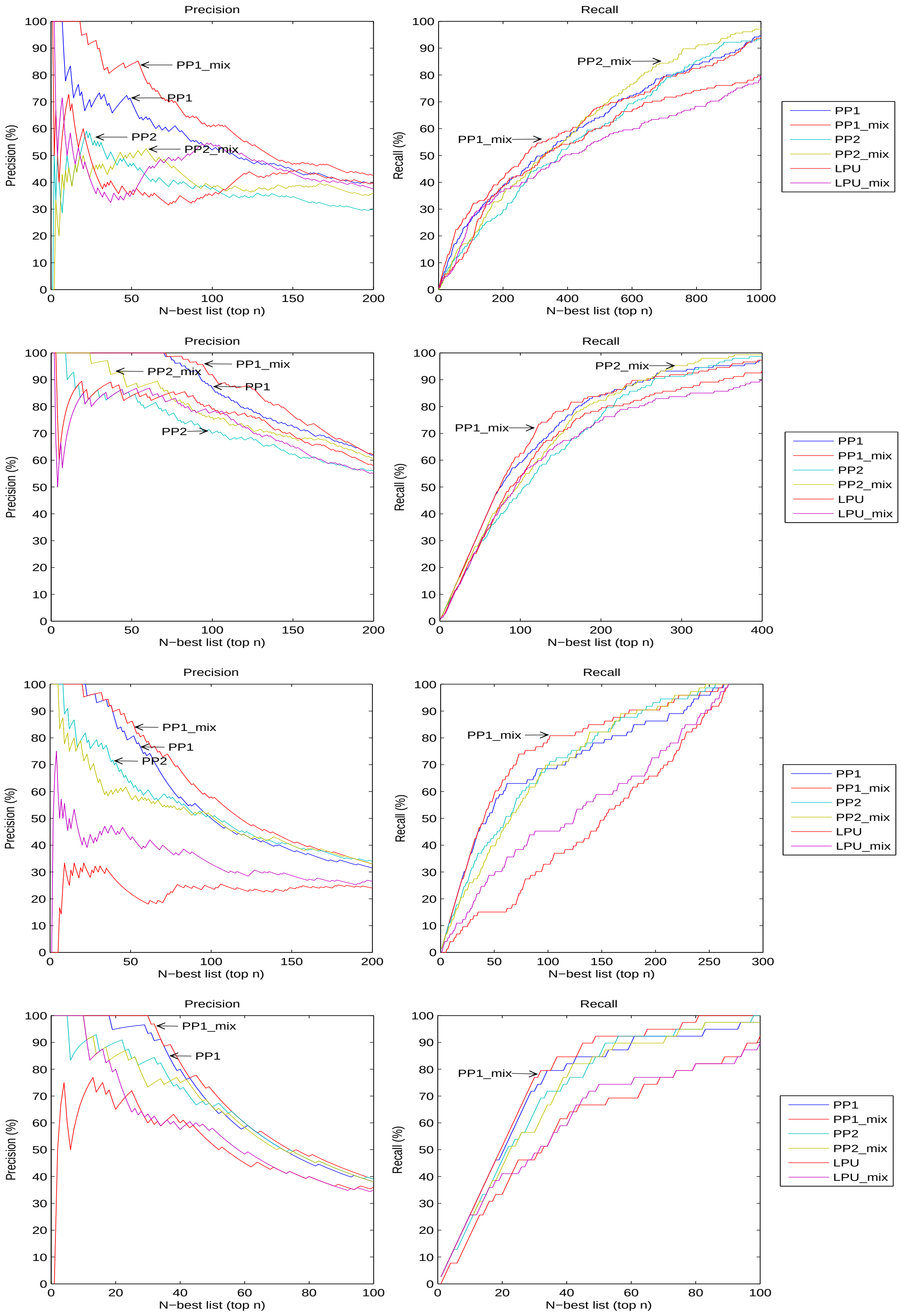


Figure 3: N-best(top *n*th) evaluation (Burnin period = 100): comparison of precision-recall for different methods on four movie comment collections. The PP1 method with BOW and dependency information mixed performs the best among all the measures. Other six methods such as dependency only and KL-based which do not give good performance are ignored in this figure to make it readable. Full comparison is available at: http://sites.google.com/site/ldaspoiler/

ing only keywords typically will not work. Finally, the approach to directly calculating the symmetrized KL divergence seems to be not suitable, either.

4.4 LDA iteration analysis

We also compared the *average precision* values and *normalized discounted cumulative gain* (nDCG) values (Croft et al., 2009; Järvelin and Kekäläinen, 2002) of the ranking results with different parameters for Gibbs sampling, such as burnin period and sample size. Average precision is calculated by averaging the precision values from the ranking positions where a valid spoiler is found, and the nDCG value for the top-p list is calculated as $nDCG_p = \frac{DCG_p}{IDCG} \cdot DCG_p$ is defined as: $DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$ where rel_i is 1 when the *i*-th comment in the list is judged as a real spoiler, and 0, otherwise. IDCG denotes the maximum possible DCG value when all the real spoilers are ranked at the top (*perfect ranking*) (Järvelin and Kekäläinen, 2002).

Table 4: Comparison of ranking by PP_mix using different parameters for Gibbs sampling (analyzed on the top 150 ranking lists, and the values in the table are the mean of the accuracy from four movie comment collections).

Burnin	<S=100; Lag=2>		<S=10; Lag=2>		<S=1; Lag=2>	
	AvgP (%)	nDCG	AvgP (%)	nDCG	AvgP (%)	nDCG
400	80.85	0.951	78.2	0.938	78.1	0.94
200	80.95	0.951	80.5	0.948	79.1	0.94
100	87.25	0.974	80.2	0.943	82.4	0.96
50	81.5	0.958	79.5	0.942	80.0	0.94
10	78.9	0.944	79.5	0.949	75.9	0.92
1	79.4	0.940	79.2	0.952	58.0	0.86

As we can see from Table 4, the accuracy is not affected too much as long as the burin period for the MCMC process is longer than 50 and the sample size retained is larger than 10. In our experiments, we use 100 as the burin parameter, and beyond that, 100 samples were retained with sample lag of 2.

4.5 Representative results

As shown in Table 5, we find that the basic BOW strategy prefers the longer comments whereas the strategy that uses dependency information prefers the shorter ones. Although it is reasonable that a longer comment would have a higher probability of revealing the plot, methods which prefers the longer comments usually leave out the short spoiler comments. By incorporating the dependency information together with the basic BOW, the new method reduces this shortcoming. For instance, consider one short comment for the movie "*Unbreakable* (2000)":

> This is the same formula as Sixth Sense – from the ability to see things other people don't, to the shocking ending. Only this movie is just not plausible – I mean Elijah goes around causing disasters, trying to see if anyone is "Unbreakable" – it's gonna take a lot of disasters because its a big world.

whcih is ranked as the 27th result in the PP1_mix method, whereas the BOW based PP1 method places it at the 398th result in the list. Obviously, this comment reveals the twisting end that it is Elijah who caused the disasters.

Table 5: Comparison of average length of the top-50 comments of 4 movies from 2 strategies.

	Role Models	Shooter	Blood Diamond	Unbreakable
BOW	2162.14	2259.36	2829.86	1389.18
Dependency	1596.14	1232.12	2435.58	1295.72

5 Conclusions and future work

We have introduced the spoiler detection problem and proposed using topic models to rank movie comments according to the extent they reveal the movie's plot. In particular, integrating linguistic cues from dependency information into our topic model significantly improves the ranking accuracy.

In future work, we seek to study schemes which can segment comments to potentially identify the relevant spoiler portion automatically. The automatic labeling idea of (Mei et al., 2007) can also be studied in our framework. Deeper linguistic analysis, such as named entity recognition and semantic role labeling, can also be conducted. In addition, evaluating topic models or choosing the right number of topics using dependency information can be further studied. Finally, integrating the dependency relationships more directly into the probabilistic graphical model is also worthy of study.

References

Blei, David M. and John D. Lafferty. 2007. A correlated topic model of science. *Annals of Applied Statistics*, 1(1):17–35.

Blei, David M. and Jon D. Mcauliffe. 2007. Supervised topic models. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of machine learning research*, 3:993–1022.

Blei, David M., T. Gri, M. Jordan, and J. Tenenbaum. 2004. Hierarchical topic models and the nested chinese restaurant process. In *Proceedings of the 18th Annual Conference on Neural Information Processing Systems*.

Boyd-Graber, Jordan, Jonathan Chang, Sean Gerrish, Chong Wang, and David Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*.

Croft, Bruce, Donald Metzler, and Trevor Strohman. 2009. *Search Engines: Information Retrieval in Practice*. Addison Wesley, 1 edition.

Evert, Stefan and Brigitte Krenn. 2001. Methods for the qualitative evaluation of lexical association measures. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*.

Griffiths, Thomas L. and M. Steyvers. 2004. Finding scientific topics. In *Proceedings of the National Academy of Sciences of the United States of America*, 101 Suppl 1:5228–5235, April.

Griffiths, Thomas L., Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. 2005. Integrating topics and syntax. In *Proceedings of the 19th Annual Conference on Neural Information Processing Systems*.

Griffiths, Thomas L., Mark Steyvers, and Joshua B. Tenenbaum. 2007. Topics in semantic representation. *Psychological Review*, 114(2):211–244, April.

Heinrich, Gregor. 2008. Parameter estimation for text analysis. Technical report, University of Leipzig.

Hofmann, Thomas. 1999. Probabilistic latent semantic analysis. In *Proceedings of 15th Conference on Uncertainty in Artificial Intelligence*.

Järvelin, Kalervo and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446.

Liu, Bing, Yang Dai, Xiaoli Li, Wee Lee, and Philip S. Yu. 2003. Building text classifiers using positive and unlabeled examples. In *Proceedings of the 3rd IEEE International Conference on Data Mining*.

Marneffe, M., B. Maccartney, and C. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*.

Mei, Qiaozhu, Xuehua Shen, and ChengXiang Zhai. 2007. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD conference*.

Pecina, Pavel and Pavel Schlesinger. 2006. Combining association measures for collocation extraction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

Steyvers, Mark and Tom Griffiths, 2007. *Probabilistic Topic Models*. Lawrence Erlbaum Associates.

Steyvers, Mark, Padhraic Smyth, Michal R. Zvi, and Thomas Griffiths. 2004. Probabilistic author-topic models for information discovery. In *Proceedings of the 10th ACM SIGKDD conference*.

Teh, Yee Whye, Jordan, I. Michael, Beal, J. Matthew, Blei, and M. David. 2006. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, December.

Teh, Yee W., David Newman, and Max Welling. 2007. A collapsed variational bayesian inference algorithm for latent dirichlet allocation. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*.

Wallach, Hanna M. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd International Conference on Machine Learning*.

Wallach, Hanna M. 2008. *Structured topic models for language*. Ph.D. thesis, University of Cambridge.

Wang, Xuerui and Andrew Mccallum. 2005. A note on topical n-grams. Technical report, University of Massachusetts Amherst.

Wang, Xuerui, Andrew McCallum, and Xing Wei. 2007. Topical n-grams: Phrase and topic discovery, with an application to information retrieval. In *Proceedings of the 7th IEEE International Conference on Data Mining*.

Wei, Xing and Bruce W. Croft. 2006. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference*.

Detection of Simple Plagiarism in Computer Science Papers

Yaakov HaCohen-Kerner
Department of Computer Science, Jerusalem College of Technology (Machon Lev)
kerner@jct.ac.il

Aharon Tayeb
Department of Computer Science, Jerusalem College of Technology (Machon Lev)
aharontayeb@gmail.com

Natan Ben-Dror
Department of Computer Science, Jerusalem College of Technology (Machon Lev)
bd.natan@gmail.com

Abstract

Plagiarism is the use of the language and thoughts of another work and the representation of them as one's own original work. Various levels of plagiarism exist in many domains in general and in academic papers in particular. Therefore, diverse efforts are taken to automatically identify plagiarism. In this research, we developed software capable of simple plagiarism detection. We have built a corpus (C) containing 10,100 academic papers in computer science written in English and two test sets including papers that were randomly chosen from C. A widespread variety of baseline methods has been developed to identify identical or similar papers. Several methods are novel. The experimental results and their analysis show interesting findings. Some of the novel methods are among the best predictive methods.

1 Introduction

In light of the explosion in the number of available documents, fast and accurate searching for plagiarism is becoming more needed. Identification of identical and similar documents is becoming very important.

Plagiarism is the use of the language and thoughts of another work and the representation of them as one's own original work (Wikipedia, 2010; Library and Information Services, 2010). Plagiarism can be committed by "recycling" other's work as well as by one’s own work (self-plagiarism).

Various levels of plagiarism exist in many domains in general and in academic papers in particular. In addition to the ethical problem, plagiarism in Academics can be illegal if copyright of the previous publication has been transferred to another entity.

It is important to mention, that in many cases similar papers are different versions of the same work, e.g., a technical report, a poster paper, a conference paper, a journal paper and a Ph. D. dissertation.

To avoid any kind of plagiarism, all sources which were used in the completion of a work/research must be mentioned (Library and Information Services, 2010).

Over the last decade, various softwares have been built to automatically identify plagiarism (e.g., Collberg et al. (2005), Sorokina et al. (2006), and Keuskamp and Sliuzas (2007)).

In this research, we developed such a system. This system is planned to deal with simple kinds of plagiarism, e.g., copying of sentences or part of sentences. We have built a corpus that contains academic papers in computer science written in English. Most of the papers are related to the domain research of Natural Language Processing (NLP) and are from the last ten years.

The remainder of this paper is organized as follows: Section 2 gives a background regarding plagiarism. Section 3 overviews researches and systems dealing with detection of plagiarism. Section 4 describes five groups of baseline methods, which have been implemented by us to detect plagiarism. Section 5 presents the experiments that have been performed and their analysis. Section 6 gives an illustrative example. Section 7 concludes and proposes future directions for research.

2 Plagiarism

Plagiarism is defined in the 1995 Random House Compact Unabridged Dictionary as the "use or close imitation of the language and thoughts of another author and the representation of them as one's own original work."

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 421–429,
Beijing, August 2010

Self-plagiarism is the reuse of significant, identical, or nearly identical parts of one’s own work without citing the original work. In addition to the ethical issue, this phenomenon can be illegal if copyright of the previous work has been transferred to another entity. Usually, self-plagiarism is considered to be a serious ethical problem in cases where a publication needs to contain an important portion of a new material, such as in academic papers (Wikipedia, 2010).

On the other hand, it is common for researchers to rephrase and republish their research, tailoring it for different academic journals and conference articles, to disseminate their research to the widest possible interested public. However, these researchers must include in each publication a meaningful or an important portion of a new material (Wikipedia, 2010).

There are various classifications for levels of plagiarism. For instance, IEEE (2010) categorized plagiarism into five levels, or degrees, of misconduct, ranging from the most serious (Level One) to the least serious (Level Five):

Level One: The uncredited verbatim copying of a full paper, or the verbatim copying of a major portion (greater than half of the original paper)

Level Two: The uncredited verbatim copying of a large portion (less than half of the original paper).

Level Three: The uncredited verbatim copying of individual elements (e.g., paragraphs, sentences, figures).

Level Four: The uncredited improper paraphrasing of pages or paragraphs.

Level Five: The credited verbatim copying of a major portion of a paper without clear delineation (e.g., quotes or indents).

Loui (2002) handled eight allegations of plagiarism related to students' works. Collberg et al. (2005) proposes eight ranks of plagiarism.

3 Related Research

There are two main attitudes concerning discovery of similar documents: ranking and fingerprinting. Ranking methods are derived from information retrieval (IR) and are widely used in IR systems and Internet search engines. Known ranking methods are the cosine measure, the inner product, and the normalized inner product. Hoad and Zobel (2003) extended the ranking family by defining identity measures, designed for identification of co-derivative documents.

Fingerprinting aims to compare between two documents based on their fingerprints. Fingerprint methods have been used by many previous researches, e.g., Manber (1994). Heintze (1996), Lyo et al. (2001), Hoad and Zobel (2003), and Shivakumar and Garcia-Molina (1996).

3.1 Full Fingerprinting

Given a document, a full fingerprint of the document consists of the set of all the possible sequential substrings of length α in words (a definition that is based on characters is also possible). There are $N-\alpha+1$ such substrings, where N is the length of the document in words. This fingerprinting selects overlapping sub-strings. For instance, if α is 3, this method selects the 3-word phrases that begin at position 0; 1; 2; etc. The size of α is known as the fingerprint granularity. This variable can have a significant impact of the accuracy of fingerprinting (Shivakumar and Garcia-Molina, 1996).

Comparing a document X to a document Y where X's size is $|X|$ and if n is the number of substrings common to both documents then $n/|X|$ is the measure of how much of X is contained in Y.

3.2 Selective Fingerprinting

To decrease the size of a full fingerprint, there are various versions of selective fingerprints.

The simplest kind of selective fingerprinting is the "All substrings selection" described in Hoad and Zobel (2003). This fingerprinting is similar to the full fingerprinting, but it does not select overlapping sub-strings. Rather, it selects all non-overlapping substrings of size α (in words) from the document. For example, if α is 3, this strategy selects the 3-word phrases that begin at position 0; 3; 6; 9; etc.

Heintze (1996) performed various experiments using a fixed number of fingerprints independent of the size of the document and a fixed number of substrings of size α (in characters). The best results were achieved by 1,000 fingerprints with α=50. Another possibility is to work with a fixed proportion of the substrings, so that the size of the selective fingerprint is proportional to the size of the document. The main disadvantage of this possibility is space consumption.

Hoad and Zobel (2003) suggested many additional general types of selective fingerprinting, e.g., positional, frequency-based, and structure-based.

3.3 Additional Similarity Measures

SymetricSimilarity

Monostori1 et al. (2002) defined a measure called SymetricSimilarity as follows:

$SS(X, Y) = |d(X) \cap d(Y)| / |d(X) + d(Y)|$

where X and Y are the two compared documents, d(X) and d(Y) are the number of the fingerprints of X and Y, respectively, and $|d(X) \cap d(Y)|$ is the number of the common fingerprints.

S2 and S3

Bernstein and Zobel (2004) defined several additional similarity measures, such as S2 and S3:

$S2(X, Y) = |d(X) \cap d(Y)| / \min(|d(X)|, |d(Y)|)$

$S3(X, Y) = |d(X) \cap d(Y)| / ((|d(X)| + |d(Y)|)/2)$

where $\min(|d(X)|, |d(Y)|)$ is the minimal number of the fingerprints of X and Y, respectively, and $|d(X) \cap d(Y)|$ is the average number of the fingerprints of X and Y.

Rarest-in-document

The Rarest-in-Document method is one of the frequency-based methods defined by Hoad and Zobel (2003). This method chooses the substrings that produce the rarest substrings with length of k words in the document. This means that all of the substrings must be calculated and sorted according to their frequency in the document, and then the rarest of them are selected. The intuition is that sub-strings, which are less common, are more effective discriminators when comparing documents for similarity.

Anchor methods

Hoad and Zobel (2003) defined anchor methods. These methods are based on specific, predefined strings (called anchors), in the text of the document. The anchors are chosen to be common enough that there is at least one in almost every document, but not so common that the fingerprint becomes very large (Manber, 1994).

Various anchors were used by Hoad and Zobel. The anchors were randomly selected, but extremely common strings such as "th" and "it" were rejected. The 35 2-character anchor method detects all of the documents that were considered as similar by a human expert.

Additional experiments have been applied to identify the optimal size of an anchor. Manber (1994) used 50-character anchors in a collection of over 20,000 "readme" documents, identifying 3,620 sets of identical files and 2,810 sets of similar files. Shivakumar and Garcia-Molina (1996) achieved the best results with one-sentence anchors and Heintze (1996) achieved the best results with 1000-character anchors.

4 Baseline Detection Methods

To find whether there is a plagiarism, novel and old baseline methods have been implemented. These methods can be divided into five groups: full fingerprint methods, selective fingerprint methods, anchor methods, word comparison methods, and combinations of methods.

Full fingerprint methods

All the full fingerprint methods are defined for overlapping substrings of length k in words from the beginning of the document.

1. FF(k) - Full Fingerprints of length k
2. SSF(k) - SymetricSimilarity for Full fingerprints of length k
3. S2F(k) - S2 for Full fingerprints of length k
4. S3F(k) - S3 for Full fingerprints of length k
5. RDF(k) - Rarest-in-Document for Full fingerprints of length k
6. CA - Compare between the abstracts of the two documents using FF(3)

Selective Fingerprint methods

In this research, all the selective fingerprint methods are selective by the sense of non-overlapping substrings of length k in words from the beginning of the document.

7. SF(k) - Selective Fingerprints of length k

8. SSS(k) - SymetricSimilarity for Selective fingerprints of length k
9. S2S(k) - S2 for Selective fingerprints of length k
10. S3S(k) - S3 for Selective fingerprints of length k
11. RDS(k) - Rarest-in-Document for Selective fingerprints of length k

Anchor methods

We decided to work with seventy (N=70) 3-character anchors. Based on these anchors we have defined the following methods:

12. AFW - Anchor First Words - First 3-charcters from each one of the first N words in the tested document
13. AFS - Anchor First Sentences - First 3-charcters from each one of the first N sentences in the tested document
14. AF - most Frequent Anchors - N most frequent 3-charcter prefixes in the tested document
15. AR - Rarest Anchors - N rarest frequent 3-charcter prefixes in the tested document
16. ALW - Anchor Last Words - First 3-charcters from each one of the last N words in the tested document
17. ALS - Anchor Last Sentences - First 3-charcters from each one of the last N sentences in the tested document Word comparisons
18. CR - CompareReferences. This method compares between the titles of the papers included in the references section of the two examined papers.

Combinations of methods

19. CARA- CompareAbstractReferencesAverage. This method returns the average value of CA and CR.
20. CARM - CompareAbstractReferencesMin. This method returns the minimal value between CA and CR.

As mentioned above, Hoad and Zobel (2003) defined anchor methods based on the first/last N sentences/words/3-charcter prefixes in the tested document. As shown in Table 1 and in its analysis, the anchor methods are not successful, probably because they use a small portion of data. Therefore, we decided to implement methods defined for the following portions of the paper: the first third (*first*), the middle third (*middle*), and the last third (*end*) of the paper according to the number of the words in the discussed paper. All the *first*, *middle* and *end* methods use FF(3). These methods were combined with CA or CR. CA was not combined with the *first* methods because the abstract is included in the first part of the paper. CR was not combined with the *last* methods because the references are included in the end part of the paper.

21. CAMA- CompareAbstractMiddleAve. This method returns the average value of CA and FF(3) computed for the middle parts of the two examined papers.
22. CAMM - CompareAbstractMiddleMin. This method returns the minimal value between CA and FF(3) computed for the middle parts of the two examined papers.
23. CAEA - CompareAbstractEndAverage. This method returns the average value of CA and FF(3) computed for the end parts of the two examined papers.
24. CAEM - CompareAbstractEndMin. This method returns the minimal value between CA and FF(3) computed for the end parts of the two examined papers.
25. CRFA - CompareReferencesFirstAverage. This method returns the average value of CR and FF(3) computed for the first parts of the two examined papers.
26. CRFM - CompareReferencesFirstMin. This method returns the minimal value between between CR and FF(3) computed for the first parts of the two examined papers.
27. CRMA - CompareReferencesMiddleAverage. This method returns the average value of CR and FF(3) computed for the middle parts of the two examined papers.
28. CRMM - CompareReferencesMiddleMin. This method returns the minimal value between CR and FF(3) computed for the middle parts of the two examined papers.

To the best of our knowledge, we are the first to implement methods that compare special and important sections in academic papers: abstract and references: CA and CR, and combinations of them. In addition, we implemented new methods defined for the three thirds: the first (F) third, the middle (M) third, and the last (E) third of the paper. These methods were combined with CA and CR in various variants. All in total, we have defined 12 new baseline methods.

5 Experimental Results

5.1 Dataset

As mentioned above, the examined dataset includes 10,100 academic papers in computer science. Most of the papers are related to NLP and are from the last ten years. Most of the papers were downloaded from http://www.aclweb.org/anthology/.

These documents include 52,909,234 words that are contained in 3,722,766 sentences. Each document includes in average 5,262 words. The maximum number of words in a document is 28,758. The minimum number of words in a document is 305.

The original PDF files were downloaded using IDM - Internet Download Manager (http://www.internetdownloadmanager.com/). Then we convert them to TXT files using ghostscript (http://pages.cs.wisc.edu/~ghost/). Many PDF files were not papers and many others were converted to gibberish files. Therefore, the examined corpus contains only 10,100 papers.

5.2 Experiment I

Table 1 presents the results of the 38 implemented methods regarding the corpus of 10,100 documents. The test set includes 100 papers that were randomly chosen from the examined dataset. For each tested document, all the other 10,099 documents were compared using the various baseline methods.

The IDN, VHS, HS, MS columns present the number of the document pairs that found as identical, very high similar, high similar, and medium similar to the 100 tested documents, respectively. The IDN, VHS, HS, MS levels were granted to document pairs that got the following similarity values: 100%, [80%, 100%), [60%, 80%), and [40%, 60%), respectively.

The first left column indicates a simple ordinal number. The second left column indicates the serial number of the baseline method (Section 4) and the number in parentheses indicates the number of the chosen words (3 or 4) to be included in each substring.

On the one hand, the anchor methods (# 12-17) tried on the interval of 70-500 anchors report on relatively high numbers of suspicious document pairs, especially at the MS level. According to our expert, these high numbers are rather exaggerated. The reason for this finding might be that such fix numbers of anchors are not for detection of similar papers in various degrees of similarity.

#	#(k)	Method	IDN	VHS	HS	MS
1	1(3)	FF(3)	9	0	2	1
2	1(4)	FF(4)	9	0	1	1
3	2(3)	SSF(3)	0	0	0	9
4	2(4)	SSF(4)	0	0	0	9
5	3(3)	S2F(3)	9	0	2	2
6	3(4)	S2F(4)	9	0	1	1
7	4(3)	S3F(3)	0	0	9	0
8	4(4)	S3F(4)	0	0	9	0
9	5(3)	RDF(3)	1	5	1	3
10	5(4)	RDF(4)	1	6	0	3
11	6	CA	9	0	1	0
12	7(3)	SF(3)	9	0	0	1
13	7(4)	SF(4)	9	0	0	1
14	8(3)	SSS(3)	0	0	0	9
15	8(4)	SSS(4)	0	0	0	9
16	9(3)	S2S(3)	9	0	0	1
17	9(4)	S2S(4)	9	0	0	1
18	10(3)	S3S(3)	0	0	9	0
19	10(4)	S3S(4)	0	0	9	0
20	11(3)	RDS(3)	0	0	0	1
21	11(4)	RDS(4)	0	0	0	0
22	12	AFW	4	6	18	2772
23	13	AFS	6	3	10	708
24	14	AF	6	4	4	313
25	15	AR	4	6	19	2789
26	16	ALW	4	6	9	500
27	17	ALS	4	5	12	704
28	18	CR	9	0	1	3
29	19	CARA	8	2	1	0
30	20	CARM	8	0	2	0
31	21	CAMA	9	0	1	0
32	22	CAMM	9	0	0	1
33	23	CAEA	9	0	1	0
34	24	CAEM	9	0	0	1
35	25	CRFA	8	0	3	0
36	26	CRFM	8	0	2	0
37	27	CRMA	8	0	3	0
38	28	CRMM	8	0	1	1

Table 1. Results of the 38 implemented methods for 100 tested papers.

On the other hand, the SSF(k), S3F(k), S3S(k), RDF(k), and RDS(k) methods report on relatively very low numbers of suspicious document pairs. According to our expert, these numbers are too low. The reason for this finding might be that these methods are quite stringent for detection of similar document pairs.

The full fingerprint methods: FF(k), S2F(k) and the selective fingerprint methods SF(k), and S2S(k) present very similar results, which are reasonable according to our expert. Most of these methods report on 9 IDN, 0 VHS, 0-2 HS, and 1-2 MS document pairs. The full fingerprint methods report on slightly more HS and MS document pairs. According to our expert, these methods are regarded as the best.

Our novel methods: CA and CR also report on 9 IDN, 0 VHS, one HS, and 0 or 3 MS document pairs, respectively. The sum (10-13) of the IDN, VHS, HS and MS document pairs found by the best full and selective fingerprint methods mentioned in the last paragraph is the same sum of the IDN, VHS, HS and MS document pairs found by the CA and CR methods. That is, the CA and CR are very close in their quailty to the best methods. However, the CA and the CR have a clear advantage on the other methods. They check a rather small portion of the papers, and therfore their run time is much more smaller.

On the one hand, CR seems to be better than CA (and even the best selective fingerprint methods SF(k), and S2S(k)) because it reports on more MS document pairs, which means that CR is closer in its quality to the best full fingerprint methods. On the other hand, according to our expert CA is better than CR, since CR has more detection failures.

The combinations of CA and/or CR and/or the methods defined for the three thirds of the papers report on results that are less or equal from the viewpoint of their quality to CA or CR.

Several general conclusions can be drawn from the experimental results as follows:

(1) There are 9 documents (in the examined corpus) that are identical to one of the 100 tested papers. According to our expert, each one of these documents is IDN to a different paper from the 100 tested papers. This means that at least 9% of our random tested papers have IDN files in a corpus that contains 10, 099 files (for each test file).

(2) Several papers that have been found as IDN might be legal copies. For example: (a) by mistake, the same paper might be stored twice at the same conference website or (b) the paper, which is stored in its conference website might also be stored at its author's website.

(3) All the methods that run with two possible values of k (3 or 4 words) present similar results for the two values of k.

(4) FF(3) found as better than FF(4). FF(3) discovers 9 IDN papers, 2 HS papers, and 1 MS paper. These results were approved by a human expert. FF(4) missed one paper. One HS paper identified by FF(3) was identified as MS by FF(4) and one MS paper identified by FF(3) was identified as less than MS by FF(4). Moreover, also for other methods, variants with K=3 were better or equal to those with K=4. The main reason for these findings might be that the variants with K=4 check less substrings because the checks are done for each sentence. Substrings that end at the sequential sentence are not checked. Therefore, it is likely that additional equal substrings from the checked papers are not identified.

(5) S2F(3) discovers one more MS paper compared to FF(3). According to the human expert, the similarity measure of this paper should be less than MS. Therefore, we decided to select FF(3) as the best method.

(6) FF(3)'s run time is very high since it works on overlapping substrings for the whole papers.

(7) Our two novel methods: CA and CR are among the best methods for identification of various levels of plagiarism. As mentioned before, CA was found as a better predictor.

5.3 Selection of Methods and Experiment II

Sixteen methods out of the thirty-eight methods presented in Table 1, were selected for additional experiments. All the methods with k=4, the anchor methods, SSF, S3F, S3S, RDF, and RDS methods were omitted, due to their faulty results (as explained above). The remaining 16 methods (with k=3) are: FF, S2F, S2F, SF, S2S and all our 12 baseline methods: CA, and CR- CRMM.

Table 2 presents the results of these methods regarding the corpus of 10,100 documents. Since we selected less than half of the original methods we allow ourselves to test 1,000 documents instead of 100.

#	Method	IDN	VHS	HS	MS	Time d:h:m
1	FF	38	0	11	5	1:3:57.3
2	S2F	41	1	10	18	32:00.0
3	SF	37	1	1	6	31:12.2
4	S2	38	1	1	14	20:10.8
5	CA	38	1	11	5	09:16.7
6	CR	41	2	11	67	05:57.7
7	CARA	33	2	1	21	31:53.4
8	CARM	30	4	1	5	33:40.1
9	CAMA	38	0	5	6	11:26.5
10	CAMM	38	0	3	4	10:09.8
11	CAEA	38	0	6	7	10:42.1
12	CAEM	38	0	3	4	12:35.3
13	CRFA	32	1	3	25	54:20.7
14	CRFM	30	3	3	6	54:10.0
15	CRMA	33	2	3	25	58:52.2
16	CRMM	30	2	2	5	54:17.7

Table 2. Results of the 16 selected methods for 1,000 tested papers.

Again, according to our expert, FF has been found as the best predictive method. Surprisingly, CA achieved the second best results with one additional VHS paper. 11 HS documents and 5 MS documents have been identified by CA as by FF. The meaning of this finding is that the abstracts in almost all the simple similar documents were not significantly changed. That is, the authors of the non-IDN documents did not invest enough to change their abstracts.

CR indentified 41 documents as identical. The reason for this is probably because 3 additional papers have the same reference section as in 3 other tested papers, although these 3 document pairs are different in other sections. Furthermore, CR reports on relatively high number of suspicious document pairs, especially at the MS level. The meaning of this finding is that the references in many document pairs are not significantly different although these documents have larger differences in other sections. Consequently, combinations with CA achieved better results than combinations with CR.

A very important finding is that the run time of FF was very expensive (one day, 3 hours and 57.3 minutes) compared to the run time of CA (9 hours and 16.7 minutes). In other words, CA achieved almost the same results as FF but more efficiently.

5.4 An Error Analysis

The selected methods presented in Table 2 were analyzed according to the results of FF. Table 3 shows the distributions of false true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), regarding the 10,099 retrieved documents for the 1,000 tested document.

The false positive rate is the proportion in percents of positive test results (i.e., a plagiarism was identified by a baseline function) that are really negative values (i.e., the truth is that there is no plagiarism). The false negative rate is the proportion of negative test results that are really positive values.

#	Method	TP	FP	TN	FN
1	FF	0.534	0	99.465	0
2	S2F	0.524	0.168	99.296	0.010
3	SF	0.425	0.019	99.445	0.108
4	S2	0.435	0.099	99.366	0.099
5	CA	0.534	0.010	99.455	0
6	CR	0.534	0.663	98.801	0
7	CARA	0.386	0.178	99.287	0.148
8	CARM	0.356	0.039	99.425	0.178
9	CAMA	0.475	0	99.465	0.059
10	CAMM	0.445	0	99.465	0.089
11	CAEA	0.485	0.020	99.445	0.049
12	CAEM	0.445	0	99.465	0.089
13	CRFA	0.396	0.207	99.257	0.138
14	CRFM	0.376	0.039	99.425	0.158
15	CRMA	0.405	0.217	99.247	0.128
16	CRMM	0.366	0.020	99.445	0.168

Table 3. Distributions of the various possible statistical results.

FF is the only method that detects all cases of simple plagiarism. According to FF, there are 0.534% true positives. That is, 54 papers out of 10,099 are suspected as plagiarized versions of

54 papers of the 1,000 tested papers. This finding fits the results of FF(3) in Table 2, where there are 38 IDN, 11 HS, and 5 MS.

CA, the second best method has 0% false positives, and 0.01% false negatives, which means that CA identified one suspected plagiarized version that is really a non-plagiarized document. This finding is presented in Table 2, where CA identified 55 suspected plagiarized documents, one more than FF.

CR has 0% false positives, and 0.663% false negatives, which means that CR identified 67 suspected plagiarized versions that are really non-plagiarized documents. This finding is presented in Table 2, where CR identified 121 suspected plagiarized documents, 67 more than FF.

6 Illustrative Example

Due to space limitations, we briefly present an illustrative example of comparison between a couple of papers found as HS (High Similar) according to FF(3), the best detection method.

The tested paper (Snider and Diab, 2006A) contains 4 pages and it was published on June 06. The retrieved paper (Snider and Diab, 2006B) contains 8 pages and it was published a month later. The title of the tested paper is identical to the first eight words of the title of the retrieved paper. The authors of both papers are the same and their names appear in the same order. Most of the abstracts are the same. One of the main differences is the report of other results (probably updated results).

A relatively big portion of the beginning of the Introduction section in both papers is identical. Very similar sentences are found at the beginning of different sections (Section 2 in the 4-page paper and Section 3 in the the 8-page paper).

Many sentences or phrases from the rest of the papers are identical and some are very similar (e.g., addition of 'The' before "verbs are classified" in the abstract of the retrieved paper.

It is imoprtant to point that the authors in their 8-page paper wrote "This paper is an extension of our previous work in Snider and Diab (2006)". This sentence together with the detailed reference prove that the authors cite their previous work as required.

Concerning the references in both papers, at the first glance we found many differences between the two papers. The short paper contains only 7 references while the larger paper contains 14 references. However, a second closer look identifies that 5 out of the 7 references in the shorter paper are found in the reference section of the larger paper. Indeed, regarding the reference sections we did not find HS; but we have to remember that the larger paper include 8 pages twice than the shorter paper and therfore, more references could be included.

7 Conclusions and Future Work

To the best of our knowledge, we are the first to implement the CA and CR methods that compare two basic and important sections in academic papers: the abstract and references, respectively. In addition, we defined combinations of them. Furthermore, we implemented methods defined for the three thirds of the paper. These methods were combined with CA or CR in various variants. All in total, we have defined 12 new baseline methods.

Especially CA and also CR are among the best methods for identification of various levels of plagiarism. In contrast to the best full and selective fingerprint methods, CA and CR check a rather small portion of the papers, and therfore, their run time is much more smaller.

The success of CA and CR teaches us that most documents that are suspected as simple plagiarized papers include abstracts and references, which have not been significantly changed compared to other documents or vice versa.

There is a continuous need for automatic detection of plagiarism due to web influences, and advanced and more complex levels of plagiarism. Therefore, some possible future directions for research are: (1) Developing new kinds of selective fingerprint methods and new combinations of methods to improve detection, (2) Applying this research to larger and/or other corpora, and (3) Dealing with complex kinds of plagiarism, e.g., the use of synonyms, paraphrases, and transpositions of active sentences to passive sentences and vice versa.

References

Bernstein, Y., and Zobel, J., 2004. A Scalable System for Identifying Co-Derivative Documents. In *Proceedings of 11th International Conference on String Processing and Information Retrieval (SPIRE)*, vol. 3246, pp. 55-67.

Bretag, T., and Carapiet, S., 2007. A Preliminary Study to Identify the Extent of Self Plagiarism in Australian Academic Research. *Plagiary*, 2(5), pp. 1-12.

Collberg, C., Kobourov, S., Louie, J., and Slattery, T., 2005. Self-Plagiarism in Computer Science. *Communications of the ACM*, 48(4), pp. 88-94.

Heintze, N., 1996. Scalable Document Fingerprinting. In *Proceedings of the USENIX Workshop on Electronic Commerce*, Oakland California.

Hoad, T. C., and Zobel, J., 2003. Methods for Identifying Versioned and Plagiarised Documents. *Journal of the American Society for Information Science and Technology*, Vol 54(3), pp. 203-215.

IEEE, 2010. Introduction to the Guidelines for Handling Plagiarism Complaints. http://www.ieee.org/publications_standards/publications/rights/plagiarism.html.

Keuskamp, D., and Sliuzas, R., 2007. Plagiarism Prevention or Detection? The Contribution of Text-Matching Software to Education about Academic Integrity. *Journal of Academic Language and Learning*, Vol 1(1), pp. 91-99.

Library and Information Services, 2010. Cyprus University of Technology in Scopus, http://www.cut.ac.cy/library/english/services/references_en.html#plagiarism.

Loui, M. C., 2002. Seven Ways to Plagiarize: Handling Real Allegations of Research Misconduct. *Science and Engineering Ethics*, 8, pp. 529-539.

Lyon, C., Malcolm, J., and Dickerson, B., 2001. Detecting Short Passages of Similar Text in Large Document Collections. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pp. 118-125.

Manber, U., 1994. Finding Similar Files in a Large File System, In *Proceedings of the USENIX Technical Conference*, pp. 1-10.

Monostori1, K., Finkel, R., Zaslavsky, A., Hodasz, G., and Patke, M., 2002. Comparison of Overlap Detection Techniques. In *Proceedings of the 2002 International Conference on Computational Science*, Lecture Notes in Computer Science, vol 2329, pp. 51-60.

Shivakumar, N., and Garcia-Molina, H., 1996. Building a Scalable and Accurate Copy Detection Mechanism. In *Proceedings of the International Conference on Digital Libraries*, pp. 160-168.

Snider, N., and Diab, M., JUNE 2006A. Unsupervised Induction of Modern Standard Arabic Verb Classes. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, pp. 153- 156, June 2006.

Snider, N., and Diab, M., JULY 2006B. Unsupervised Induction of Modern Standard Arabic Verb Classes Using Syntactic Frames and LSA. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pp. 795- 802.

Sorokina, D., Gehrke, J., Warner, S., Ginsparg, P., 2006. Plagiarism Detection in arXiv. In *Proceedings of Sixth International Conference on Data Mining (ICDM)*, pp. 1070-1075.

Wikipedia, 2010. Plagiarism. http://en.wikipedia.org/wiki/Plagiarism.

Witten, I. H., Moffat, A., and Bell, T. C., 1999. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, second edition.

A Structured Vector Space Model for Hidden Attribute Meaning in Adjective-Noun Phrases

Matthias Hartung and **Anette Frank**
Computational Linguistics Department
Heidelberg University
{hartung, frank}@cl.uni-heidelberg.de

Abstract

We present an approach to model hidden attributes in the compositional semantics of adjective-noun phrases in a distributional model. For the representation of *adjective meanings*, we reformulate the pattern-based approach for attribute learning of Almuhareb (2006) in a structured vector space model (VSM). This model is complemented by a structured vector space representing attribute dimensions of *noun meanings*. The combination of these representations along the lines of compositional semantic principles exposes the underlying semantic relations in adjective-noun phrases. We show that our compositional VSM outperforms simple pattern-based approaches by circumventing their inherent sparsity problems.

1 Introduction

In formal semantic theory, the compositional semantics of adjective-noun phrases can be modeled in terms of *selective binding* (Pustejovsky, 1995), i.e. the adjective selects one of possibly several roles or attributes[1] from the semantics of the noun.

(1) a. a blue car
b. COLOR(car)=blue

In this paper, we define a distributional framework that models the compositional process underlying the modification of nouns by adjectives. We focus on property-denoting adjectives as they are valuable for acquiring concept representations for, e.g., ontology learning. An approach for automatic subclassification of property-denoting adjectives is presented in Hartung and Frank (2010). Our goal is to expose, for adjective-noun phrases as in (1a), the attribute in the semantics of the noun that is selected by the adjective, while not being overtly realized on the syntactic level. The semantic information we intend to capture for (1a) is formalized in (1b).

Ideally, this kind of knowledge could be extracted from corpora by searching for patterns that paraphrase (1a), e.g. *the color of the car is blue*. However, linguistic patterns that explicitly relate nouns, adjectives and attributes are very rare.

We avoid these sparsity issues by reducing the triple $r{=}\langle noun, attribute, adjective\rangle$ that encodes the relation illustrated in (1b) to tuples $r'{=}\langle noun, attribute\rangle$ and $r''{=}\langle attribute, adjective\rangle$, as suggested by Turney and Pantel (2010) for similar tasks. Both r' and r'' can be observed much more frequently in text corpora than r. Moreover, this enables us to model adjective and noun meanings as distinct semantic vectors that are built over attributes as dimensions. Based on these semantic representations, we make use of vector composition operations in order to reconstruct r from r' and r''. This, in turn, allows us to infer complete noun-attribute-adjective *triples* from individually acquired noun-attribute and adjective-attribute representations.

The contributions of our work are as follows: (i) We propose a framework for attribute selection based on structured vector space models (VSM), using as meaning dimensions attributes elicited

[1] In the original statement of the theory, adjectives select *qualia roles* that can be considered as collections of attributes.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 430–438,
Beijing, August 2010

by adjectives; (ii) we complement this novel representation of adjective meaning with structured vectors for *noun meanings* similarly built on attributes as meaning dimensions; (iii) we propose a composition of these representations that mirrors principles of compositional semantics in mapping adjective-noun phrases to their corresponding ontological representation; (iv) we propose and evaluate several metrics for the selection of meaningful components from vector representations.

2 Related Work

Adjective-noun meaning composition has not been addressed in a distributional framework before (cf. Mitchell and Lapata (2008)). Our approach leans on related work on attribute learning for ontology induction and recent work in distributional semantics.

Attribute learning. Early approaches to attribute learning include Hatzivassiloglou and McKeown (1993), who cluster adjectives that denote values of the same attribute. A weakness of their work is that the type of the attribute cannot be made explicit. More recent attempts to attribute learning from adjectives are Cimiano (2006) and Almuhareb (2006). Cimiano uses attributes as features to arrange sets of concepts in a lattice. His approach to attribute acquisition harnesses adjectives that occur frequently as concept modifiers in corpora. The association of adjectives with their potential attributes is performed by dictionary look-up in WordNet (Fellbaum, 1998). Similarly, Almuhareb (2006) uses adjectives and attributes as (independent) features for the purpose of concept learning. He acquires adjective-attribute pairs using a pattern-based approach.

As a major limitation, these approaches are confined to adjective-attribute pairs. The polysemy of adjectives that can only be resolved in the context of the modified noun is entirely neglected.

From a methodological point of view, our work is similar to Almuhareb's, as we will also build on lexico-syntactic patterns for attribute selection. However, we extend the task to involve nouns and rephrase his approach in a distributional framework based on the composition of structured vector representations.

Distributional semantics. We observe two recent trends in distributional semantics research: (i) The use of VSM tends to shift from measuring unfocused semantic similarity to capturing increasingly fine-grained semantic information by incorporating more linguistic structure. Following Baroni and Lenci (to appear), we refer to such models as *structured vector spaces*. (ii) Distributional methods are no longer confined to word meaning, but are noticeably extended to capture meaning on the *phrase level*. Prominent examples for (i) are Padó and Lapata (2007) and Rothenhäusler and Schütze (2009) who use syntactic dependencies rather than single word co-occurrences as dimensions of semantic spaces. Erk and Padó (2008) extend this idea to the argument structure of verbs, while also accounting for compositional meaning aspects by modelling predication over arguments. Hence, their work is also representative for (ii).

Baroni et al. (2010) use lexico-syntactic patterns to represent concepts in a structured VSM whose dimensions are interpretable as empirical manifestations of properties. We rely on similar techniques for the acquisition of structured vectors, whereas our work focusses on exposing the hidden meaning dimensions involved in compositional processes underlying concept modification.

The commonly adopted method for modelling compositionality in VSM is vector composition (Mitchell and Lapata, 2008; Widdows, 2008). Showing the benefits of vector composition for language modelling, Mitchell and Lapata (2009) emphasize its potential to become a standard method in NLP.

The approach pursued in this paper builds on both lines of research sketched in (i) and (ii) in that we model a specific meaning layer in the semantics of adjectives and nouns in a structured VSM. Vector composition is used to expose their hidden meaning dimensions on the phrase level.

3 Structured Vector Representations for Adjective-Noun Meaning

3.1 Motivation

Contrary to prior work, we model attribute selection as involving *triples* of nouns, attributes and

	COLOR	DIRECTION	DURATION	SHAPE	SIZE	SMELL	SPEED	TASTE	TEMPERATURE	WEIGHT
v_e	1	1	0	1	45	0	4	0	0	21
v_b	14	38	2	20	26	0	45	0	0	20
$v_e \times v_b$	14	38	0	20	**1170**	0	180	0	0	420
$v_e + v_b$	15	39	2	21	**71**	0	49	0	0	41

Figure 1: Vectors for *enormous* (v_e) and *ball* (v_b)

adjectives, as in (2). The triple r can be broken down into tuples $r' = \langle noun, attribute\rangle$ and $r'' = \langle attribute, adjective\rangle$. Previous learning approaches focussed on r' (Cimiano, 2006) or r'' (Almuhareb, 2006) only.

(2) a. a blue$_{\text{value}}$ car$_{\text{concept}}$
b. ATTR(concept) = value

In semantic composition of adjective-noun compounds, the adjective (e.g. *blue*) contributes a value for an attribute (here: COLOR) that characterizes the concept evoked by the noun (e.g. *car*). Thus, the attribute in (2) constitutes a 'hidden variable' that is not overtly expressed in (2a), but constitutes the central axis that relates r' and r''.

Structured vectors built on extraction patterns. We model the semantics of adjectives and nouns in a structured VSM that conveys the hidden relationship in (2). The dimensions of the model are defined by attributes, such as COLOR, SIZE or SPEED, while the vector components are determined on the basis of carefully selected acquisition patterns that are tailored to capturing the particular semantic information of interest for r' and r''. In this respect, lexico-syntactic patterns serve a similar purpose as dependency relations in Padó and Lapata (2007) or Rothenhäusler and Schütze (2009). The upper part of Fig. 1 displays examples of vectors we build for adjectives and nouns.

Composing vectors along hidden dimensions. The fine granularity of lexico-syntactic patterns that capture the triple r comes at the cost of their sparsity when applied to corpus data. Therefore, we construct separate vector representations for r' and r''. Eventually, these representations are joined by vector composition to reconstruct the triple r. Apart from avoiding sparsity issues, this compositional approach has several prospects from a linguistic perspective as well.

Ambiguity and disambiguation. Building vectors with attributes as meaning dimensions enables us to model (i) ambiguity of adjectives with regard to the attributes they select, and (ii) the disambiguation capacity of adjective and noun vectors when considered jointly. Consider, for example, the phrase *enormous ball* that is ambiguous for two reasons: *enormous* may select a set of possible attributes (SIZE or WEIGHT, among others), while *ball* elicits several attributes in accordance with its different word senses[2]. As seen in Fig. 1, these ambiguities are nicely captured by the separate vector representations for the adjective and the noun (upper part); by composing these representations, the ambiguity is resolved (lower part).

3.2 Building a VSM for Adjective-Noun Meaning

In this section, we introduce the methods we apply in order to (i) acquire vector representations for adjectives and nouns, (ii) select appropriate attributes from them, and (iii) compose them.

3.2.1 Attribute Acquisition Patterns

We use the following patterns[3] for the acquisition of vectors capturing the tuple $r'' = \langle attribute, adjective\rangle$. Even though some of these patterns (A1 and A4) match triples of nouns, attributes and adjectives, we only use them for the extraction of binary tuples (underlined), thus abstracting from the modified noun.

```
(A1)  ATTR of DT? NN is|was JJ
(A2)  DT? RB? JJ ATTR
(A3)  DT? JJ or JJ ATTR
(A4)  DT? NN's ATTR is|was JJ
(A5)  is|was|are|were JJ in|of ATTR
```

To acquire noun vectors capturing the tuple $r' = \langle noun, attribute\rangle$, we rely on the following patterns. Again, we only extract pairs, as indicated by the underlined elements.

```
(N1)  NN with|without DT? RB? JJ? ATTR
(N2)  DT ATTR of DT? RB? JJ? NN
(N3)  DT NN's RB? JJ? ATTR
(N4)  NN has|had a|an RB? JJ? ATTR
```

[2] WordNet senses for the noun *ball* include, among others: 1. *round object [...] in games*; 2. *solid projectile*, 3. *object with a spherical shape*, 4. *people [at a] dance*.

[3] Some of these patterns are taken from Almuhareb (2006) and Sowa (2000). The descriptions rely on the Penn Tagset (Marcus et al., 1999). ? marks optional elements.

3.2.2 Target Filtering

Some of the adjectives extracted by A1-A5 are not property-denoting and thus represent noise. This affects in particular pattern A2, which extracts adjectives like *former* or *more*, or relational ones such as *economic* or *geographic*.

This problem may be addressed in different ways: By *target filtering*, extractions can be checked against a predicative pattern P1 that is supposed to apply to property-denoting adjectives only. Vectors that fail this test are suppressed.

(P1) DT <u>NN</u> is|was <u>JJ</u>

Alternatively, extractions obtained from low-confidence patterns can be awarded reduced weights by means of a *pattern value function* (defined in 3.3; cf. Pantel and Pennacchiotti (2006)).

3.2.3 Attribute Selection

We intend to use the acquired vectors in order to detect attributes that are implicit in adjective-noun meaning. Therefore, we need a method that selects appropriate attributes from each vector. While, in general, this task consists in distinguishing semantically meaningful dimensions from noise, the requirements are different depending on whether attributes are to be selected from adjective or noun vectors. This is illustrated in Fig. 1, a typical configuration, with one vector representing a typical property-denoting adjective that exhibits relatively strong peaks on one or more dimensions, whereas noun vectors show a tendency for broad and flat distributions over their dimensions. This suggests using a strict selection function (choosing few very prominent dimensions) for adjectives and a less restrictive one (licensing the inclusion of more dimensions of lower relative prominence) for nouns. Moreover, we are interested in finding a selection function that relies on as few free parameters as possible in order to avoid frequency or dimensionality effects.

MPC Selection (MPC). An obvious method for attribute selection is to choose the most prominent component from any vector (i.e., the highest absolute value). If a vector exhibits several peaks, all other components are rejected, their relative importance notwithstanding. MPC obviously fails to capture polysemy of targets, which affects adjectives such as *hot*, in particular.

Threshold Selection (TSel). TSel recasts the approach of Almuhareb (2006), in selecting all dimensions as attributes whose components exceed a frequency threshold. This avoids the drawback of MPC, but introduces a parameter that needs to be optimized. Also, it is difficult to apply absolute thresholds to composed vectors, as the range of their components is subject to great variation, and it is unclear whether the method will scale with increased dimensionality.

Entropy Selection (ESel). In information theory, entropy measures the average uncertainty in a probability distribution (Manning and Schütze, 1999). We define the entropy $H(v)$ of a vector $v{=}\langle v_1, \ldots, v_n\rangle$ over its components as $H(v) = -\sum_{i=1}^{n} P(v_i) \log P(v_i)$, where $P(v_i) = v_i / \sum_{i=1}^{n} v_i$.

We use $H(v)$ to assess the impact of singular vector components on the overall entropy of the vector: We expect entropy to detect components that contribute noise, as opposed to those that contribute important information.

We define an algorithm for entropy-based attribute selection that returns a list of informative dimensions. The algorithm successively suppresses (combinations of) vector components one by one. Given that a gain of entropy is equivalent to a loss of information and vice versa, we assume that every combination of components that leads to an increase in entropy when being suppressed is actually responsible for a substantial amount of information. The algorithm includes a back-off to MPC for the special case that a vector contains a single peak (i.e., $H(v) = 0$), so that, in principle, it should be applicable to vectors of any kind. Vectors with very broad distributions over their dimensions, however, pose a problem to this method. For *ball* in Fig. 1, for instance, the method does not select any dimension.

Median Selection (MSel). As a further method we rely on the median m that can be informally defined as the value that separates the upper from the lower half of a distribution (Krengel, 2003). It is less restrictive than MPC and TSel and overcomes the particular drawback of ESel. Using this measure, we choose all dimensions whose components exceed m. Thus, for the vector representing

Pattern Label	# Hits (Web)	# Hits (ukWaC)
A1	2249	815
A2	36282	72737
A3	3370	1436
A4	–	7672
A5	–	3768
N1	–	682
N2	–	5073
N3	–	953
N4	–	56

Table 1: Number of pattern hits on the Web (Almuhareb, 2006) and on ukWaC

ball, WEIGHT, DIRECTION, SHAPE, SPEED and SIZE are selected.

3.2.4 Vector Composition

We use vector composition as a hinge to combine adjective and noun vectors in order to reconstruct the triple $r{=}\langle noun, attribute, adjective\rangle$. Mitchell and Lapata (2008) distinguish two major classes of vector composition operations, namely multiplicative and additive operations, that can be extended in various ways. We use their standard definitions (denoted $\times$ and $+$, henceforth). For our task, we expect $\times$ to perform best as it comes closest to the linguistic function of *intersective* adjectives, i.e. to select dimensions that are prominent both for the adjective and the noun, whereas $+$ basically blurs the vector components, as can be seen in the lower part of Fig. 1.

3.3 Model Parameters

We follow Padó and Lapata (2007) in defining a semantic space as a matrix $M = B \times T$ relating a set of target elements T to a set of basis elements B. Further parameters and their instantiations we use in our model are described below. We use p to denote an individual lexico-syntactic pattern.

The **basis elements** of our VSM are nouns denoting attributes. For comparison, we use the attributes selected by Almuhareb (2006): COLOR, DIRECTION, DURATION, SHAPE, SIZE, SMELL, SPEED, TASTE, TEMPERATURE, WEIGHT.

The **context selection function** *cont(t)* determines the set of patterns that contribute to the representation of each target word $t \in T$. These are the patterns A1-A5 and N1-N4 (cf. Section 3.2.1).

The **target elements** represented in the vector space comprise all adjectives T_A that match the patterns A1 to A5 in the corpus, provided they exceed a frequency threshold n. During development, n was set to 5 in order to filter noise.

As for the target nouns T_N, we rely on a representative dataset compiled by Almuhareb (2006). It contains 402 nouns that are balanced with regard to semantic class (according to the WordNet supersenses), ambiguity and frequency.

As **association measure** that captures the strength of the association between the elements of B and T, we use raw frequency counts[4] as obtained from the PoS-tagged and lemmatized version of the ukWaC corpus (Baroni et al., 2009). Table 1 gives an overview of the number of hits returned by these patterns.

The **basis mapping function** μ creates the dimensions of the semantic space by mapping each extraction of a pattern p to the attribute it contains.

The **pattern value function** enables us to subdivide dimensions along particular patterns. We experimented with two instantiations: pv_{const} considers, for each dimension, all patterns, while weighting them equally. $pv_f(p)$ awards the extractions of pattern p with weight 1, while setting the weights for all patterns different from p to 0.

4 Experiments

We evaluate the performance of the structured VSM on the task of inferring attributes from adjective-noun phrases in three experiments: In Exp1 and Exp2, we evaluate vector representations capturing r' and r'' independently of one another. Exp3 investigates the selection of hidden attributes from vector representations constructed by composition of adjective and noun vectors.

We compare all results against different *gold standards*. In Exp1, we follow Almuhareb (2006), evaluating against WordNet 3.0. For Exp2 and Exp3, we establish gold standards manually: For Exp2, we construct a test set of nouns annotated with their corresponding attributes. For Exp3, we manually annotate adjective-noun phrases with the attributes appropriate for the whole phrase. All experiments are evaluated in terms of precision, recall and F_1 score.

[4] We experimented with the conditional probability ratio proposed by Mitchell and Lapata (2009). As it performed worse on our data, we did not consider it any further.

4.1 Exp1: Attribute Selection for Adjectives

The first experiment evaluates the performance of structured vector representations on attribute selection for adjectives. We compare this model against a re-implementation of Almuhareb (2006).

Experimental settings and gold standard. To reconstruct Almuhareb's approach, we ran his patterns `A1`-`A3` on the ukWaC corpus. Table 1 shows the number of hits when applied to the Web (Almuhareb, 2006) vs. ukWaC. `A1` and `A3` yield less extractions on ukWaC as compared to the Web.[5] We introduced two additional patterns, `A4` and `A5`, that contribute about 10,000 additional hits. We adopted Almuhareb's manually chosen thresholds for attribute selection for `A1`-`A3`; for `A4`, `A5` and a combination of all patterns, we manually selected optimal thresholds.

We experiment with pv_{const} and all variants of $pv_f(p)$ for pattern weighting (see sect. 3.3). For attribute selection, we compare TSel (as used by Almuhareb), ESel and MSel.

The gold standard consists of all adjectives that are linked to at least one of the ten attributes we consider by WordNet's `attribute` relation (1063 adjectives in total).

Evaluation results. Results for Exp1 are displayed in Table 2. The settings of pv are given in the rows, the attribute selection methods (in combination with target filtering[6]) in the columns.

The results for our re-implementation of Almuhareb's individual patterns are comparable to his original figures[7], except for `A3` that seems to suffer from quantitative differences of the underlying data. Combining all patterns leads to an improvement in precision over (our reconstruction of) Almuhareb's best individual pattern when TSel and target filtering are used in combination. MPC and MSel perform worse (not reported here). As for target filtering, `A1` and `A3` work best.

Both TSel and ESel benefit from the combination with the target filter, where the largest improvement (and the best overall result) is observable for ESel on pattern `A1` only. This is the pattern that performs worst in Almuhareb's original setting. From this, we conclude that both ESel and target filtering are valuable extensions to pattern-based structured vector spaces if precision is in focus. This also underlines a finding of Rothenhäusler and Schütze (2009) that VSMs intended to convey specific semantic information rather than mere similarity benefit primarily from a linguistically adequate choice of contexts.

	MPC			**ESel**			**MSel**		
	P	R	F	P	R	F	P	R	F
pv_f(`N1`)	0.22	0.06	0.10	**0.29**	0.04	0.07	0.22	**0.09**	**0.13**
pv_f(`N2`)	**0.29**	0.18	0.23	0.20	0.06	0.09	0.28	**0.39**	**0.33**
pv_f(`N3`)	**0.34**	0.05	0.09	0.20	0.02	0.04	0.25	**0.08**	**0.12**
pv_f(`N4`)	0.25	0.02	0.04	**0.29**	0.02	0.03	0.26	0.02	**0.05**
pv_{const}	**0.29**	0.18	0.22	0.20	0.06	0.09	0.28	**0.43**	**0.34**

Table 3: Evaluation results for Experiment 2

Similar to Almuhareb, recall is problematic. Even though ESel leads to slight improvements, the scores are far from satisfying. With Almuhareb, we note that this is mainly due to a high number of extremely fine-grained adjectives in WordNet that are rare in corpora.[8]

4.2 Exp2: Attribute Selection for Nouns

Exp2 evaluates the performance of attribute selection from noun vectors tailored to the tuple r''.

Construction of the gold standard. For evaluation, we created a gold standard by manually annotating a set of nouns with attributes. This gold standard builds on a random sample extracted from T_N (cf. section 3.3). Running `N1`-`N4` on ukWaC returned semantic vectors for 216 concepts. From these, we randomly sampled 100 concepts that were manually annotated by three human annotators.

The annotators were provided a matrix consisting of the nouns and the set of ten attributes for each noun. Their task was to remove all inappropriate attributes. They were free to decide how many attributes to accept for each noun. In order to deal with word sense ambiguity, the annotators were instructed to consider all senses of a noun and to retain every attribute that was acceptable for at least one sense.

Inter-annotator agreement amounts to κ= 0.69 (Fleiss, 1971). Cases of disagreement were adjudicated by majority-voting. The gold standard

[5] The difference for `A2` is an artifact of Almuhareb's extraction methodology.

[6] Regarding target filtering, we only report the best filter pattern for each configuration.

[7] P(`A1`)=0.176, P(`A2`)=0.218, P(`A3`)=0.504

	Almuhareb (reconstr.)				VSM (TSel + Target Filter)					VSM (ESel)			VSM (ESel + Target Filter)			
	P	R	F	Thr	P	R	F	Patt	Thr	P	R	F	P	R	F	Patt
$pv_f(\texttt{A1}) = 1$	0.183	0.005	0.009	5	0.300	0.004	0.007	A3	5	0.231	**0.045**	**0.076**	**0.519**	0.035	0.065	A3
$pv_f(\texttt{A2}) = 1$	0.207	0.039	0.067	50	**0.300**	0.033	0.059	A1	50	0.084	**0.136**	**0.104**	0.240	0.049	0.081	A3
$pv_f(\texttt{A3}) = 1$	0.382	0.020	0.039	5	**0.403**	0.014	0.028	A1	5	0.192	**0.059**	**0.090**	0.375	0.027	0.050	A1
$pv_f(\texttt{A4}) = 1$					**0.301**	0.020	0.036	A3	10	0.135	**0.055**	**0.078**	0.272	0.020	0.038	A1
$pv_f(\texttt{A5}) = 1$					0.295	0.008	0.016	A3	24	0.105	**0.056**	**0.073**	**0.315**	0.024	0.045	A3
pv_{const}					**0.420**	0.024	0.046	A1	183	0.076	**0.152**	**0.102**	0.225	0.054	0.087	A3

Table 2: Evaluation results for Experiment 1

contains 424 attributes for 100 nouns.

Evaluation results. Results for Exp2 are given in Table 3. Performance is lower in comparison to Exp1. We hypothesize that the tuple r'' might not be fully captured by overt linguistic patterns. This needs further investigation in future research.

Against this background, MPC is relatively precise, but poor in terms of recall. ESel, being designed to select more than one prominent dimension, counterintuitively fails to increase recall, suffering from the fact that many noun vectors show a rather flat distribution without any strong peak. MSel turns out to be most suitable for this task: Its precision is comparable to MPC (with N3 as an outlier), while recall is considerably higher. Overall, these results indicate that attribute selection for adjectives and nouns, though similar, should be viewed as distinct tasks that require different attribute selection methods.

4.3 Exp3: Attribute Selection for Adjective-Noun Phrases

In this experiment, we compose noun and adjective vectors in order to yield a new combined representation. We investigate whether the semantic information encoded by the components of this new vector is sufficiently precise to disambiguate the attribute dimensions of the original representations (see section 3.1) and, thus, to infer hidden attributes from adjective-noun phrases (see (2)) as advocated by Pustejovsky (1995).

Construction of the gold standard. For evaluation, we created a manually annotated test set of adjective-noun phrases. We selected a subset of property-denoting adjectives that are appropriate modifiers for the nouns from T_N using the predicative pattern P1 (see sect. 3) on ukWaC. This yielded 2085 adjective types that were further reduced to 386 by frequency filtering ($n = 5$). We sampled our test set from all pairs in the cartesian product of the 386 adjectives and 216 nouns (cf. Exp2) that occurred at least 5 times in a subsection of ukWaC. To ensure a sufficient number of ambiguous adjectives in the test set, sampling proceeded in two steps: First, we sampled four nouns each for a manual selection of 15 adjectives of all ambiguity levels in WordNet. This leads to 60 adjective-noun pairs. Second, another 40 pairs were sampled fully automatically.

The test set was manually annotated by the same annotators as in Exp2. They were asked to remove all attributes that were not appropriate for a given adjective-noun pair, either because it is not appropriate for the noun or because it is not selected by the adjective. Further instructions were as in Exp2, in particular regarding ambiguity.

The overall agreement is κ=0.67. After adjudication by majority voting, the resulting gold standard contains 86 attributes for 76 pairs. 24 pairs could not be assigned any attribute, either because the adjective did not denote a property, as in *private investment*, or the most appropriate attribute was not offered, as in *blue day* or *new house*.

We evaluate the vector composition methods discussed in section 3.2.4. Individual vectors for the adjectives and nouns from the test pairs were constructed using all patterns A1-A5 and N1-N4. For attribute selection, we tested MPC, ESel and MSel. The results are compared against three baselines: BL-P implements a purely pattern-based method, i.e. running the patterns that extract the triple r (A1, A4, N1, N3 and N4, with JJ and NN instantiated accordingly) on the pairs from the test set. BL-N and BL-Adj are back-offs for vector composition, taking the respective noun or adjective vector, as investigated in Exp1 and Exp2, as surrogates for a composed vector.

[8] For instance: *bluish-lilac*, *chartreuse* or *pink-lavender* as values of the attribute COLOR.

	MPC			**ESel**			**MSel**		
	P	R	F	P	R	F	P	R	F
×	0.60	0.58	**0.59**	**0.63**	0.46	0.54	0.27	0.72	0.39
+	0.43	0.55	0.48	0.42	0.51	0.46	0.18	**0.91**	0.30
BL-Adj	0.44	0.60	0.50	0.51	0.63	0.57	0.23	0.83	0.36
BL-N	0.27	0.35	0.31	0.37	0.29	0.32	0.17	0.73	0.27
BL-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 4: Evaluation results for Experiment 3

Evaluation results. Results are given in Table 4. Attribute selection based on the composition of adjective and noun vectors yields a considerable improvement of both precision and recall as compared to the individual results obtained in Exp1 and Exp2. Comparing the results of Exp3 against the baselines reveals two important aspects of our work. First, the complete failure of BL-P[9] underlines the attractiveness of our method to build structured vector representations from patterns of reduced complexity. Second, vector composition is suitable for selecting hidden attributes from adjective-noun phrases that are jointly encoded by adjective and noun vectors: Both composition methods we tested outperform BL-N.

However, the choice of the composition method matters: × performs best with a maximum precision of 0.63. This confirms our expectation that vector multiplication is a good approximation for attribute selection in adjective-noun semantics. Being outperformed by BL-Adj in most categories, + is less suited for this task.

All selection methods outperform BL-Adj in precision. Comparing MPC and ESel, ESel achieves better precision when combined with the ×-operator, while doing worse for recall. The robust performance of MPC is not surprising as the test set contains only ten adjective-noun pairs that are still ambiguous with regard to the attributes they elicit. The stronger performance of the entropy-based method with the ×-operator is mainly due to its accuracy on detecting false positives, in that it is able to return ”empty” selections. In terms of precision, MSel did worse in general, while recall is decent. This underlines that vector composition generally promotes meaningful components, but MSel is too inaccurate to select them.

Given the performance of the baselines and the noun vectors in Exp2, we consider this a very promising result for our approach to attribute selection from structured vector representations. The results also corroborate the insufficiency of previous approaches to attribute learning from adjectives alone.

5 Conclusions and Outlook

We proposed a structured VSM as a framework for inferring hidden attributes from the compositional semantics of adjective-noun phrases.

By reconstructing Almuhareb (2006), we showed that structured vector representations of adjective meaning consistently outperform simple pattern-based learning, up to 13 pp. in precision. A combination of target filtering and pattern weighting turned out to be effective here, by selecting particulary meaningful lexico-syntactic contexts and filtering adjectives that are not property-denoting. Further studies need to investigate this phenomenon and its most appropriate formulation in a vector space framework.

Moreover, the VSM offers a natural representation for sense ambiguity of adjectives. Comparing attribute selection methods on adjective and noun vectors shows that they are sensitive to the distributional structure of the vectors, and need to be chosen with care. Future work will investigate these selection methods in high-dimensional vectors spaces, by using larger sets of attributes.

Exp3 shows that the composition of pattern-based adjective and noun vectors robustly reflects aspects of meaning composition in adjective-noun phrases, with attributes as a hidden dimension. It also suggests that composition is effective in disambiguation of adjective and noun meanings. This hypothesis needs to be substantiated in further experiments.

Finally, we showed that composition of vectors representing complementary meaning aspects can be beneficial to overcome sparsity effects. However, our compositional approach meets its limits if the patterns capturing adjective and noun meaning in isolation are too sparse to acquire sufficiently populated vector components from corpora. For future work, we envisage using vector similarity to acquire structured vectors for infrequent targets from semantic spaces that convey less linguistic structure to address these remaining sparsity issues.

[9] The patterns used yield no hits for the test pairs at all.

References

Almuhareb, Abdulrahman. 2006. *Attributes in Lexical Acquisition*. Ph.D. Dissertation, Department of Computer Science, University of Essex.

Baroni, Marco and Alessandro Lenci. to appear. Distributional Memory. A General Framework for Corpus-based Semantics. *Computational Linguistics*.

Baroni, Marco, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: A collection of very large linguistically processed web-crawled corpora. *Journal of Language Resources and Evaluation*, 43(3):209–226.

Baroni, Marco, Brian Murphy, Eduard Barbu, and Massimo Poesio. 2010. Strudel. A Corpus-based Semantic Model of Based on Properties and Types. *Cognitive Science*, 34:222–254.

Cimiano, Philipp. 2006. *Ontology Learning and Population from Text. Algorithms, Evaluation and Applications*. Springer.

Erk, Katrin and Sebastian Padó. 2008. A Structured Vector Space Model for Word Meaning in Context. In *Proceedings of EMNLP*, Honolulu, HI.

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

Fleiss, Joseph L. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Hartung, Matthias and Anette Frank. 2010. A Semi-supervised Type-based Classification of Adjectives. Distinguishing Properties and Relations. In *Proceedings of the 7th International Conference on Language Resources and Evaluation,* Valletta, Malta, May.

Hatzivassiloglou, Vasileios and Kathleen McKeown. 1993. Towards the Automatic Identification of Adjectival Scales. Clustering Adjectives According to Meaning. In *Proceedings of the 31st Annual Meeting of the Association of Computational Linguistics*, pages 172–182.

Krengel, Ulrich. 2003. *Wahrscheinlichkeitstheorie und Statistik*. Vieweg, Wiesbaden.

Manning, Christopher D. and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Marcus, Mitchell P., Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. 1999. Treebank-3, ldc99t42. CD-ROM. Philadelphia, Penn.: Linguistic Data Consortium.

Mitchell, Jeff and Mirella Lapata. 2008. Vector-based Models of Semantic Composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, June.

Mitchell, Jeff and Mirella Lapata. 2009. Language Models Based on Semantic Composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, Singapore, August 2009, pages 430–439, Singapore, August.

Padó, Sebastian and Mirella Lapata. 2007. Dependency-based Construction of Semantic Space Models. *Computational Linguistics*, 33:161–199.

Pantel, Patrick and Marco Pennacchiotti. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, 17–21 July 2006, pages 113–120.

Pustejovsky, James. 1995. *The Generative Lexicon*. MIT Press, Cambridge, Mass.

Rothenhäusler, Klaus and Hinrich Schütze. 2009. Unsupervised Classification with Dependency Based Word Spaces. In *Proceedings of the EACL Workshop on Geometrical Models of Natural Language Semantics (GEMS)*, pages 17–24, Athens, Greece, March.

Sowa, John F. 2000. *Knowledge Representation. Logical, Philosophical, and Computational Foundations*. Brooks Cole.

Turney, Peter D. and Patrick Pantel. 2010. From Frequency to Meaning. Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Widdows, Dominic. 2008. Semantic Vector Products. Some Initial Investigations. In *Proceedings of the 2nd Conference on Quantum Interaction*, Oxford, UK, March.

Hierarchical Phrase-based Machine Translation with Word-based Reordering Model

Katsuhiko Hayashi*, Hajime Tsukada**
Katsuhito Sudoh, Kevin Duh**, Seiichi Yamamoto***
*Doshisha University
katsuhiko-h@is.naist.jp, seyamamo@mail.doshisha.ac.jp
**NTT Communication Science Laboratories
tsukada, sudoh, kevinduh@cslab.kecl.ntt.co.jp

Abstract

Hierarchical phrase-based machine translation can capture global reordering with synchronous context-free grammar, but has little ability to evaluate the correctness of word orderings during decoding. We propose a method to integrate word-based reordering model into hierarchical phrase-based machine translation to overcome this weakness. Our approach extends the synchronous context-free grammar rules of hierarchical phrase-based model to include reordered source strings, allowing efficient calculation of reordering model scores during decoding. Our experimental results on Japanese-to-English basic travel expression corpus showed that the BLEU scores obtained by our proposed system were better than those obtained by a standard hierarchical phrase-based machine translation system.

1 Introduction

Hierarchical phrase-based machine translation (Chiang, 2007; Watanabe et al., 2006) is one of the promising statistical machine translation approaches (Brown et al., 1993). Its model is formulated by a synchronous context-free grammar (SCFG) which captures the syntactic information between source and target languages. Although the model captures global reordering by SCFG, it does not explicitly introduce reordering model to constrain word order. In contrast, lexicalized reordering models (Tillman, 2004; Koehn et al., 2005; Nagata et al., 2006) are extensively used for phrase-based translation. These lexicalized reordering models cannot be directly applied to hierarchical phrased-based translation since the hierarchical phrase representation uses nonterminal symbols.

To handle global reordering in phrase-based translation, various preprocessing approaches have been proposed, where the source sentence is reordered to target language order beforehand (Xia and McCord, 2004; Collins et al., 2005; Li et al., 2007; Tromble and Eisner, 2009). However, preprocessing approaches cannot utilize other information in the translation model and target language model, which has been proven helpful in decoding.

This paper proposes a method that incorporates word-based reordering model into hierarchical phrase-based translation to constrain word order. In this paper, we adopt the reordering model originally proposed by Tromble and Eisner (2009) for the preprocessing approach in phrase-based translation. To integrate the word-based reordering model, we added a reordered source string into the right-hand-side of SCFG's rules. By this extension, our system can generate the reordered source sentence as well as target sentence and is able to efficiently calculate the score of the reordering model. Our method utilizes the translation model and target language model as well as the reordering model during decoding. This is an advantage of our method over the preprocessing approach.

The remainder of this paper is organized as follows. Section 2 describes the concept of our approach. Section 3 briefly reviews our proposed method on hierarchical phrase-based ma-

Standard SCFG	$X \rightarrow< X1$ wa jinsei no $X2$ da , $X1$ is $X2$ of life$>$
SCFG (move-to-front)	$X \rightarrow< X1$ wa jinsei no $X2$ da , wa $X1$ da $X2$ no jinsei , $X1$ is $X2$ of life$>$
SCFG (attach)	$X \rightarrow< X1$ wa jinsei no $X2$ da , $X1$ wa da $X2$ no jinsei , $X1$ is $X2$ of life$>$

Table 1: A Japanese-to-English example of various SCFG's rule representations. Japanese words are romanized. Our proposed representation of rules has reordered source string to generate reordered source sentence $S^{'}$ as well as target sentence T. The "move-to-front" means Tromble and Eisner (2009) 's algorithm and the "attach" means Al-Onaizan and Papineni (2006) 's algorithm.

chine translation model. We experimentally compare our proposed system to a standard hierarchical phrase-based system on Japanese-to-English translation task in Section 4. Then we discuss on related work in Section 5 and conclude this paper in Section 6.

2 The Concept of Our Approach

The preprocessing approach (Xia and McCord, 2004; Collins et al., 2005; Li et al., 2007; Tromble and Eisner, 2009) splits translation procedure into two stages:

$$S \rightarrow S^{'} \rightarrow T \tag{1}$$

where S is a source sentence, $S^{'}$ is a reordered source sentence with respect to the word order of target sentence T. Preprocessing approach has the very deterministic and hard decision in reordering. To overcome the problem, Li et al. (2007) proposed k-best appoach. However, even with a k-best approach, it is difficult to generate good hypotheses $S^{'}$ by using only a reordering model.

In this paper, we directly integrated the reordering model into the decoder in order to use the reordering model together with other information in the hierarchical phrase-based translation model and target language model. Our approach is expressed as the following equation.

$$S \rightarrow (S^{'}, T). \tag{2}$$

Our proposed method generates the reordered source sentence $S^{'}$ by SCFG and evaluates the correctness of the reorderings using a word-based reordering model of S' which will be introduced in section 3.4.

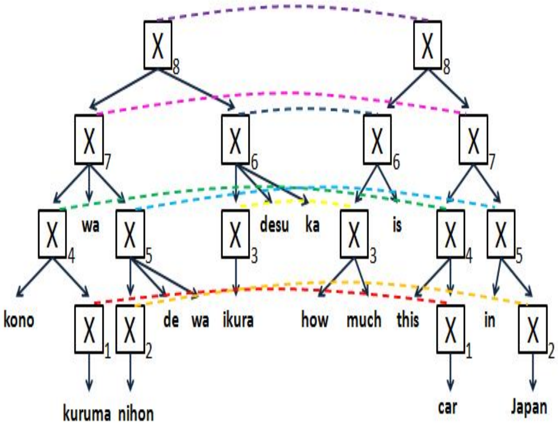


Figure 1: A derivation tree for Japanse-to-English translation.

3 Hierarchical Phrase-based Model Extension

3.1 Hierarchical Phrase-based Model

Hierarchical phrase-based model (Chiang, 2007) induces rules of the form

$$X \rightarrow< \gamma, \alpha, \sim, w > \tag{3}$$

where X is a non-terminal symbol, γ is a sequence string of non-terminals and source terminals, α is a sequence string of non-terminals and target terminals. $\sim$ is a one-to-one correspondence for the non-terminals appeared in γ and α.

Given a source sentence S, the translation task under this model can be expressed as

$$\hat{T} = T\left(\underset{D:S(D)=S}{\operatorname{argmax}}\ w(D)\right) \tag{4}$$

where D is a derivation and $w(D)$ is a score of the derivation. Decoder seeks a target sentence

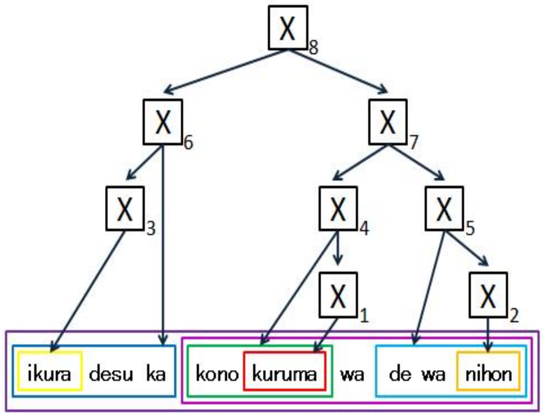


Figure 2: Reordered source sentence generated by our proposed system.

$T(D)$ which has the highest score $w(D)$. $S(D)$ is a source sentence under a derivation D. Figure 1 shows the example of Japanese-to-English translation by hierarchical phrase-based machine translation model.

3.2 Rule Extension

To generate reordered source sentence $S^{'}$ as well as target sentence T, we extend hierarchical phrase rule expressed in Equation 3 to

$$X \rightarrow < \gamma, \gamma^{'}, \alpha, \sim, w > \tag{5}$$

where $\gamma^{'}$ is a sequence string of non-terminals and source terminals, which is reordered γ with respect to the word order of target string α. The reason why we add $\gamma^{'}$ to rules is to efficiently calculate the reordering model scores. If each rule does not have $\gamma^{'}$, the decoder need to keep word alignments because we cannot know word order of $S^{'}$ without them. The calculation of reordering model scores using word alignments is very wasteful when decoding.

The translation task under our model extends Equation 4 to the following equation:

$$\hat{T} = (\hat{S}^{'}, \hat{T}) = (S^{'}, T)\left(\underset{D:S(D)=S}{\operatorname{argmax}}\ w(D)\right). \tag{6}$$

Our system generates the reordered source sentence $S^{'}$ as well as target sentence T. Figure 2 shows the generated reordered source sentence $S^{'}$ when translating the example of Figure 1. Note that the structure of $S^{'}$ is the same as that of target sentence T. The decoder generates both Figure 2 and the right hand side of Figure 1, allowing us to score both global and local word reorderings.

Uni-gram Features
s_r, $s\text{-}pos_r$
s_r
$s\text{-}pos_r$
s_l, $s\text{-}pos_l$
s_l
$s\text{-}pos_l$

Bi-gram Features
s_r, $s\text{-}pos_r$, s_l, $s\text{-}pos_l$
$s\text{-}pos_r$, s_l, $s\text{-}pos_l$
s_r, s_l, $s\text{-}pos_l$
s_r, $s\text{-}pos_r$, $s\text{-}pos_l$
s_r, $s\text{-}pos_r$, s_l
s_r, s_l
$s\text{-}pos_r$, $s\text{-}pos_l$

Table 2: Features used by Word-based Reordering Model. pos means part-of-speech tag.

To add $\gamma^{'}$ to rules, we permuted γ into $\gamma^{'}$ after rule extraction based on Grow-diag-final (Koehn et al., 2005) alignment by GIZA++ (Och and Ney, 2003). To do this permutation on rules, we applied two methods. One is the same algorithm as Tromble and Eisner (2009), which reorders aligned source terminals and nonterminals in the same order as that of target side and moves unaligned source terminals to the front of aligned terminals or nonterminals (move-to-front). The other is the same algorithm as AI-Onaizan and Papineni (2006), which differs from Tromble and Eisner's approach in attaching unaligned source terminals to the closest prealigned source terminals or nonterminals (attach). This extension of adding $\gamma^{'}$ does not increase the number of rules.

Table 1 shows a Japanese-to-English example of the representation of rules for our proposed system. Japanese words are romanized. Suppose that source-side string is (X1 wa jinsei no X2 da) and target-side string is (X1 is X2 of life) and their word alignments are a=((jinsei , life) , (no , of) , (da , is)). Source-side aligned words and nonterminal symbols are sorted into the same order of target string. Source-side unaligned word (wa) is moved to the front or right of the prealigned symbol (X1).

Surrounding Word Pos Features
$s\text{-}pos_r$, $s\text{-}pos_r + 1$, $s\text{-}pos_l - 1$, $s\text{-}pos_l$
$s\text{-}pos_r - 1$, $s\text{-}pos_r$, $s\text{-}pos_l - 1$, $s\text{-}pos_l$
$s\text{-}pos_r$, $s\text{-}pos_r + 1$, $s\text{-}pos_l$, $s\text{-}pos_l + 1$
$s\text{-}pos_r - 1$, $s\text{-}pos_r$, $s\text{-}pos_l$, $s\text{-}pos_l + 1$

Table 3: The Example of Context Features

3.3 Word-based Reordering Model

We utilize the following $score(S^{'})$ as a feature for the word-based reordering model. This is incorporated into the log-linear model (Och and Ney, 2002) of statistical machine translation.

$$score(S^{'}) = \sum_{i,j:1 \leq i < j \leq n} B[s_i^{'}, s_j^{'}] \qquad (7)$$

$$B[s_l^{'}, s_r^{'}] = \theta \cdot \phi(s_l^{'}, s_r^{'}) \qquad (8)$$

where n is the length of reordered source sentence $S^{'}$ $(= (s_1^{'} \dots s_n^{'}))$, θ is a weight vector and ϕ is a vector of features. This reordering model, which is originally proposed by Tromble and Eisner (2009), can assign a score to any possible permutation of source sentences. Intuitively $B[s_l^{'}, s_r^{'}]$ represents the score of ordering $s_l^{'}$ before $s_r^{'}$; the higher the value, the more we prefer word $s_l^{'}$ occurs before $s_r^{'}$. Whether $S_l^{'}$ should occur before $S_r^{'}$ depends on how often this reordering occurs when we reorder the source to target sentence order.

To train B, we used binary feature functions ϕ as used in (Tromble and Eisner, 2009), which were introduced for dependency parsing by McDonald et al. (2005). Table 2 shows the kind of features we used in our experiments. We did not use context features like surrounding word pos features in Table 3 because they were not useful in our preliminary experiments and propose an efficient implementation described in the next section in order to calculate this reordering model when decoding. To train the parameter θ, we used the perceptron algorithm following Tromble and Eisner (2009).

3.4 Integration to Cube Pruning

CKY parsing and cube-pruning are used for decoding of hierarchical phrase-based model (Chiang, 2007). Figure 3 displays that hierarchical phrase-based decoder seeks new span [1,7] items with rules, utilizing subspan [1,3] items and subspan [4,7] items. In this example, we use 2-gram language model and +LM decoding. uni(•) means 1-gram language model cost for heuristics and interaction usually means language model cost that cannot be calculated offline. Here, we introduce our two implementations to calculate word-based reordering model scores in this decoding algorithm.

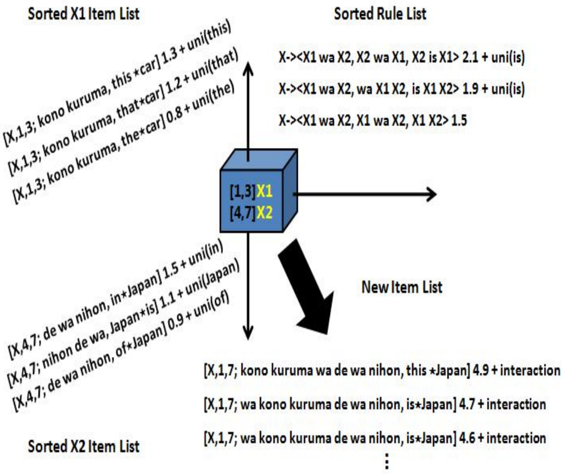

Sorted X1 Item List
Sorted Rule List
[X,1,3; kono kuruma, this *car] 1.3 + uni(this)
[X,1,3; kono kuruma, that*car] 1.2 + uni(that)
[X,1,3; kono kuruma, the*car] 0.8 + uni(the)
X-><X1 wa X2, X2 wa X1, X2 is X1> 2.1 + uni(is)
X-><X1 wa X2, wa X1 X2, is X1 X2> 1.9 + uni(is)
X-><X1 wa X2, X1 wa X2, X1 X2> 1.5
[1,3] X1
[4,7] X2
New Item List
[X,4,7; de wa nihon, in*Japan] 1.5 + uni(in)
[X,4,7; nihon de wa, Japan*is] 1.1 + uni(Japan)
[X,4,7; de wa nihon, of*Japan] 0.9 + uni(of)
[X,1,7; kono kuruma wa de wa nihon, this *Japan] 4.9 + interaction
[X,1,7; wa kono kuruma de wa nihon, is*Japan] 4.7 + interaction
[X,1,7; wa kono kuruma de wa nihon, is*Japan] 4.6 + interaction
⋮
Sorted X2 Item List

Figure 3: Creating new items from subitems and rules, that have a span [1,7] in source sentence.

First, we explain a naive implementation shown in the left side of Figure 4. This algorithm performs the same calculation of reordering model as that of language model. Each item keeps a part of reordered source sentence. The reordering score of new item can be calculated as interaction cost when combining subitems with the rule.

The right side of Figure 4 shows our proposed implementation. This implementation can be adopted to decoding only when we do not use context features like surrounding word pos features in Table 3 (and consider a distance between words in features). If a span is given, the reordering scores of new item can be calculated for each rule, being independent from the word order of reordered source segment of a subitem. So, the reordering model scores can be calculated for all rules with spans by using a part of the input source sentence before sorting them for cube pruning. We expect this sorting of rules with reordering

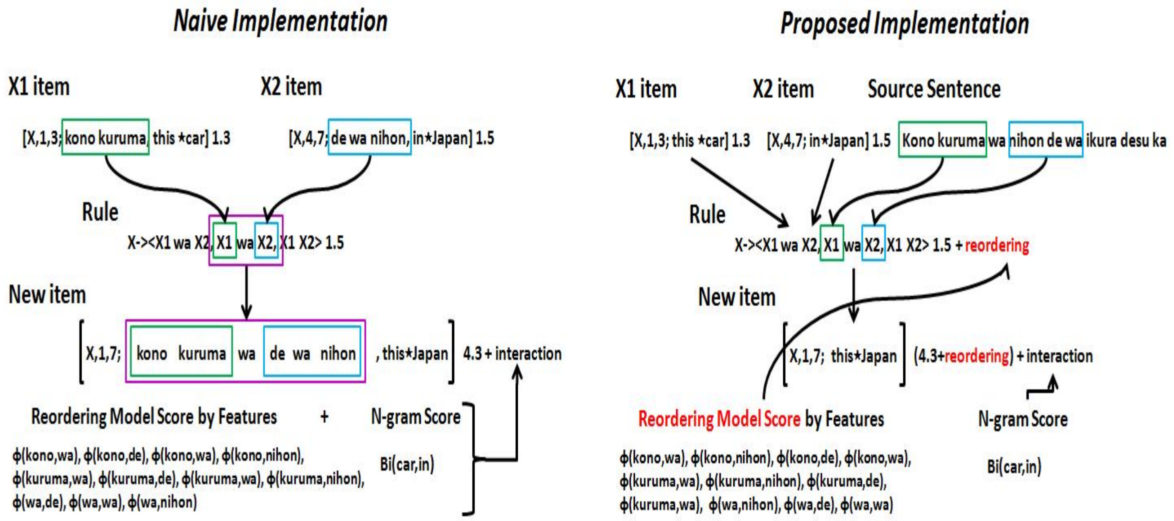

Naive Implementation
X1 item
X2 item
[X,1,3; kono kuruma, this ★car] 1.3
[X,4,7; de wa nihon, in★Japan] 1.5
Rule
X-><X1 wa X2, X1 wa X2, X1 X2> 1.5
New item
X,1,7; kono kuruma wa de wa nihon , this★Japan
4.3 + interaction
Reordering Model Score by Features
+
N-gram Score
ϕ(kono,wa), ϕ(kono,de), ϕ(kono,wa), ϕ(kono,nihon),
ϕ(kuruma,wa), ϕ(kuruma,de), ϕ(kuruma,wa), ϕ(kuruma,nihon),
ϕ(wa,de), ϕ(wa,wa), ϕ(wa,nihon)
Bi(car,in)
Proposed Implementation
X1 item
X2 item
Source Sentence
[X,1,3; this ★car] 1.3
[X,4,7; in★Japan] 1.5
Kono kuruma wa nihon de wa ikura desu ka
Rule
X-><X1 wa X2, X1 wa X2, X1 X2> 1.5 + reordering
New item
X,1,7; this★Japan
(4.3+reordering) + interaction
Reordering Model Score by Features
N-gram Score
ϕ(kono,wa), ϕ(kono,nihon), ϕ(kono,de), ϕ(kono,wa),
ϕ(kuruma,wa), ϕ(kuruma,nihon), ϕ(kuruma,de),
ϕ(kuruma,wa), ϕ(wa,nihon), ϕ(wa,de), ϕ(wa,wa)
Bi(car,in)

Figure 4: The "naive" and "proposed" implementation to calculate the reordering cost of new items.

model scores will have good influence on cube pruning. The right hand side of Figure 4 shows the diffrence between naive and proposed implementation (S' is not shown to allow for a clear presentation). Note the difference is in where/when the reordering scores are inserted: together with the N-gram scores in the case of naive implementation; incorpolated into sorted rules for the proposed implementation.

4 Experiment

4.1 Purpose

To reveal the effectiveness of integrating the reordering model into decoder, we compared the following setups:

- baseline: a standard hierarchical phrase-based machine translation (Hiero) system.
- preprocessing: applied Tromble and Eisner's approach, then translate by Hiero system.
- Hiero system + reordering model: integrated reordering model into Hiero system.

We used the Joshua Decoder (Li and Khudanpur, 2008) as the baseline Hiero system. This decoder uses a log-linear model with seven features, which consist of N-gram language model $P_{LM}(T)$, lexical translation model $P_w(\gamma|\alpha)$, $P_w(\alpha|\gamma)$, rule translation model $P(\gamma|\alpha)$, $P(\alpha|\gamma)$, word penalty and arity penalty.

The "Hiero + Reordering model" system has word-based reordering model as an additional feature to baseline features. For this approach, we use two systems. One has "move-to-front" system and the other is "attach" system explained in Section 3.2. We implemented our proposed algorithm in Section 3.4 to both "Hiero + Reordering model" systems. As for beam width, we use the same setups for each system.

4.2 Data Set

Data		Sent.	Word.	Avg. leng
Training	ja	200.8K	2.4M	12.0
	en	200.8K	2.3M	11.5
Development	ja	1.0K	10.3K	10.3
	en	1.0K	9.8K	9.8
Test	ja	1.0K	14.2K	14.2
	en	1.0K	13.5K	13.5

Table 4: The Data statistics

For experiments we used a Japanese-English basic travel expression corpus (BTEC). Japanese word order is linguistically very different from English and we think Japanese-English pair is a very good test bed for evaluating reordering model.

Metrics System	BLEU	PER
Baseline (Hiero)	28.09	39.68
Preprocessing	17.32	45.27
Hiero + move-to-front	**28.85**	39.89
Hiero + attach	**29.25**	**39.43**

Table 5: BLEU and PER scores on the test set.

Our training corpus contains about 200.8k sentences. Using the training corpus, we extracted hierarchical phrase rules and trained 4-gram language model and word-based reordering model. Parameters were tuned over 1.0k sentences (development data) with single reference by minimum error rate training (MERT) (Och, 2003). Test data consisted of 1.0k sentences with single reference. Table 4 shows the condition of corpus in detail.

4.3 Results

Table 5 shows the BLEU (Papineni et al., 2001) and PER (Niesen et al., 2000) scores obtained by each system. The results clearly indicated that our proposed system with word-based reordering model (move-to-front or attach) outperformed baseline system on BLEU scores. In contrast, there is no significant improvement from baseline on PER. This suggests that the improvement of BLEU mainly comes from reordering. In our experiment, preprocessing approach resulted in very poor scores.

4.4 Discussion

Table 6 displays examples showing the cause of the improvements of our system with reordering model (attach) comparing to baseline system. We can see that the outputs of our system are more fluent than those of baseline system because of reordering model.

As a further analysis, we calculated the BLEU scores of Japanese $S^{'}$ predicted from reordering model against true Japanese $S^{'}$ made from GIZA++ alignments, were only 26.2 points on development data. We think the poorness mainly comes from unaligned words since they are untractable for the word-based reordering model. Actually, Japanese sentences in our training data include 34.7% unaligned words. In spite of the poorness, our proposed method effectively utilize this reordering model in contrast to preprocessing approach.

5 Related Work

Our approach is similar to preprocessing approach (Xia and McCord, 2004; Collins et al., 2005; Li et al., 2007; Tromble and Eisner, 2009) in that it reorders source sentence in target order. The difference is this sentence reordering is done in decoding rather than in preprocessing.

A lot of studies on lexicalized reordering (Tillman, 2004; Koehn et al., 2005; Nagata et al., 2006) focus on the phrase-based model. These works cannnot be directly applied to hierarchical phrase-based model because of the difference between normal phrases and hierarchical phrases that includes nonterminal symbols.

Shen et al. (2008,2009) proposed a way to integrate dependency structure into target and source side string on hierarchical phrase rules. This approach is similar to our approach in extending the formalism of rules on hierarchical phrase-based model in order to consider the constraint of word order. But, our approach differs from (Shen et al., 2008; Shen et al., 2009) in that syntax annotation is not necessary.

6 Conclusion and Future Work

We proposed a method to integrate word-based reordering model into hierarchical phrase-based machine translation system. We add $\gamma^{'}$ into the hiero rules, but this does not increase the number of rules. So, this extension itself does not affect the search space of decoding. In this paper we used Tromble and Eisner's reordering model for our method, but various reordering model can be incorporated to our method, for example $S^{'}$ N-gram language model. Our experimental results on Japanese-to-English task showed that our system outperformed baseline system and preprocessing approach.

In this paper we utilize $\gamma^{'}$ only for reordering model. However, it is possible to use $\gamma^{'}$ for other modeling, for example we can use it for rule translation probabilities $P(\gamma^{'}|\gamma)$, $P(\gamma|\gamma^{'})$ for additional feature functions. Of course, we can

S	america de seihin no hanbai wo hajimeru keikaku ga ari masu ka .	kono tegami wa koukuubin de nihon made ikura kakari masu ka .
T_B	sales of product in america are you planning to start ?	this letter by airmail to japan . how much is it ?
T_P	are you planning to start products in the u.s. ?	how much does it cost to this letter by airmail to japan ?
R	do you plan to begin selling your products in the u.s. ?	how much will it cost to send this letter by air mail to japan ?

Table 6: Examples of outputs for input sentence S from baseline system T_B and our proposed system (attach) T_P. R is a reference. The underlined portions have equivalent meanings and show the reordering differences.

also utilize reordered target sentence $T^{'}$ for various modeling as well. Addtionally we plan to use $S^{'}$ for MERT because we hypothesize the fluent $S^{'}$ leads to fluent T.

References

AI-Onaizan, Y. and K. Papineni. 2006. Distortion models for statistical machine translation. In *Proc. the 44th ACL*, pages 529–536.

Brown, P. F., S. A. D. Pietra, V. D. J. Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguitics*, 19:263–312.

Chiang, D., K. Knight, and W. Wang. 2009. 11,001 new features for statistical machine translation. In *Proc. NAACL*, pages 216–226.

Chiang, D. 2007. Hierachical phrase-based translation. *Computational Linguitics*, 33:201–228.

Collins, M., P. Koehn, and I. Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proc. the 43th ACL*, pages 531–540.

Collins, M. 2002. Discriminative training methods for hidden markov models. In *Proc. of EMNLP*.

Freund, Y. and R. E. Schapire. 1996. Experiments with a new boosting algorithm. In *Proc. of the 13th ICML*, pages 148–156.

Koehn, P., A. Axelrod, A-B. Mayne, C. Callison-Burch, M. Osborne, and D. Talbot. 2005. Edinburgh system description for 2005 iwslt speech translation evaluation. In *Proc. the 2nd IWSLT*.

Li, Z. and S. Khudanpur. 2008. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proc. ACL SSST*.

Li, C-H., D. Zhang, M. Li, M. Zhou, K. Li, and Y. Guan. 2007. A probabilistic approach to syntax-based reordering for statistical machine translation. In *Proc. the 45th ACL*, pages 720–727.

McDonald, R., K. Crammer, and F. Pereira. 2005. Spanning tree methods for discriminative training of dependency parsers. In *Thechnical Report MS-CIS-05-11, UPenn CIS*.

Nagata, M., K. Saito, K. Yamamoto, and K. Ohashi. 2006. A clustered global phrase reordering model for statistical machine translation. In *COLING-ACL*, pages 713–720.

Niesen, S., F.J. Och, G. Leusch, and H. Ney. 2000. An evaluation tool for machine translation: Fast evaluation for mt research. In *Proc. the 2nd International Conference on Language Resources and Evaluation*.

Och, F. J. and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. the 40th ACL*, pages 295–302.

Och, F. and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29:19–51.

Och, F. J. 2003. Minimum error rate training in statistical machine translation. In *Proc. the 41th ACL*, pages 160–167.

Papineni, K. A., S. Roukos, T. Ward, and W-J. Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *Proc. the 39th ACL*, pages 311–318.

Shen, L., J. Xu, and R. Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. ACL*, pages 577–585.

Shen, L., J. Xu, B. Zhang, S. Matsoukas, and R. Weischedel. 2009. Effective use of linguistic and contextual information for statistical machine translation. In *Proc. EMNLP*, pages 72–80.

Tillman, C. 2004. A unigram orientation model for statistical machine translation. In *Proc. HLT-NAACL*, pages 101–104.

Tromble, R. and J. Eisner. 2009. Learning linear ordering problems for better translation. In *Proc. EMNLP*, pages 1007–1016.

Watanabe, T., H. Tsukada, and H. Isozaki. 2006. Left-to-right target generation for hierarchical phrase-based translation. In *Proc. COLING-ACL*, pages 777–784.

Xia, F. and M. McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *Proc. the 18th ICON*, pages 508–514.

A Novel Reordering Model Based on Multi-layer Phrase for Statistical Machine Translation

Yanqing He[1], **Yu Zhou**[2], **Chengqing Zong**[2], **Huilin Wang**[1]

[1]Institute of Scientific and Technical Information of China
{heyq,wanghl}@istic.ac.cn

[2]Institute of Automation, Chinese Academy of Sciences
{yzhou,cqzong}@nlpr.ia.ac.cn

Abstract

Phrase reordering is of great importance for statistical machine translation. According to the movement of phrase translation, the pattern of phrase reordering can be divided into three classes: monotone, BTG (Bracket Transduction Grammar) and hierarchy. It is a good way to use different styles of reordering models to reorder different phrases according to the characteristics of both the reordering models and phrases itself. In this paper a novel reordering model based on multi-layer phrase (PRML) is proposed, where the source sentence is segmented into different layers of phrases on which different reordering models are applied to get the final translation. This model has some advantages: different styles of phrase reordering models are easily incorporated together; when a complicated reordering model is employed, it can be limited in a smaller scope and replaced with an easier reordering model in larger scope. So this model better trade-offs the translation speed and performance simultaneously.

1 Introduction

In statistical machine translation (SMT), phrase reordering is a complicated problem. According to the type of phrases, the existing phrase reordering models are divided into two categories: contiguous phrase-based reordering models and non-contiguous phrase-based reordering models.

Contiguous phrase-based reordering models are designed to reorder contiguous phrases. In such type of reordering models, a contiguous phrase is reordered as a unit and the movements of phrase don't involve insertions inside the other phrases. Some of these models are content-independent, such as distortion models (Och and Ney, 2004; Koehn et al., 2003) which penalize translation according to jump distance of phrases, and flat reordering model (Wu, 1995; Zens et al., 2004)which assigns constant probabilities for monotone order and non-monotone order. These reordering models are simple and the contents of phrases have not been considered. So it's hard to obtain a satisfactory translation performance. Some lexicalized reordering models (Och et al., 2004; Tillmann 2004, Kumar and Byrne, 2005, Koehn et al., 2005) learn local orientations (monotone or non-monotone) with probabilities for each bilingual phrase from training data. These models are phrase-dependent, so improvements over content-independent reordering models are obtained. However, many parameters need to be estimated.

Non-contiguous phrase-based reordering models are proposed to process non-contiguous phrases and the movements of phrase involve insertion operations. This type of reordering models mainly includes all kinds of syntax-based models where more structural information is employed to obtain a more flexible phrase movement. Linguistically syntactic approaches (Yamada and Knight, 2001; Galley et al., 2004, 2006; Marcu et al., 2006; Liu et al., 2006; Shieber et al., 1990; Eisner, 2003; Quirk et al., 2005; Ding and Palmer, 2005) employ linguistically syntactic information to enhance their reordering capability and use non-contiguous phrases to

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 447–455,
Beijing, August 2010

obtain some generalization. The formally syntax-based models use synchronous context-free grammar (SCFG) but induce a grammar from a parallel text without relying on any linguistic annotations or assumptions (Chiang, 2005; Xiong et al., 2006). A hierarchical phrase-based translation model (HPTM) reorganizes phrases into hierarchical ones by reducing sub-phrases to variables (Chiang 2005). Xiong et al. (2006) is an enhanced bracket transduction grammar with a maximum entropy-based reordering model (MEBTG). Compared with contiguous phrase-based reordering model, Syntax-based models need to shoulder a great deal of rules and have high computational cost of time and space. The type of reordering models has a weaker ability of processing long sentences and large-scale data, which heavily restrict their application.

The above methods have provided various phrases reordering strategies. According to the movement of phrase translation, the pattern of phrase reordering can be divided into three classes: monotone, BTG (Bracket Transduction Grammar) (Wu, 1995) and hierarchy. In fact for most sentences, there may be some phrases which have simple reordering patterns, such as monotone or BTG style. It is not necessary to reorder them with a complicated mechanism, e.g. hierarchy. It is a good idea that different reordering models are employed to reorder different phrases according to the characteristics of both the reordering models and the phrases itself. This paper thus gives a novel reordering model based on multi-layer phrase (PRML), where the source sentence is segmented into different layers of phrases on which different reordering models are applied to get the final translation. Our model has the advantages as follow: (1) PRML segments source sentence into multiple-layer phrases by using punctuation and syntactic information and the design of segmentation algorithm corresponds to each reordering model. Different reordering models are chosen for each layer of phrases. (2) In our model different reordering models can be easily integrated together to obtain a combination of multiple phrase reordering models. (3) Our model can incorporate some complicated reordering models. We limit them in relatively smaller scopes and replace them with easier reordering models in larger scopes. In such way our model better trade-offs the translation speed and performance simultaneously. (4) Our segmentation strategy doesn't impair translation quality by controlling phrase translation tables to determine the scope of each reordering model in each source sentence. The poor phrase translations generated by the former reordering model, still have chances of being revised by the latter reordering model.

Our work is similar to the phrase-level system combination (Mellebeek et al., 2006). We share one important characteristic: we decompose input sentence into chunks and recompose the translated chunks in output. The differences are that, we segment the input sentence into multi-layer phrases and we reorder their translations with a multi-layer decoder.

The remainder of the paper is organized as follows: Section 2 gives our reordering model PRML. Section 3 presents the details of the sentence segmentation algorithm and the decoding algorithm. Section 4 shows the experimental results. Finally, the concluding remarks are given in Section 5.

2 The Model

We use an example to demonstrate our motivation. Figure 1 shows a Chinese and English sentence pair with word alignment. Each solid line denotes the corresponding relation between a Chinese word and an English word. Figure 2 shows our reordering mechanism. For the source sentence, the phrases in rectangle with round corner in row 2 obviously have a monotone translation order. For such kinds of phrase a monotone reordering model is enough to arrange their translations. Any two neighbor consecutive phrases in the ellipses in row 3 have a straight orders or inverted order. So BTG reordering model is appropriate to predict the order of this type of phrases. Inside the phrases in the ellipses in row 3 there are possibly more complicated hierarchical structures. For the phrase "*通往 和平 之 路*", a rule "$\mathrm{X}\rightarrow\langle$通往 X_1 之路, $\textit{towards the road to } X_1\rangle$" has the ascendancy over the monotone and BTG style of reordering model. Hierarchy style of reordering models, such as HPTM reordering model, can translate non-contiguous phrases and has the advantage of capturing the translation of such kind of phrases.

The whole frame of our model PRML is shown in Figure 3. PRML is composed of a segmentation sentence module and a decoder which consists of three different styles of phrase reordering models. The source sentence is segmented into 3 layers of phrases: the original whole sentence, sub-sentences and chunks. The original whole sentence is considered as the first-layer phrase and is segmented into sub-sentences to get the second-layer phrase. By further segmenting these sub-sentences, the chunks are obtained as the third-layer phrase. The whole translation process includes three steps: **1)** In order to capture the most complicated structure of phrases inside chunks, HPTM reordering model are chosen to translate the chunks. So the translations of chunks are obtained. 2) Combine the bilingual chunks generated by step 1 with those bilingual phases generated by the MEBTG training model as the final phrase table and translate the sub-sentences with MEBTG reordering model, the translations of sub-sentences are obtained. 3) Combine the bilingual sub-sentences generated by step 2 with those bilingual phases generated by the Monotone training model as the final phrase table and translate the original whole sentences with monotone reorder-

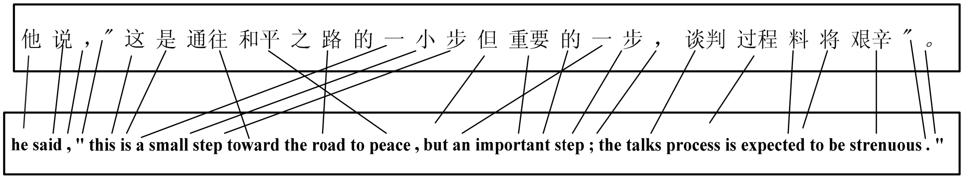

他 说 ，″ 这 是 通往 和平 之 路 的 一 小 步 但 重要 的 一 步 ， 谈判 过程 料 将 艰辛 ″ 。
he said , " this is a small step toward the road to peace , but an important step ; the talks process is expected to be strenuous . "

Figure 1. An example of Chinese-English sentence pair with their word alignment

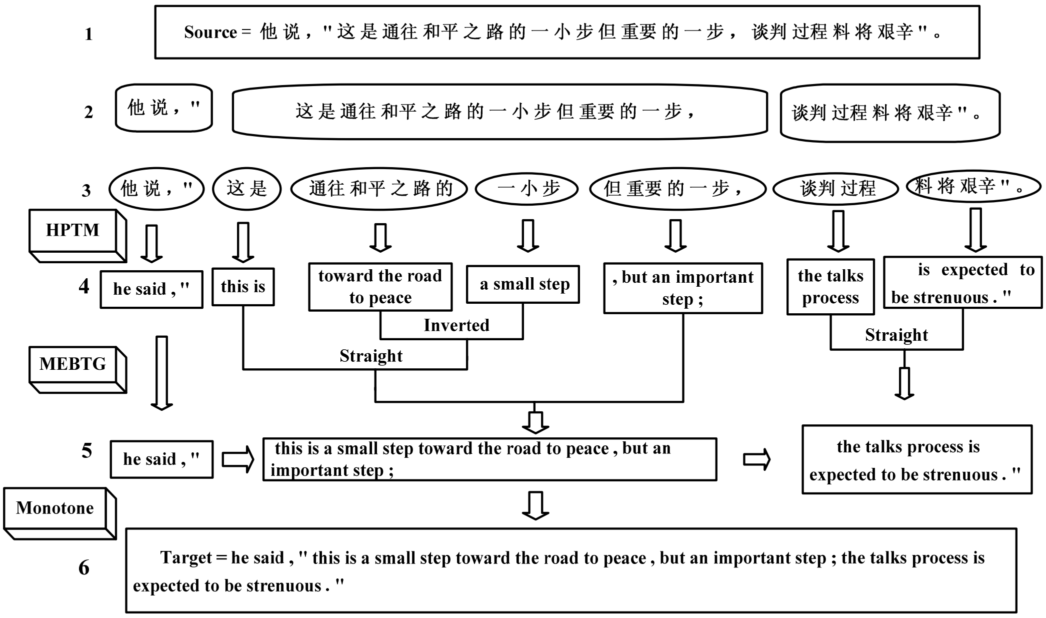

1
Source = 他 说 ，" 这 是 通往 和平 之 路 的 一 小 步 但 重要 的 一 步 ， 谈判 过程 料 将 艰辛 " 。
2
他 说 ，"
这 是 通往 和平 之 路 的 一 小 步 但 重要 的 一 步 ，
谈判 过程 料 将 艰辛 " 。
3
他 说 ，"
这 是
通往 和平 之 路 的
一 小 步
但 重要 的 一 步 ，
谈判 过程
料 将 艰辛 " 。
HPTM
4
he said , "
this is
toward the road to peace
a small step
, but an important step ;
the talks process
is expected to be strenuous . "
Inverted
Straight
Straight
MEBTG
5
he said , "
this is a small step toward the road to peace , but an important step ;
the talks process is expected to be strenuous . "
Monotone
6
Target = he said , " this is a small step toward the road to peace , but an important step ; the talks process is expected to be strenuous . "

Figure 2. Diagram of Translation Using PRML.

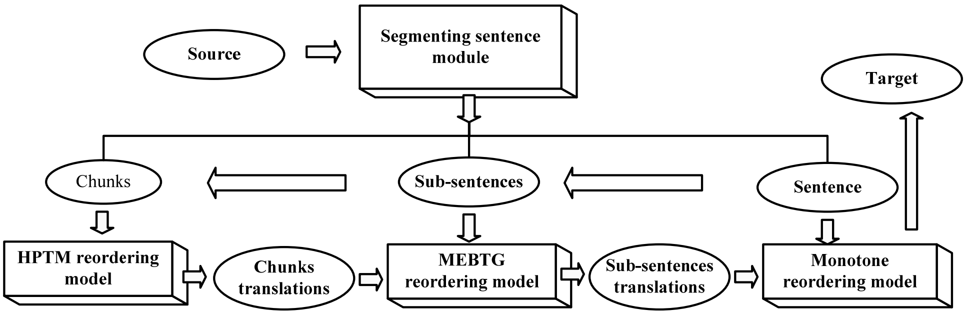


Figure 3. Frame of PRML

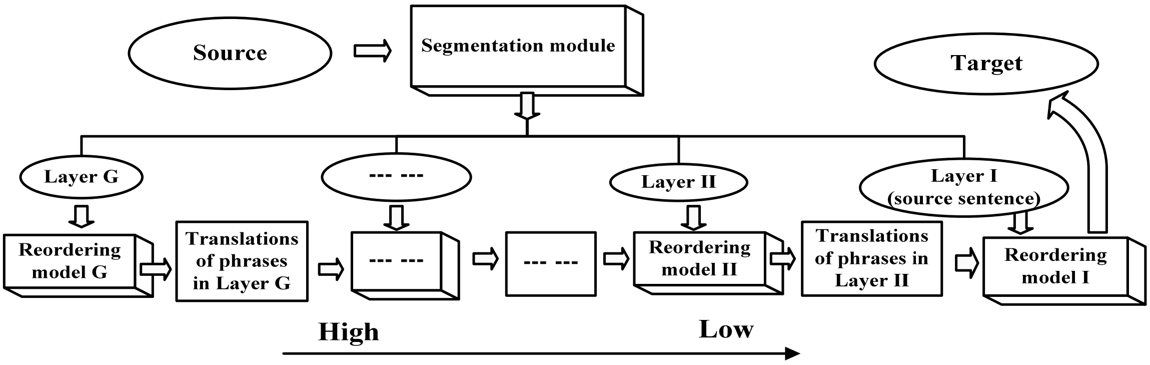


Figure 4. General frame of our model

ing model, the translations of the original whole sentences are obtained.

We also give a general frame of our model in Figure 4. In the segmentation module, an input source sentence is segmented into *G* layers of contiguous source strings, Layer 1, Layer 2, …, Layer *G*. The phrases of lower-order layer are re-segmented into the phrases of higher-order layer. The phrases of the same layer can be combined into the whole source sentence. The decoding process starts from the phrases of the highest-order layer. For each layer of phrases a reordering model is chosen to generate the translations of phrases according to their characteristics. The generated translations of phrases in the higher-order layer are fed as a new added translation source into the next lower-order reordering model. After the translations of the phrase in Layer 2 are obtained, they are fed into the Reordering model 1 as well as the source sentence (the phrase in Layer 1) to get the target translation.

Due to the complexity of the language, there may be some sentences whose structures don't conform to the pattern of the reordering models we choose. So in our segmentation module, if the sentence doesn't satisfy the segmentation conditions of current layer, it will be fed into the segmentation algorithm of the next layer. Even in the worst condition when the sentence isn't segmented into any phrase by segmentation module, it will be translated as the whole sentence to get the final translation by the highest-order reordering model.

Our model tries to grasp firstly the simple reordering modes in source sentence by the lower layer of phrase segmentations and controls more complicated reordering modes inside the higher layers of phrases. Then we choose some complicated reordering models to translate those phrases. Thus search space and computational complexity are both reduced. After obtaining the translation of higher layer's phrases, it is enough for simple reordering models to reorder them. Due to phrase segmentation some phrases may be translated poorly by the higher layer of reordering models, but they still have chances of being revised by the lower layer of reordering model because in lower layer of reordering model the input phrases have not these hard segmentation boundary and our model uses phrase translation tables to determine the scope of each reordering model.

There are two key issues in our model. The first one is how to segment the source sentence into different layers of phrases. The second one is how to choose a reordering model for different layer of phrases. In any case the design of segmenting sentence module should consider the characteristic of the reordering model of phrases.

3 Implementation

The segmentation module consists of the sub-sentence segmentation and chunk segmentation. The decoder combines three reordering models, HPTM, MEBTG, and a monotone reordering model.

3.1 Segmentation module

We define the sub-sentence as the word sequence which can be translated in monotone order. The following six punctuations: 。 ！ ？ ， ： ； in Chinese, and . ! ? , : ; in English are chosen as the segmentation anchor candidates. Except Chinese comma, all the other five punctuations can express one semantic end and another semantic beginning. In most of the time, it has high error risk to segment the source sentence by commas. So we get help from syntactic information of Chinese dependency tree to guarantee the monotone order of Chinese sub-sentences.

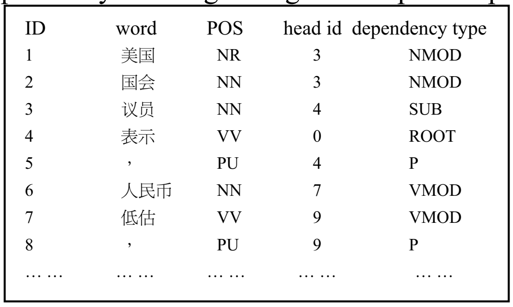
The whole process of sub-sentence segmentation includes training and segmenting.

Training: 1) The word alignment of training parallel corpus is obtained; **2)** The parallel sentence pairs in training corpus are segmented into sub-sentences candidates. For a Chinese-English sentence pair with their word alignment in training data, all bilingual punctuations are found firstly, six punctuations respectively "？！。，：；" in Chinese and "? ! . , : ;" in English. The punctuation identification number (id) sets in Chinese and English are respectively extracted. For a correct punctuation id pair (*id_c*, *id_e*), the phrase before *id_e* in English sentence should be the translation of the phrase before *id_c* in Chinese sentence, namely the number of the links[1] between the two phrases should be equal. In order to guarantees the property we calculate a bilingual alignment ratio for each Chinese-English punctuation id pair according to the following equation. For the punctuation id pair (*id_c*, *id_e*), bilingual alignment ratio consists of two value, Chinese-English alignment ratio (*CER*) and English-Chinese alignment ratio (*ECR*).

$$CER = \frac{\sum_{\substack{1 \le i \le id\_c \\ 1 \le j \le J}} \delta(A_{ij})}{\sum_{\substack{1 \le j \le id\_e \\ 1 \le i \le I}} \delta(A_{ij})} \qquad ECR = \frac{\sum_{\substack{1 \le j \le id\_e \\ 1 \le i \le I}} \delta(A_{ij})}{\sum_{\substack{1 \le i \le id\_c \\ 1 \le j \le J}} \delta(A_{ij})}$$

where $\delta(A_{ij})$ is an indicator function whose value is 1 when the word id pair (i, j) is in the word alignment and is 0 otherwise. *I* and *J* are the length of the Chinese English sentence pair. *CER* of a correct punctuation id pair will be equal to 1.0. So does *ECR*. In view of the error rate of word alignment, the punctuation id pairs will be looked as the segmentation anchor if both *CER* and *ECR* are falling into the threshold range (*minvalue*, *maxvalue*). Then all the punctuation id pairs are judged according to the same method and those punctuation id pairs satisfying the requirement segment the sentence pair into sub-sentence pairs. **3)** The first word of Chinese sub-sentence in each bilingual sub-sentence pair is collected. We filter these words whose frequency is larger than predefined threshold to get segmentation anchor word set (*SAWS*).

Segmenting: 1) The test sentence in Chinese is segmented into segments by the six Chinese punctuation "。！？，：；" in the sentence. **2)** If the first word of a segment is in *SAWS* the punctuation at the end of the segment is chosen as the segmentation punctuation. **3)** If a segment satisfies the property of "dependency integrity" the punctuation at the end of the segment is also chosen as the segmentation punctuation. Here "dependency integrity" is defined in a dependency tree. Figure 5 gives the part output of "lexical dependency parser"[2] for a Chinese sentence. There are five columns of data for each word which are respectively the word id, the word itself, its speech of part, the id of its head word and their dependency type. In the sentence the Chinese word sequence "*美国 国会 议员 表示* (US congressional representatives say that)" has such a property: Each word in the sequence has a dependency relation with the word which is still in the sequence except one word which has a dependency relation with the root, e.g. id 4. We define the property as "dependency integrity". Our reason is: a sub-sentence with the property of "dependency integrity" has relatively independent semantic meaning and a large possibility of monotone translation order. **4)** The union of the segmentation punctuations in step 2) and 3) are the final sub-sentence segmentation tags.



ID	word	POS	head id	dependency type
1	美国	NR	3	NMOD
2	国会	NN	3	NMOD
3	议员	NN	4	SUB
4	表示	VV	0	ROOT
5	，	PU	4	P
6	人民币	NN	7	VMOD
7	低估	VV	9	VMOD
8	，	PU	9	P
……	……	……	……	……

Figure 5. The part dependency parser output of a Chinese sentence.

[1] Here a link between a Chinese word and an English word means the word alignment between them.

[2] http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html

After sub-sentence segmentation, chunks segmentation is carried out in each sub-sentence. We define the chunks as the word sequence which can be translated in monotone order or inverted order. Here the knowledge of the "phrase structure parser"[3] and the "lexicalized dependency parser" are integrated to segment the sub-sentence into chunks. In a Chinese phrase structure parser tree the nouns phrase (NP) and preposition phrase (PP) are relatively independent in semantic expressing and relatively flexible in translation. So in the chunk segmentation, only the NP structure and PP structure in the Chinese structure parsing tree are found as phrase structure chunk. The process of chunk segmentation is described as follows: **1)** the test sub-sentence is parsed to get the phrase structure tree and dependency parsing tree; **2)** We traverse the phrase structure tree to extract sub-tree of "NP" and "PP" to obtain the phrase structure chunks. **3)** We mark off the word sequences with "dependency integrity" in the dependency tree. **4)** Both the two kinds of chunks are recombined to obtain the final result of chunk segmentation.

3.2 Decoding

Our decoder is composed of three styles of reordering models: HPTM, MEBTG and a monotone reordering model.

According to Chiang (2005), given the chunk c_{chunk}, a CKY parser finds $\hat{e}_{chunk}$, the English yield of the best derivation $\hat{D}_{hptm}$ that has Chinese yield c_{chunk}:

$$\begin{aligned}\hat{e}_{chunk} &= e_{chunk}(\hat{D}_{hptm}) \\ &= e_{chunk}(\underset{C(D_{hptm})=C_{chunk}}{\arg\max} \Pr(D_{hptm}))\end{aligned}$$

Here the chunks not the whole source sentence are fed into HPTM decoder to get the *L*-best translations and feature scores of the chunks. We combine all the chunks, their *L*-best translations and the feature scores into a phrase table, namely chunk phrase table. We only choose 4 translation scores (two translation probability based on frequency and two lexical weights based on word alignment) because the language model score, phrase penalty score and word penalty score will be re-calculated in the lower layer of reordering model and need not be kept here. Meantime we change the log values of the scores into probability value. In the chunk phrase table each phrase pair has a Chinese phrase, an English phrase and four translations feature scores. In each phrase pair the Chinese phrase is one of our chunks, the English phrase is one translation of *L*-best of the chunk.

In MEBTG (Xiong et al., 2006), three rules are used to derive the translation of each sub-sentence: lexical rule, straight rule and inverted rule. Given a source sub-sentence $C_{sub-sent}$, it finds the final sub-sentence translation $\hat{E}_{sub-sent}$ from the best derivation $\hat{D}_{mebtg}$:

$$\begin{aligned}\hat{E}_{sub-sent} &= E_{sub-sent}(\hat{D}_{mebtg}) \\ &= E(\underset{C(D_{mebtg})=C_{sub-sent}}{\arg\max} \Pr(D_{mebtg}))\end{aligned}$$

Generally chunk segmentation will make some HPTM rules useless and reduce the translation performance. So in MEBTG we also use base phrase pair table which contains the contiguous phrase translation pairs consistent with word alignment. We merge the chunk phrase table and base phrase table together and feed them into MEBTG to translate each sub-sentence. Thus the *K*-Best translation and feature scores of each sub-sentence are obtained and then are recombined into a new phrase table, namely sub-sentence phrase table, by using the same method with chunk phrase table.

Having obtained the translation of each sub-sentence we generate the final translation of the whole source sentence by a monotone reordering model. Our monotone reordering model employs a log-linear direct translation model. Three phrase tables: chunk phrase table, sub-sentence phrase table and base phrase table are merged together and fed into the monotone decoder. Thus the decoder will automatically choose those phrases it need. In each phrase table each source phrase only has four translation probabilities for its candidate translation. So it's easy to merge them together. In such way all kinds of phrase pairs will automatically compete according to their translation probabilities. So our PRML model can automatically decide which reordering model is employed in each phrase scope of the whole source sentence. It's worth noting that the inputs of the three reordering

[3] http://nlp.stanford.edu/software/lex-parser.shtml

model have no segmentation tag. Because any segmentation for the input before decoding will influence the use of some rules or phrase pairs and may cause some rules or phrase pairs losses. It would be better to employ different phrase table to limit reordering models and let each decoder automatically decide reordering model for each segments of the input. Thus by controlling the phrase tables we apply different reordering models on different phrases. For each reordering model we perform the maximum BLEU training (Venugopal et al. 2005) on a development set. For HPTM the training is same as Chiang 2007. For MEBTG we use chunk phrase table and base table to obtain translation parameters. For monotone reordering model all the three phrase tables are merged to get translation weights.

4 Experiments

This section gives the experiments with Chinese-to-English translation task in news domain. Our evaluation metric is case-insensitive BLEU-4 (Papineni et al. 2002). We use NIST MT 2005, NIST MT 2006 and NIST MT 2008 as our test data. Our training data is filtered from the LDC corpus[4]. Table 1 gives the statistics of our data.

4.1 Evaluating translation Performance

We compare our PRML against two baselines: MEBTG system developed in house according to Xiong (2006, 2008) and HPTM system[5] in PYTHON based on HPTM reordering model (Chiang 2007). In MEBTG phrases of up to 10 words in length on the Chinese side are extracted and reordering examples are obtained without limiting the length of each example. Only the last word of each reordering example is used as lexical feature in training the reordering model by the maximum entropy based classifier[6]. We also set a swapping window size as 8 and the beam threshold as 10. It is worth noting that our MEBTG system uses cube-pruning algorithm (Chiang 2005) from bottom to up to generate the *N*-best list not the lazy algorithm of (Huang and Chiang, 2005). We also limit the length of the HPTM initial rules no more than 10 words and the number of non-terminals within two. In the decoding for the rules the beam pruning parameter is 30 and threshold pruning parameter is 1.0. For hypotheses the two pruning parameters are respectively 30 and 10. In our PRML *minvalue*=0.8, *maxvalue*=1.25, which are obtained by minimum error rate training on the development set. The predefined value for filtering *SAWS* is set as 100.

Set	Language	Sentence	Vocabulary	A. S. L
Train data	Chinese	297,069	6,263	11.9
	English	297,069	8,069	13.6
NIST 05	Chinese	1,082	5669	28.2
	English	4,328	7575	32.7
NIST 06	Chinese	1,664	6686	23.5
	English	6,656	9388	28.9
NIST 08	Chinese	1,357	6,628	24.5
	English	5,428	9,594	30.8

Table 1. The statistics of training data and test data, A. S. L is average sentence length.

The translation performance of the three reordering model is shown in Table 2. We can find that PRML has a better performance than MEBTG with a relatively 2.09% BLEU score in NIST05, 5.60% BLEU score in NIST06 and 5.0% BLEU score in NIST08. This indicates that the chunk phrase table increases the reordering ability of MEBTG. Compared with HPTM, PRML has a comparable translation performance in NIST08. In NIST05 and NIST06 our model has a slightly better performance than HPTM. Because PRML limit hierarchical structure reordering model in chunks while HPTM use them in the whole sentence scope (or in a length scope), HPTM has a more complicated reordering mechanism than PRML. The experiment result shows even though we use easier reordering moels in larger scope, e.g. MEBTG and monoto-

Model	Nist05	Nist06	Nist08
HPTM	0.3183	0.1956	0.1525
MEBTG	0.3049	0.1890	0.1419
PRML	0.3205	0.1996	0.1495

Table 2. The translation performance

[4] LDC corpus lists: LDC2000T46, LDC2000T50, LDC2002E18, LDC2002E27, LDC2002L27, LDC2002T01, LDC2003E07, LDC2003E14, LDC2003T17, LDC2004E12, LDC2004T07, LDC2004T08, LDC2005T01, LDC2005T06, LDC2005T10, LDC2005T34, LDC2006T04, LDC2007T09

[5] We are extremely thankful to David Chiang who originally implement the PYTHON decoder and share with us.

[6] http://maxent.sourceforge.net/

ne reordering model, we have a comparatively translation performance as HPTM.

4.2 Evaluating translation speed

Table 3 shows the average decoding time on test data for the three phrase reordering models on a double processor of a dual 2.0 Xeon machine. Time denotes mean time of per-sentence, in seconds. It is seen that PRML is the slower than MEBTG but reduce decoding time with a relatively 54.85% seconds in NIST05, 75.67% seconds in NIST06 and 65.28% seconds in NIST08. For PRML, 93.65% average decoding time in NIST05 is spent in HPTM, 4.89% time in MEBTG and 1.46% time in monotone reordering decoder.

Model	Nist05	Nist06	Nist08
HPTM	932.96	1235.21	675
MEBTG	43.46	27.16	10.24
PRML	421.20	300.52	234.33

Table 3. The average decoding time

4.3 Evaluating the performance of each layer of phrase table

In order to evaluate the performance of each reordering model, we run the monotone decoder with different phrase table in NIST05. Table 4 list the size of each phrase table. From the results in Table 5 it is seen that the performance of using three phrase tables is the best. Compared with the base phrase table, the translation performances are improved with relatively 10.86% BLEU score by adding chunk phrase table and 11% BLEU score by adding sub-sentence table. The result of row 4 has a comparable to the one in row 5. It indicates the sub-sentence phrase table has contained the information of HPTM reordering model. The case of row 4 to row 2 is the same.

Phrase table	Phrase pair
Base	732732
Chunk	86401
Sub-sentence	24710

Table 4. The size of each phrase table.

Phrase table	Reordering model	BLEU
Base	Monotone	0.2871
Base +chunk	monotone+HPTM	0.3180
Base +sub-sentence table	monotone+HPTM +MEBTG	0.3187
Base +chunk +subsentence	monotone+HPTM +MEBTG	0.3205

Table 5. The performance of phrase table

5 Conclusions

In this paper, we propose a novel reordering model based on multi-layer phrases (PRML), where the source sentence is segmented into different layers of phrases and different reordering models are applied to get the final translation. Our model easily incorporates different styles of phrase reordering models together, including monotone, BTG, and hierarchy or other more complicated reordering models. When a complicated reordering model is used, our model can limit it in a smaller scope and replace it with an easier reordering model in larger scope. In such way our model better trade-offs the translation speed and performance simultaneously.

In the next step, we will use more features to segment the sentences such as syntactical features or adding a dictionary to supervise the segmentation. And also we will try to incorporate other systems into our model to improve the translation performance.

6 Acknowledgements

The research work has been partially funded by the Natural Science Foundation of China under Grant No. 6097 5053, and 60736014, the National Key Technology R&D Program under Grant No. 2006BAH03B02, the Hi-Tech Research and Development Program ("863" Program) of China under Grant No. 2006AA010108-4, and also supported by the China-Singapore Institute of Digital Media (CSIDM) project under grant No. CSIDM-200804, and Research Project "Language and Knowledge Technology" of Institute of Scientific and Technical Information of China (2009DP01-6).

References

David Chiang. 2005. *A hierarchical phrase-based model for statistical machine translation*. In Proceedings of ACL 2005, pages 263–270.

David Chiang, 2007. *Hierarchical Phrase-based Translation*. Computational Linguistics,33(2):201-228.

Yuan Ding and Martha Palmer. 2005. *Machine translation using probabilistic synchronous dependency insertion grammars*. In proceeding of 43th Meeting of the Association for Computational Linguistics, 541-548

Jason Eisner. 2003. *Learning non-isomorphic tree mappings for machine translation*. In proceedings of the 41th Meeting of the Association for Computational Linguistics (companion volume).

Michel Galley, Mark Hopkins, Kevin Knight and Daniel Marcu. 2004. *What's in a translation rule?* In proceedings of HLTNAACL- 2004.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, Ignacio Thayer. 2006. *Scalable Inference and Training of Context-Rich Syntactic Translation Models.* In Proceedings of the joint conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics. Sydney, Australia.

Liang Huang and David Chiang. 2005. *Better k-best parsing*. In Proceedings of the Ninth International Workshop on Parsing Technology, Vancouver, October, pages 53–64.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu, 2002. *BLEU: a method for automatic evaluation of machine translation.* In Proceedings of the 40th Annual Meeting of the ACL. page 311-318, Philadelphia, PA.

Philipp Koehn, Franz J. Och and Daniel Marcu. 2003. *Statistical phrase-based translation*. In proceedings of HLT-NAACL-03, 127-133

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne and David Talbot. 2005. *Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation*. In International Workshop on Spoken Language Translation.

Shankar Kumar and William Byrne. 2005. *Local phrase reordering models for statistical machine translation*. In Proceedings of HLT-EMNLP.

Yang Liu, Qun Liu and Shouxun Lin. 2006. *Tree-to-String Alignment Template for Statistical Machine Translation.* In proceedings of ACL-06, 609-616.

Daniel Marcu and William Wong. 2002. *A phrase-based, joint probability model for statistical machine translation.* In proceedings of EMNLP-02, 133-139.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. *SPMT: Statistical Machine Translation with Syntactified Target Language Phrases*. In Proceedings of EMNLP-2006, 44-52, Sydney, Australia

Bart Mellebeek, Karolina Owczarzak, Josef Van Genabith, Andy Way. 2006. *Multi-Engine Machine Translation By Recursive Sentence Decomposition*. In Proceedings of AMTA 2006

Franz J. Och and Hermann Ney. 2004. *The alignment template approach to statistical machine translation.* Computational Linguistics, 30(4):417-449

Franz Josef Och, Ignacio Thayer, Daniel Marcu, Kevin Knight, Dragos Stefan Munteanu, Quamrul Tipu, Michel Galley, andMark Hopkins. 2004. *Arabic and Chinese MT at USC/ISI*. Presentation given at NIST Machine Translation Evaluation Workshop.

Chris Quirk, Arul Menezes and Colin Cherry. 2005. *Dependency treelet translation: Syntactically informed phrasal SMT*. In proceedings of the 43th Meeting of the Association for Computational Linguistics, 271-279

S. Shieber and Y. Schabes. 1990. *Synchronous tree adjoining grammars*. In proceedings of COLING-90.

Christoph Tillmann. 2004. *A block orientation model for statistical machine translation*. In HLT-NAACL, Boston, MA, USA.

Ashish Venugopal, Stephan Vogel and Alex Waibel. 2003. *Effective Phrase Translation Extraction from Alignment Models*, in Proceedings of the 41st ACL, 319-326.

Dekai Wu. 1995. *Stochastic inversion transduction grammars, with application to segmentation, bracketing, and alignment of parallel corpora.* In proceeding of IJCAL 1995, 1328-1334,Montreal, August.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. *Maximum Entropy Based phrase reordering model for statistical machine translation*. In proceedings of COLING-ACL, Sydney, Australia.

Deyi Xiong, Min Zhang, Ai Ti Aw, Haitao Mi, Qun Liu and Shouxun Lin. *Refinements in BTG-based Statistical Machine Translation.* In Proceedings of IJCNLP 2008.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In proceedings of the 39th Meeting of the ACL, 523-530.

R. Zens, H. Ney, T. Watanabe, and E. Sumita. 2004. *Reordering Constraints for Phrase-Based Statistical MachineTranslation.* In Proceedings of CoLing 2004, Geneva, Switzerland, pp. 205-211.

Standardizing Wordnets in the ISO Standard LMF: Wordnet-LMF for GermaNet

Verena Henrich
Universtiy of Tübingen
Department of Linguistics
verena.henrich@uni-tuebingen.de

Erhard Hinrichs
Universtiy of Tübingen
Department of Linguistics
erhard.hinrichs@uni-tuebingen.de

Abstract

It has been recognized for quite some time that sustainable data formats play an important role in the development and curation of linguistic resources. The purpose of this paper is to show how GermaNet, the German version of the Princeton WordNet, can be converted to the Lexical Markup Framework (LMF), a published ISO standard (ISO-24613) for encoding lexical resources. The conversion builds on Wordnet-LMF, which has been proposed in the context of the EU KYOTO project as an LMF format for wordnets. The present paper proposes a number of crucial modifications and a set of extensions to Wordnet-LMF that are needed for conversion of wordnets in general and for conversion of GermaNet in particular.

1 Introduction

It has been recognized for quite some time that sustainable data formats play an important role in the development and curation of linguistic resources. As witnessed by the success of the guidelines of the Text Encoding Initiative[1] (TEI) and of published standards issued by the International Standards Organization[2] (ISO), markup languages such as XML[3] (short for: Extensible Markup Language) have become lingua francas for encoding linguistic resources of different types, including phonetic transcriptions, (annotated) text corpora, and dictionaries. It is fair to say that it has become common practice among developers of new linguistic resources to consult TEI guidelines and ISO standards in order to develop standard-conformant encoding schemes that serve as an interchange format and that can be documented and validated by Document Type Definitions (DTD) and XML schemata.

However, for resources that were developed prior to or largely in parallel with the emerging acceptance of markup languages and of emerging encoding standards, the situation is far more heterogeneous. A wide variety of legacy formats exists, many of which have persisted due to existing user communities and the availability of tools that can process only such idiosyncratic formats. The development of wordnets for a large number of languages is a typical example of a type of linguistic resource, where legacy formats still persist as a de facto standard. WordNet 1.6 is encoded in the data format of lexicographer files[4] that was designed for the English Princeton WordNet (Fellbaum, 1998). It is a plain-text format for storing wordnet data and allows lexicographers to encode lexical and conceptual relations among lexical units and synsets by use of special-purpose diacritics. There exist numerous tools that can process WordNet 1.6 lexicographer files to extract relevant information or to transform the data into other special-purpose formats such as Prolog-fact databases. Even tough still widely used for the reasons just mentioned, the complexity of the format itself has a number of undesirable consequences. As Henrich and Hinrichs (2010) have pointed out,

[1] See http://www.tei-c.org
[2] See http://www.iso.org
[3] See http://www.w3.org/TR/REC-xml/
[4] See http://wordnet.princeton.edu/man/lexnames.5WN.html

the editing of lexicographer files is highly error-prone and time-consuming in actual lexicographic development. Moreover, format validation of the data as well as development of new tools for data visualization and data extraction become increasingly difficult since they cannot be based on generic state-of-the-art tools, that are, for example, available for XML-based encodings.

For exactly these reasons, XML-based interchange formats have been proposed in recent years also for wordnets. One of the first, if not the first, example is the XML format for GermaNet[5], a wordnet for German (Lemnitzer and Kunze, 2002; Henrich and Hinrichs, 2010). An even more recent development along these lines is the specification of Wordnet-LMF (see Soria et al., 2009), an instantiation of the Lexical Markup Framework[6] (LMF, (Francopoulo et al., 2006)) customized for wordnets.

Since LMF is an ISO standard (ISO-24613), it is a particularly attractive candidate for encoding wordnets. Everything else being equal, ISO standards have a high chance of being adopted by a wide user community and of being recognized as an interchange format.[7] Such agreed-upon interchange formats are a crucial prerequisite for interoperable linguistic resources in the context of web services and of processing pipelines for linguistic resources.

The purpose of this paper is threefold:

1. To compare and contrast the GermaNet XML initially proposed by Lemnitzer and Kunze (2002) with the Wordnet-LMF. This comparison is instructive since it reveals two completely different conceptions of representing semantic knowledge at the lexical level.
2. To point out a number of open issues that need to be resolved if Wordnet-LMF is to be adopted widely among wordnets for a steadily increasing number of languages.
3. To show how these open issues can be resolved in a customized version of Wordnet-LMF suitable for GermaNet.

The remainder of this paper is structured as follows: section 2 provides a general introduction to GermaNet. Details about the adapted XML format used for GermaNet up until now are provided in section 3. Section 4 introduces the challenge of how to represent a wordnet in the Lexical Markup Framework. As one possibility, Wordnet-LMF is regarded. Issues that arise during the conversion of GermaNet into Wordnet-LMF lead to a modified version of Wordnet-LMF. Finally, section 5 concludes with a comparison of the two representation formats.

2 GermaNet

GermaNet is a lexical semantic network that is modeled after the Princeton WordNet for English. It partitions the lexical space into a set of concepts that are interlinked by semantic relations. A semantic concept is modeled by a *synset*. A synset is a set of words (called *lexical units*) where all the words are taken to have (almost) the same meaning. Thus a synset is a set-representation of the semantic relation of synonymy, which means that it consists of a list of lexical units and a paraphrase (represented as a string). The lexical units in turn have frames (which specify the syntactic valence of the lexical unit) and examples. The list of lexical units for a synset is never empty, but any of the other properties may be.

There are two types of semantic relations in GermaNet: *conceptual* and *lexical relations*. Conceptual relations hold between two semantic concepts, i.e. synsets. They include relations such as hyperonymy, part-whole relations, entailment, or causation. Lexical relations hold between two individual lexical units. Antonymy, a pair of opposites, is an example of a lexical relation.

GermaNet covers the three word categories of adjectives, nouns, and verbs, each of which is hierarchically structured in terms of the hyperonymy relation of synsets.

[5] See http://www.sfs.uni-tuebingen.de/GermaNet/

[6] See http://www.lexicalmarkupframework.org

[7] An anonymous reviewer raised the question why OWL is not a good candidate for encoding wordnets. On this issue, we agree with the assessment of Soria et al. (2009) who point out that *"[...] RDF and OWL are conceptual repositories representation formats that are not designed to represent polysemy and store linguistic properties of words and word meanings."*

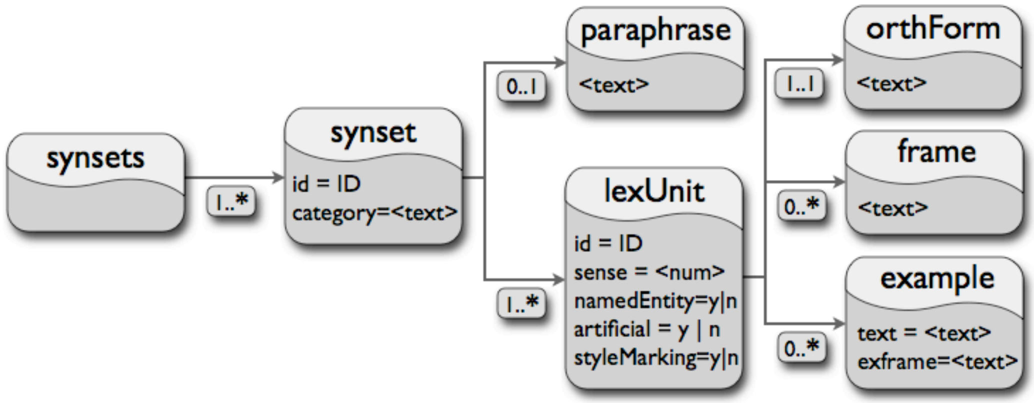


Figure 1. Structure of the XML synset files.

3 Current GermaNet XML Format

The structure of the XML files closely follows the internal structure of GermaNet, which means that the file structure mirrors the underlying relational organization of the data. There are two DTDs that jointly describe the XML-encoded GermaNet. One DTD represents all synsets with their lexical units and their attributes (see subsection 3.1). The other DTD represents all relations, both conceptual and lexical relations (see subsection 3.2).

The GermaNet XML format was initially developed by Kunze and Lemnitzer (2002), but modifications of the GermaNet data itself led to an adopted XML format, which is presented here.[8]

3.1 XML Synset Files

The XML files that represent all synsets and lexical units of GermaNet are organized around the three word categories currently included in GermaNet: nouns, adjectives, and verbs (altogether 54 synset files since the semantic space for each word category is divided into a number of semantic subfields).

The structure of each of these files is illustrated in Figure 1[9]. Each *synset* represents a set of lexical units (*lexUnits*) which all express the same meaning. This grouping represents the semantic relation of synonymy. Further properties of a *synset* (e.g., the word *category* or a describing *paraphrase*) and a lexical unit (e.g., a *sense* number or the orthographical form (*orthForm*)) are encoded appropriately.

Figure 1 describes the underlying XML structure. Each box in the figure stands for an element in the XML files, and the properties in each box (listed underneath the wavy line) represent the attributes of an XML element. This means, for example, that a *synset* element has the attributes of an *id* and a *category*.[10]

Figure 2 shows an example of a *synset* with two lexical units (*lexUnit* elements) and a *paraphrase*. The *lexUnit* elements in turn contain several attributes and an orthographical form (the *orthForm* element), e.g., *leuchten* (German verb for: *to shine*). The first of the two lexical units even has a *frame* and an *example*.

```
<synset id="s58377" category="verben">
  <lexUnit id="l82207"
           sense="1"
           namedEntity="no"
           artificial="no"
           styleMarking="no">
    <orthForm>leuchten</orthForm>
    <frame>NN</frame>
    <example>
      <text>
        Der Mond leuchtete in der Nacht.
      </text>
      <exframe>NN</exframe>
    </example>
  </lexUnit>
  <lexUnit id="l82208"
```

[8] The interested reader might compare the version at hand with (Lemnitzer and Kunze, 2002) or (Kunze and Lemnitzer, 2002), which both describe the initial GermaNet XML version.

[9] In fact, this figure is not quite complete for the reason of simplicity.

[10] Note that XML element or attribute names appear *italic* if they are referenced in the text.

```
          sense="2"
          namedEntity="no"
          artificial="no"
          styleMarking="no">
    <orthForm>strahlen</orthForm>
  </lexUnit>
  <paraphrase>
    Lichtstrahlen aussenden,
    große Helligkeit verbreiten
  </paraphrase>
</synset>
```

Figure 2. Synset file example.

3.2 XML Relation File

This type of XML file represents both kinds of relations: conceptual and lexical relations. All relations are encoded within one XML file, whose structure is illustrated in Figure 3.

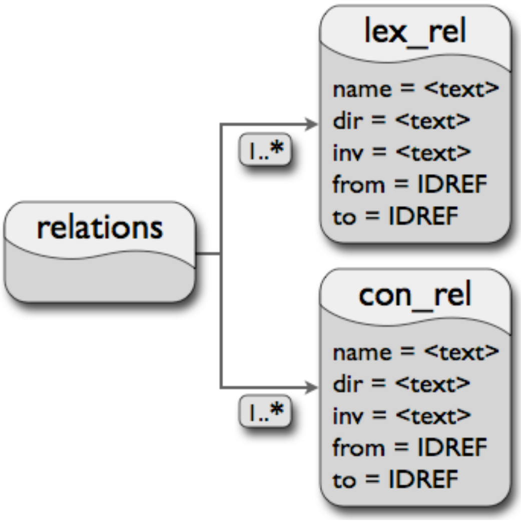


Figure 3. Structure of the XML relation file.

The boxes in Figure 3 again represent XML elements, which means that there is one *relations* element that contains all lexical relations (*lex_rel* elements) and conceptual relations (*con_rel* elements). Both relation types contain several attributes.

Figure 4 illustrates an example for each of the two relation types. The type of the conceptual relation is *hyperonymy* (indicated by the *name* attribute), and it holds between the synset with ID *s58377* (*from* attribute) and the synset with ID *s58376* (*to* attribute). The lexical relation is of type *antonymy* (again indicated by the *name* attribute), and holds between the lexical units with the IDs *l2471* (*from* attribute) and *l12470* (*to* attribute).

```
<con_rel name="hyperonymy"
         from="s58377" to="s58376"
         dir="revert" inv="hyponymy" />
<lex_rel name="antonymy"
         from="l2471" to="l2470"
         dir="both" />
```

Figure 4. Example from relation file.

4 Wordnet-LMF

The Lexical Markup Framework (ISO-24613) is an ISO standard for encoding natural language processing lexicons and machine readable dictionaries (Francopoulo et al., 2006). The intention of LMF is to provide a common model for the creation and use of lexical resources, to manage the exchange of data between and among these resources, and to enable the merging of a large number of individual electronic resources to form extensive global electronic resources.

4.1 The Challenge

The core structure of LMF is based on the prototypical structuring of a lexicon in terms of lexical entries, each of which enumerates the different senses of the lexical item in question. This word-driven perspective contrasts the synset-driven relational structure of wordnets – the grouping of word senses (i.e., lexical units) that express the same meaning into synsets. Exactly these two radically different organizing principles (relation-based in the case of wordnets versus lexical-entry-based in the case of LMF) constitute the challenge of encoding wordnets in LMF. We take up this challenge: *How can a synset-based wordnet, e.g. GermaNet, be represented in a word-driven format like LMF?*

4.2 Apply LMF to Wordnets

The conversion of GermaNet to LMF will build on Wordnet-LMF (Soria et al., 2009; Lee et al., 2009), an existing Lexical Markup Framework subset[11]. Wordnet-LMF has been developed in the context of the EU KYOTO

[11] Wordnet-LMF is a proper subset of LMF since there are specifications in LMF that are not in Wordnet-LMF and since there is nothing in Wordnet-LMF which is not in LMF. Soria et al. (2009) themselves refer to Wordnet-LMF as an LMF *dialect*.

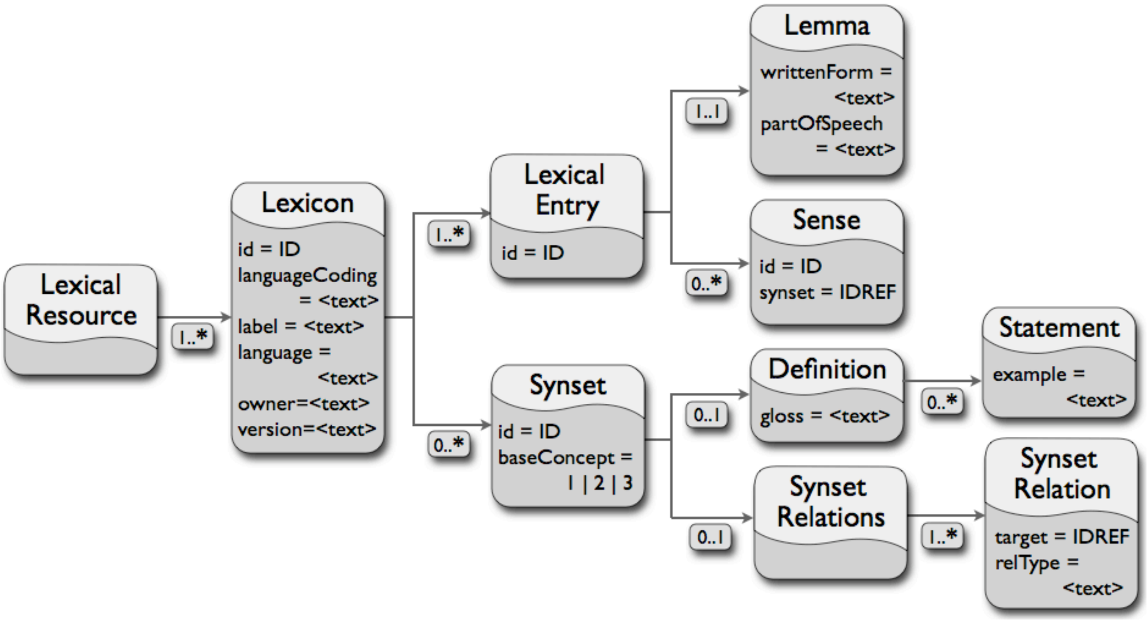


Figure 5. The Wordnet-LMF structure.

project[12] and is especially tailored to encode wordnets in the LMF standard.

Wordnet-LMF is specified by a Document Type Definition (see Appendix E in (Soria and Monachini, 2008)) and fully complies with standard LMF.

The Wordnet-LMF XML structure is shown in Figure 5[13]. There is a *Lexical Resource* which contains at least one *Lexicon* (in this case a wordnet lexicon).[14] A *Lexical Entry* represents a word entry in a *Lexicon*, where the word itself is represented by the *writtenForm* attribute of the *Lemma* element. *Lexical Entries* group different *Senses* of a particular word. The *Senses* have a *synset* attribute that relates them to a *Synset* element by the corresponding ID. If two *Senses* have the same *synset* attribute, they belong to the same *Synset* and are thus synonyms.

A *Synset* can have several relations to other *Synsets*. These relations are encoded in *SynsetRelation* elements.

4.3 Apply Wordnet-LMF to GermaNet

The differences between the synset-driven structure of GermaNet (see Figures 1 and 3) and the word-driven format of Wordnet-LMF (see Figure 5) are obvious. But there is also a strong commonality: Both formats have synset elements that cluster synonymous words. In GermaNet, the words are represented by lexical units that are child elements of a synset. In Wordnet-LMF, senses, which correspond to the lexical units in GermaNet, are linked to a synset (by an attribute containing a synset ID).

The conversion of GermaNet to Wordnet-LMF proceeds as follows: Each lexical unit of GermaNet is turned into a *Sense* element in Wordnet-LMF (see Figure 5). The *synset* attribute (containing a *Synset* ID) of the *Sense* element links this *Sense* with the *Synset* that it is a member of. The different *Sense* elements are grouped by their orthographical form (the *Lemma* in Wordnet-LMF) into *Lexical Entries*.

An example of a GermaNet *LexicalEntry* in Wordnet-LMF is shown in Figure 6. This *LexicalEntry* represents the word *leuchten* (German verb for: *to shine*), as the *writtenForm* attribute of the *Lemma* element indicates. This *LexicalEntry* has two *Senses*, which belong to different *Synsets* (see the different *synset* attributes of the *Sense* elements).

[12] See http://www.kyoto-project.eu

[13] Note that this figure does not show the whole Wordnet-LMF model. Only the monolingual part that is relevant for this paper is represented. The representation of multilingual resources (i.e., the optional *SenseAxis* element with its children) is not considered in this paper. For a complete picture, see Soria et Monachini (2008).

[14] Here, XML element or attribute names again appear *italic* if they are referenced in the text.

Each *Sense* has a *MonolingualExternalRefs* element with at least one *MonolingualExternalRef* representing a reference to an external system. In this case, each *Sense* is linked to the corresponding entry in the GermaNet database[15]; the *externalReference* attribute of a *MonolingualExternalRef* specifies the database table name with a database ID.

```
<LexicalEntry id="deu-52-l4601-v">
  <Lemma writtenForm="leuchten"
                      partOfSpeech="v" />
  <Sense id="deu-52-l4601-v_1"
                synset="deu-52-s58377-v">
    <MonolingualExternalRefs>
      <MonolingualExternalRef
        externalSystem="GermaNet-Database"
        externalReference=
              "lex_uni_table#id=82207" />
    </MonolingualExternalRefs>
  </Sense>
  <Sense id="deu-52-l4601-v_2"
                synset="deu-52-s58718-v">
    <MonolingualExternalRefs>
      <MonolingualExternalRef
        externalSystem="GermaNet-Database"
        externalReference=
              "lex_uni_table#id=82677" />
    </MonolingualExternalRefs>
  </Sense>
</LexicalEntry>
```

Figure 6. Example of a *LexicalEntry*.

In the next conversion step, all synsets of GermaNet are listed with their relations to other synsets. The corresponding *Synset* (with the ID *deu-52-s58377-v*) of the first *Sense* in Figure 6 is illustrated in Figure 7. It has, inter alia, a describing *gloss* and two *example* sentences.

The element *SynsetRelations* encodes relations to other *Synset* instances. The relations are simply encoded with a *target* attribute that contains the ID of the referencing *Synset*. The *Synsets* in Wordnet-LMF are logically the “same” as the synsets in GermaNet XML, i.e. the concept that a synset expresses is exactly the same in both formats.

Each *Synset* has a reference to the GermaNet database. Therefore, the *MonolingualExternalRef* element links to the corresponding entry in the GermaNet database; the *externalReference* attribute specifies the database table name with the synsets database ID.

```
<Synset id="deu-52-s58377-v"
                        baseConcept="1">
  <Definition gloss="Lichtstrahlen
             aussenden, große Helligkeit
             verbreiten">
    <Statement example="Der Mond leuchtete
                         in der Nacht."/>
    <Statement example="Die Lichter der
           Stadt strahlen in die Nacht."/>
  </Definition>
  <SynsetRelations>
    <SynsetRelation
               target="deu-52-s58376-v"
               relType="has_hyperonym"/>
  </SynsetRelations>
  <MonolingualExternalRefs>
    <MonolingualExternalRef
        externalSystem="GermaNet-Database"
        externalReference=
               "synset_table#id=58377"/>
  </MonolingualExternalRefs>
</Synset>
```

Figure 7. Example of a *Synset*.

These two Figures 6 and 7 represent the same example in Wordnet-LMF that was already shown in the GermaNet XML format in Figure 1.

4.4 Necessary Modifications to Wordnet-LMF

As the previous discussion has shown, Wordnet-LMF provides a very useful basis for converting GermaNet into LMF. However, a number of modifications to Wordnet-LMF are needed if this conversion is to preserve all information present in the original resource. The present section will discuss a number of modifications to Wordnet-LMF that are needed for conversion of wordnets in general. In addition, we will also discuss a set of extensions to Wordnet-LMF that are needed for conversion of GermaNet in particular.

The most glaring omission in Wordnet-LMF concerns the modeling of lexical relations which hold between lexical units (i.e., *Senses* in the terminology of Wordnet-LMF). In the current Wordnet-LMF DTD only conceptual relations (i.e., *SynsetRelations* in the terminology of Wordnet-LMF), which hold between synsets, are modeled. Thus antonymy, which is a typical example of a lexical relation (see (Fellbaum, 1998) for further details), can cur-

[15] For efficency reasons, GermaNet is stored in a relational database.

rently not be modeled without violating the Wordnet-LMF DTD.

Among the synset relations specified in Wordnet-LMF, the entailment relation is missing, which plays a crucial role in the modeling of verbs in the Princeton WordNet and in GermaNet alike. The list of values of attribute *relType* for *SynsetRelation* elements (see Appendix A in (Soria and Monachini, 2008)) therefore has to be amended accordingly.[16]

A third omission in the current Wordnet-LMF DTD concerns syntactic frames used in the Princeton WordNet to indicate the syntactic valence of a given word sense. Syntactic frames are also used in GermaNet, albeit using a different encoding[17]. Syntactic frames together with example sentences, which illustrate the meaning and prototypical usage of a particular word, help to distinguish among word senses.

In WordNet both syntactic frames and examples are linked to synsets. However, at least in the case of syntactic frames the linkage to synsets seems problematic since different members of the same synset may well have different valence frames. For example, the German verbs *finden* and *begegnen* both mean *meet* and thus belong to the same synset. Both are transitive verbs, but their object NPs have different cases: accusative case for *treffen* and dative case for *begegnen*. As this example shows, syntactic frames need to be associated with lexical units rather than synsets. This is exactly the design choice made in GermaNet, as shown in Figure 1.

A related question concerns the anchoring of example sentences which illustrate the meanings and prototypical usage of a particular word sense. In both the Princeton WordNet and GermaNet such examples are associated with lexical units[18]. GermaNet correlates examples additionally with particular syntactic frames and treats both examples and syntactic frames as properties of lexical units, i.e. *Senses* in the terminology of Wordnet-LMF.

The above issues lead to a modified version of the Wordnet-LMF DTD as shown in Figure 8. Compared to Figure 5, the *Sense* element is enriched by three optional subelements: *SenseRelations*, *SenseExamples*, and *SubcategorizationFrames*.

It has to be noted, though, that LMF proper contains all necessary elements. The three notions *SenseRelation*, *SenseExample*, and *SubcategorizationFrame* come from LMF proper and these elements can be used to remedy the omissions in Wordnet-LMF.

The *SenseRelation* element in Figure 8 represents relations between different *Senses* (the lexical units in GermaNet). The *SenseExamples* and *SubcategorizationFrames* elements both group several *SenseExample* or *SubcategorizationFrame* instances. A *SubcategorizationFrame* element represents the syntactic valence of a word sense. A *SenseExample* shows the prototypical usage of a word sense as an example sentence. The syntactic valence for a concrete example sentence can be specified with the optional *frame* attribute of a *SenseExample*.

5 Conclusion: Comparing GermaNet XML with Wordnet-LMF XML

We would like to conclude with a comparison between the GermaNet native XML format described in section 3 and the modified Wordnet-LMF format described in section 4.4. Since the GermaNet native XML format was particularly tailored to the structure of GermaNet, it enjoys the usual advantages of such customized solutions: it contains all and only the necessary XML elements and attributes to describe the resource. Moreover, the data are distributed over 55 different XML files, which facilitates easy data handling and efficient search by word classes and lexical fields. These properties are in fact exploited by a number of GermaNet-specific tools, including

[16] Piek Vossen (personal communication) has pointed out to us that Wordnet-LMF does not impose a list of relations as a standard yet.

[17] In WordNet, frames are encoded in a controlled language using paraphrases such as *Somebody ----s something* for a transitive verb with an animate subject and an inanimate object. The frames in GermaNet use complementation codes provided with the German version of the CELEX Lexical Database (Baayen et al., 2005) such as *NN.AN* for transitive verbs with accusative objects.

[18] In WordNet, the examples are placed at the synset level, but referencing to a word sense at the same time.

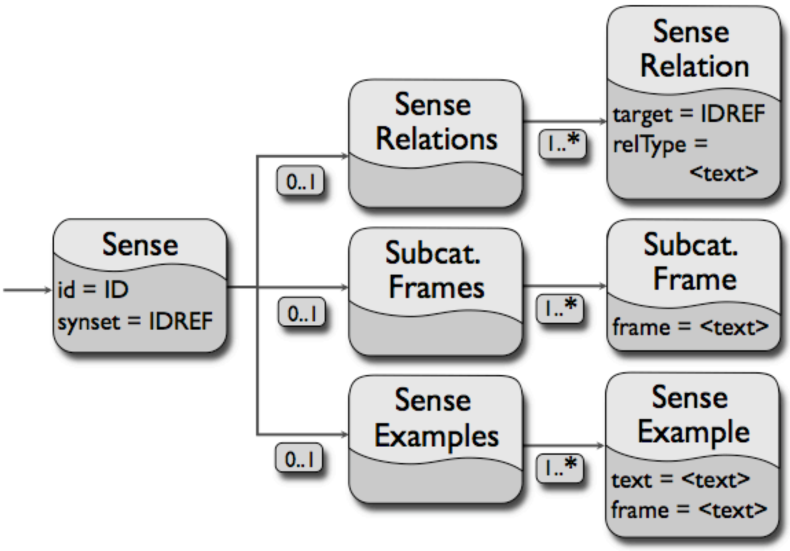


Figure 8. Revised Wordnet-LMF structure.

a GermaNet-Explorer, a tool for data exploration and retrieval, and a GermaNet Pathfinder, a tool for the calculation of semantic relatedness, similarity, and distance (Cramer and Finthammer, 2008). All of these tools utilize the Java API that has been developed for the GermaNet native XML format.

At the same time the GermaNet native XML format is a proprietary data format that was developed at a time when the only de facto encoding standard for wordnets consisted of the lexicographer files, originally developed for the Princeton WordNet. As such GermaNet XML was never developed with the goal of providing an XML standard for modeling wordnets in general. With Wordnet-LMF a candidate standard has now been proposed that is compliant with the LMF ISO standard for lexical resources and that strives to provide a general encoding standard of wordnets for different languages. As the discussion in section 4.4 has shown, the current Wordnet-LMF DTD still needs to be amended to account for the full range of wordnet relations, frames, and examples (see Figure 8). These elements are not in Wordnet-LMF because Wordnet-LMF is a subset, but these elements are defined in the ISO document 24613 where LMF proper is defined. However, Wordnet-LMF appears to be suitably mature to serve as an interchange format for wordnets of different languages as well as for linking wordnets of different languages with one another[19].

Acknowledgements

The research reported in this paper was funded by the BW-eSci(T) grant sponsored by the Ministerium für Wissenschaft, Forschung und Kunst Baden-Württemberg.

We would like to thank Piek Vossen and an anonymous reviewer for valuable comments on an earlier version of this paper.

References

Baayen, R. H., R. Piepenbrock, and L. Gulikers. 2005. *The CELEX Lexical Database (Release 2) CD-ROM.* Philadelphia, PA: Linguistic Data Consortium, University of Pennsylvania (Distributor).

Cramer, Irene, and Marc Finthammer. 2008. Tools for Exploring GermaNet in the Context of CL-Teaching. In: *Angelika Storrer, Alexander Geyken, Alexander Siebert, and Kay-Michael Würzner, (Eds.): Text Resources and Lexical Knowledge*. Selected Papers from the 9th Conference on Natural Language Processing KONVENS 2008. Berlin/New York: Mouton de Gruyter, 195-208.

[19] For example, the Interlingual Index, based on the Princeton WordNet, can be used to link different wordnets with one another.

Kunze, Claudia, and Lothar Lemnitzer. 2002. GermaNet – representation, visualization, application. *Proceedings of LREC 2002*, main conference, Vol V. pp. 1485-1491.

Fellbaum, Christiane (eds.). 1998. *WordNet – An Electronic Lexical Database*. The MIT Press.

Francopoulo, Gil, Monte George, Nicoletta Calzolari, Monica Monachini, Nuria Bel, Mandy Pet, and Claudia Soria. 2006. Lexical markup framework (LMF). *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*. Genoa, Italy.

Henrich, Verena, and Erhard Hinrichs. 2010. GernEdiT – The GermaNet Editing Tool. *Proceedings of LREC 2010*, main conference. Valletta, Malta.

Lee, Lung-Hao, Shu-Kai Hsieh, and Chu-Ren Huang. 2009. CWN-LMF: Chinese WordNet in the Lexical Markup Framework. *Proceedings of the 7th Workshop on Asian Resources*. Suntec, Singapore, August 06 - 07, 2009, pp. 123-130

Lemnitzer, Lothar, and Claudia Kunze. 2002. Adapting GermaNet for the Web. *Proceedings of the First Global Wordnet Conference*. Central Institute of Indian Languages, Mysore, India, 21.-25.01.2002, pp. 174-181

Soria, Claudia, Monica Monachini, and Piek Vossen. 2009. Wordnet-LMF: Fleshing out a Standardized Format for Wordnet Interoperability. *Proceedings of ACM Workshop on Intercultural Collaboration*.

Soria, Claudia, and Monica Monachini. 2008. *Kyoto-LMF – Wordnet representation format*. KYOTO Working paper: WP02_TR002_V04_Kyoto_LMF.

Vossen, Piek, Eneko Agirre, Nicoletta Calzolari, Christiane Fellbaum, Shu-kai Hsieh, Chu-Ren Huang, Hitoshi Isahara, Kyoko Kanzaki, Andrea Marchetti, Monica Monachini, Federico Neri, Remo Raffaelli, German Rigau, Maurizio Tescon, and Joop VanGent. 2008. KYOTO: A system for mining, structuring and distributing knowledge across languages and cultures. *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*. Marrakech, Morocco.

Normal-form parsing for Combinatory Categorial Grammars with generalized composition and type-raising

Julia Hockenmaier **Yonatan Bisk**
University of Illinois at Urbana-Champaign
{juliahmr, bisk1}@illinois.edu

Abstract

We propose and implement a modification of the Eisner (1996) normal form to account for generalized composition of bounded degree, and an extension to deal with grammatical type-raising.

1 Introduction

Combinatory Categorial Grammar (Steedman, 2000) is a linguistically expressive grammar formalism that has been used for many NLP applications, including wide-coverage parsing (Clark and Curran, 2007; Hockenmaier, 2003) and semantic interpretation (Curran et al., 2007), semantic role-labeling (Gildea and Hockenmaier, 2003; Boxwell et al., 2009), semantic parsing (Zettlemoyer and Collins, 2005) and natural language generation (Espinosa et al., 2008).

An essential feature of CCG is its flexible constituent structure, licensed by type-raising and composition rules which can create "non-standard" constituents such as *"John saw"*, or *"Mary talked to"*, required in constructions involving non-local dependencies, such as wh-extraction (Fig. 1) or right-node raising. Since *"John saw"* can now also be a constituent in *"John saw Mary"*, this leads to a combinatorial explosion of *spurious ambiguities*, i.e. multiple syntactic derivations of the same semantic interpretation (Wittenburg, 1986). This can create problems for applications based on CCG, e.g. for the induction of stochastic CCGs from text annotated with logical forms (Zettlemoyer and Collins, 2007), where spreading probability mass over equivalent derivations should be avoided. A number of *normal-form* (NF) parsing algorithms that aim to produce only one derivation per interpretation have been proposed (Wittenburg, 1986; Niv, 1994; Pareschi and Steedman, 1987; Hepple and Morrill, 1989; Eisner, 1996). Computationally, such algorithms are very attractive since they do not require costly semantic equivalence checks (Karttunen, 1989; Komagata, 2004) during parsing. Eisner's (1996) normal form is the most developed and well-known of these approaches, but is only defined for a variant of CCG where type-raising is a lexical operation and where the degree of composition is unbounded. Therefore, it and its equivalent reformulation by Hoyt and Baldridge (2008) in a multimodal variant of CCG are not safe (preserve all interpretations) and complete (remove all spurious ambiguities) for more commonly used variants of CCG. In particular, this NF is not safe when the degree of composition is bounded,[1] and not complete when type-raising is a grammatical operation. This paper defines a NF for CCG with bounded composition and grammatical type-raising.

2 Combinatory Categorial Grammar

In CCG, every constituent (*"John saw"*) has a syntactic category (S/NP) and a semantic interpretation ($\lambda x.saw(john', x)$).[2] Constituents combine according to a small set of language-

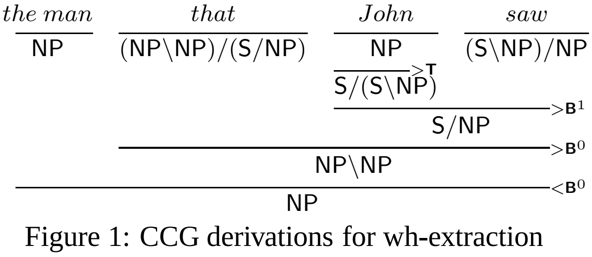


Figure 1: CCG derivations for wh-extraction

[1] Although Eisner (1996, section 5) also provides a safe and complete parsing algorithm which can return non-NF derivations when necessary to preseve an interpretation if composition is bounded or the grammar is restricted in other (arbitrary) ways.

[2] More complex representations than simple predicate-argument structures are equally possible.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 465–473,
Beijing, August 2010

Application	$(>)$	$X/Y: \lambda x.f(x)$	$Y: a$	$\Rightarrow X: f(a)$
	$(<)$	$Y: a$	$X\backslash Y: \lambda x.f(x)$	$\Rightarrow X: f(a)$
Composition	$(>\mathbf{B}^1)$	$X/Y: \lambda x.f(x)$	$Y/Z: \lambda y.g(y)$	$\Rightarrow X/Y: \lambda z.f(g(z))$
	$(<\mathbf{B}^1)$	$Y\backslash Z: \lambda y.g(y)$	$X\backslash Y: \lambda x.f(x)$	$\Rightarrow X\backslash Y: \lambda z.f(g(z))$
	$(>\mathbf{B}^1_\times)$	$X/Y: \lambda x.f(x)$	$Y\backslash Z: \lambda y.g(y)$	$\Rightarrow X\backslash Y: \lambda z.f(g(z))$
	$(<\mathbf{B}^1_\times)$	$Y/Z: \lambda y.g(y)$	$X\backslash Y: \lambda x.f(x)$	$\Rightarrow X/Y: \lambda z.f(g(z))$
	$(>\mathbf{B}^n)$	$X/Y: \lambda x.f(x)$	$Y\vert Z_1\vert ...\vert Z_n: \lambda z_n..z_1.g(z_1...z_n)$	$\Rightarrow X\vert Z_1\vert ...\vert Z_n: \lambda z_n...z_1.f(g(z_1...z_n))$
	$(<\mathbf{B}^n)$	$Y\vert Z_1\vert ...\vert Z_n: \lambda z_n..z_1.g(z_1...z_n)$	$X\backslash Y: \lambda x.f(x)$	$\Rightarrow X\vert Z_1\vert ...\vert Z_n: \lambda z_n...z_1.f(g(z_1...z_n))$
Typeraising	$(>\mathbf{T})$	For $X \in \mathcal{C}_{arg}$: $X: a$		$\Rightarrow T/_i(T\backslash_i X): \lambda f.f(a)$
	$(<\mathbf{T})$	For $X \in \mathcal{C}_{arg}$: $X: a$		$\Rightarrow T\backslash_i(T/_i X): \lambda f.f(a)$

Figure 2: CCG's combinatory rules.

independent combinatory rules (Fig. 2). The lexicon pairs words with categories and interpretations and is language-specific.

Syntax We distinguish atomic (S, NP, PP, etc.) from complex categories ((S\NP)/NP, N/N, etc.). A complex category of the form X/Y (or X\Y) represents a function which returns a *result* of type X when applied to an *argument* of type Y, which, in the case of a forward slash (/) has to follow the functor, and in the case of a backslash (\) has to preceed it. X and Y can themselves be complex again. We will use categories with vertical slashes when the direction of the slash does not matter, and may omit unnecessary parentheses (so X|Y|Z will represent (X\Y)/Z, (X\Y)\Z, ...). We will also use the shorthand $X|Y_{1..n}$ (or $X|\alpha$) to refer to a category with (possibly complex) result X and arguments $Y_1...Y_n$ (or an unspecified, possibly empty, list of arguments $\alpha = Y_{0...n}$, where $|\alpha| = n$) that can each appear with either type of slash.

Semantics If the category of a constituent is atomic (NP; S), its interpretation will also be atomic (*kim'*; *sleeps'(kim')*), and if the category is a functor of arity n ($X|Y_{1..n}$), the interpretation is a λ-expression $\lambda y_n..\lambda y_1 \phi(y_1...y_n)$ of arity n.

The lexicon Each language defines a finite set of lexical category types $\mathcal{C}_{lex}$ (e.g. (S\NP)/NP is in the English lexicon, but (S\NP)\NP is not) with maximal arity N_L. This defines a set of lexical argument category types $\mathcal{C}_{arg}$, consisting of all categories Y that are the argument of some lexical category $(X|Y)|\beta \in \mathcal{C}_{lex}$ (with $|\beta| \geq 0$). Since $\mathcal{C}_{lex}$ is finite, $\mathcal{C}_{arg}$ is strictly smaller than $\mathcal{C}_{lex}$ (and usually consists of basic categories such as NP, S, S\NP).

Combinatory Rules In addition to function application ($>, <$), CCG has three kinds of combinatory rules (Fig. 2): harmonic function composition ($>\mathbf{B}^{(1)}$, $<\mathbf{B}^{(1)}$), crossing function composition ($>\mathbf{B}_\times$,$<\mathbf{B}_\times$) and type-raising ($>\mathbf{T}$, $<\mathbf{T}$). All rules take one or two *input* categories and yield one *output* category, and consist of a syntactic and a corresponding semantic operation. Composition also has *generalized* variants $>\mathbf{B}^n$, $<\mathbf{B}^n$ up to a fixed degree $N_\mathbf{B}$.[3] Composition of unbounded degree increases the generative capacity of CCG (Weir, 1988), and should be disallowed. Application ($>, <$) can be seen as a special case of composition ($>\mathbf{B}^0$,$<\mathbf{B}^0$). When composing X|Y with Y|Z to X|Z, we call X|Y the *primary* input and Y|Z the *secondary* input. Harmonic composition allows associativity: the string A/B B/C C now has an alternative derivation where A/B and B/C compose into A/C, whereas crossing composition enables novel permutations, such as C A/B B\C.

Type-raising swaps the functor-argument relation. Although it is often assumed to take place in the lexicon, we will distinguish lexical categories (e.g. for quantifiers) that have the syntactic type of type-raised categories, but semantics that could not be obtained by type-raising a simple category from grammatically type-raised categories. We follow the common definition of CCG (Steedman, 2000) and allow only categories $X \in \mathcal{C}_{arg}$ to be type-raised.[4] Instantia-

[3] In $X|Y_{1..n}$ or $X|\alpha$=$X|Y_{1...|\alpha|}$, we do not assume the slash variable $| \in \{/, \backslash\}$ to be instantiated the same way for all Y_i. We will therefore only distinguish between forward and backward generalized composition $\mathbf{B}^{n>1}$.

[4] We stipulate that it may be further necessary to only allow those argument categories to type-raise that are not used to project unbounded dependencies, such as S/NP in

tions of the variable T should also be restricted to categories of finite arity $N_{\mathbf{T}}$ in oder to prevent an increase in generative capacity (Hoffman, 1995; Komagata, 1997). We refer to the arity of T as the degree of any particular instantation of $\mathbf{T}$. We follow Steedman (2000) and assume $N_{\mathbf{T}} = N_{\mathbf{B}}$.

Coordination requires a ternary rule (Φ) which can be binarized (Φ>, Φ<) to simplify parsing:[5]

$$\begin{array}{llll}
(\Phi) & \mathsf{X}\ \ \mathsf{conj} & \mathsf{X} & \Rightarrow \mathsf{X} \\
(\Phi>) & \mathsf{X} & \mathsf{X[conj]} & \Rightarrow \mathsf{X} \\
(\Phi<) & \mathsf{conj} & \mathsf{X} & \Rightarrow \mathsf{X[conj]}
\end{array}$$

Uses of type-raising and composition In English, type-raising and composition are required for wh-extraction and right node raising of arguments as well as so-called argument cluster coordination. In other languages, they are needed for scrambling and cross-serial dependencies.

It is important to note that when type-raising is *required*, it always occurs in tandem with composition. Since type-raising an argument Y to $\mathsf{X/(X\backslash Y)}$ and applying it to the functor $\mathsf{X\backslash Y}$ is semantically equivalent to applying $\mathsf{X\backslash Y}$ directly to Y, type-raising is never required when function application can be used instead. That is, in all cases, a type-raised argument must be composed with another constituent, usually the original functor (head). Only in argument-cluster coordination will the type-raised element be composed with a non-head constituent. In the latter case, coordination will be required before the argument cluster can be combined with the head. Composition without type-raising may occur, e.g. for adjuncts, which have categories $\mathsf{X|X}$, but may modify a constituent with category $\mathsf{X}|\alpha$.

Restrictions on type-raising and composition In order to prevent overgenerations of the form *"John speaks because Chinese, he enjoys Beijing."*, we assume a variant of CCG in which forward crossing composition $>\mathbf{B}^{1}_{\times}$ (e.g. of *because*:$\mathsf{(S/S)/S}$) into the result of backward type-raising $<\mathbf{T}$ (e.g. *Chinese*:$\mathsf{S\backslash(S/NP)}$, and, similarly, $<\mathbf{B}^{x}$ into the result of $>\mathbf{T}$, are disallowed.

Punctuation and Type-changing rules CCGbank (Hockenmaier and Steedman, 2007) uses special punctuation rules such as $\mathsf{S}\ .\Rightarrow \mathsf{S}$ or $,\ \mathsf{NP\backslash NP} \Rightarrow \mathsf{NP\backslash NP}$, and a small number of (non-recursive) type-changing rules (with idiosyncratic semantics) such as $\mathsf{N} \Rightarrow \mathsf{NP}$ (for determiner-less NPs) or $\mathsf{S[pss]\backslash NP} \Rightarrow \mathsf{NP\backslash NP}$ (for complex adjuncts, here passive VPs being used as NP postmodifiers):

$$\begin{array}{lllll}
\text{Punctuation} & (>\mathbf{P}) & \mathsf{X}{:}\phi\ [.,;] & \Rightarrow \mathsf{X}{:}\phi \\
 & (<\mathbf{P}) & [.,;]\ \mathsf{X}{:}\phi & \Rightarrow \mathsf{X}{:}\phi \\
\text{TypeChanging} & (\mathbf{TCR}) & \mathsf{X}{:}\phi & \Rightarrow \mathsf{Y}{:}\psi(\phi)
\end{array}$$

CCG parsing CCG can be parsed with a bottom-up CKY-like algorithm (Shieber et al., 1995; Steedman, 2000), which differs from standard CKY in that it requires one (or two) unary completion steps in each cell to deal with type-raising (and type changing).[6] Chart items are of the form $\langle \mathsf{X}, i, j\rangle$, where X is a category, and the indices i and j represent the span of the item. Interpretations need only to be constructed for complete derivations when unpacking the chart.

3 The Eisner normal form

The Eisner normal form Eisner (1996) presents a normal-form parsing algorithm for CCG without grammatical type raising (where the lexicon may still contain categories like $\mathsf{S/(S\backslash NP)}$, but there is no combinatory rule that changes a complex (derived) NP to e.g. $\mathsf{S/(S\backslash NP)}$). He proves that his algorithm finds only one canonical derivation for each semantic interpretation of an input string consisting of a sequence of words and their lexical categories. Since the presence of both pre- and postmodifiers (as in *"intentionally knock twice"*[7]) introduces a genuine ambiguity, Eisner proves that the only kind of spurious ambiguity that can arise in his variant of CCG is due to associative chains of composition such as $\mathsf{A/B\ B/C\ C/D}$ or $\mathsf{A/B\ B/C\ C\backslash D}$, which can be derived as either

$\mathsf{(NP\backslash NP)/(S/NP)}$ for English object relative pronouns.

[5] Here, X needs to be restricted to a finite set of categories (Weir, 1988). In multimodal CCG, conjunction have categories of the form $\mathsf{(X_\star\backslash_\star X)/_\star X}$, i.e. must apply to their argument

[6] Since composition allows the arity of derived ($\approx$ non-terminal) CCG categories to grow with the length of the input string, worst-case complexity of this naive algorithm is exponential. (Vijay-Shanker and Weir, 1993)'s $O(n^6)$ algorithm has a more compact representation of categories.

[7] This can mean $\lambda x.intentionally'(twice'(knock'(x)))$ or $\lambda x.twice'(intentionally'(knock'(x)))$.

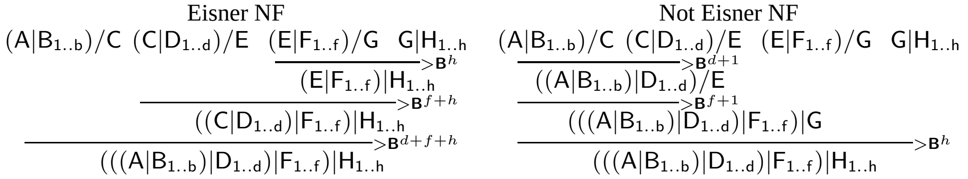

Eisner NF
$(A|B_{1..b})/C$ $(C|D_{1..d})/E$ $(E|F_{1..f})/G$ $G|H_{1..h}$
$>\mathbf{B}^{h}$
$(E|F_{1..f})|H_{1..h}$
$>\mathbf{B}^{f+h}$
$((C|D_{1..d})|F_{1..f})|H_{1..h}$
$>\mathbf{B}^{d+f+h}$
$(((A|B_{1..b})|D_{1..d})|F_{1..f})|H_{1..h}$
Not Eisner NF
$(A|B_{1..b})/C$ $(C|D_{1..d})/E$ $(E|F_{1..f})/G$ $G|H_{1..h}$
$>\mathbf{B}^{d+1}$
$((A|B_{1..b})|D_{1..d})/E$
$>\mathbf{B}^{f+1}$
$(((A|B_{1..b})|D_{1..d})|F_{1..f})|G$
$>\mathbf{B}^{h}$
$(((A|B_{1..b})|D_{1..d})|F_{1..f})|H_{1..h}$

Figure 3: Eisner NF and generalized composition $\mathbf{B}^{n>1}$

Left branching		Right branching		
$>\mathbf{B}^{0}(>\mathbf{B}^{m+1},...)$	$\Rightarrow$	$>\mathbf{B}^{m\geq 0}(...,>\mathbf{B}^{0})$	$A/B\ (B\vert D_{0..m})/C\ C$	$m \geq 0$
$>\mathbf{B}^{1}(>\mathbf{B}^{m\geq 1},...)$	$\Rightarrow$	$>\mathbf{B}^{m\geq 1}(...,>\mathbf{B}^{1})$	$A/B\ (B\vert C_{1...m-1})/D\ D/E$	$m \geq 1$
$>\mathbf{B}^{n\geq 1}(>\mathbf{B}^{1},...)$	$\Rightarrow$	$>\mathbf{B}^{n}(...,>\mathbf{B}^{m=n})$	$A/B\ B/C\ C/D_{1..n}$	$m = n \geq 1$
$\emptyset$	$\not\Leftarrow$	$>\mathbf{B}^{n>1}(...,>\mathbf{B}^{m>n})$	$A/(B\vert D_{1..k})\ B/C\ ((C\vert D_{1..k})\vert E_{1..n}$	$m > n > 1$
$>\mathbf{B}^{m}(>\mathbf{B}^{k},...)$	$\Leftarrow$	$>\mathbf{B}^{n>1}(...,>\mathbf{B}^{1<m<n})$	$A/B\ (B\vert C_{1..k-1})/D\ D\vert E_{1..m}$	$n > m > 1$

Figure 4: Associative composition chains: our NF disallows the grayed-out derivations.

$>\mathbf{B}$ (..., $>\mathbf{B}$) or $>\mathbf{B}$ ($>\mathbf{B}$, _). This is eliminated by the following constraint:

Eisner NF Constraint 1. *The output* $X|\alpha$ *of forward composition* $>\mathbf{B}^{n>0}$ *cannot be the primary input to forward application or composition* $>\mathbf{B}^{m\geq 0}$. *The output of* $<\mathbf{B}^{n>0}$ *cannot be the primary input to* $<\mathbf{B}^{m\geq 0}$.

This can be implemented by a ternary feature $H_E \in \{>\mathbf{B}^{n},\ <\mathbf{B}^{n},\ \emptyset\}$ and chart items of the form $\langle X,\ H_E,\ i,\ j\rangle$ where $H_E = >\mathbf{B}^{n}$ (or $<\mathbf{B}^{n}$) if X was produced by the corresponding composition rule (for any $n > 0$) and $\emptyset$ otherwise.

4 A new normal form for CCG

4.1 Generalized composition

Eisner NF and generalized composition Unboundedly long sequences of generalized composition are required e.g. for Dutch verb clusters that give rise to cross-serial dependencies ($N_1...N_nV_1...V_n$ with N_i the argument of V_i). These can be obtained through standard bounded-degree compositions, but the Eisner NF produces a derivation that requires compositions of unbounded degree (Fig. 3). Although this is allowed in the variant of CCG Eisner considers, compositions of unbounded degree are usually disallowed because they increase the generative capacity of CCG (Weir, 1988). We stipulate that the NF of any derivation τ should not require composition rules of higher degree than τ itself. Note that the output of function application ($\mathbf{B}^0$) always has lower arity than its functor; the output of regular composition ($\mathbf{B}^1$) has the same arity as its primary functor, but the output of generalized composition ($\mathbf{B}^{n>1}$) has an arity that is $n-1$ higher than that of the primary functor. $\mathbf{B}^{n>1}$ therefore requires a different treatment.

Our reformulation of the Eisner NF Associative composition chains for constituents A B C can lead to spurious ambiguity if both a left-branching $>\mathbf{B}^{n}(>\mathbf{B}^{m}(A\ B)\ C)$ and a right-branching $>\mathbf{B}^{n'}(A\ >\mathbf{B}^{m'}(B\ C))$ are possible and lead to the same interpretation. Figure 4 illustrates all possible cases consisting of three constituents. In most cases, the right-branching (Eisner NF) derivation is to be preferred. For generalized composition $>\mathbf{B}^{n>1}$, $>\mathbf{B}^{m>1}$, left-branching $>\mathbf{B}^{n>1}(>\mathbf{B}^{m>1}, ...)$ is always allowed, but right-branching $>\mathbf{B}^{n}(..., >\mathbf{B}^{m})$ is only allowed if $m \geq n$.

NF Constraint 1 ($\mathbf{B}^0$ and $\mathbf{B}^{n\geq 1}$). *The output of* $>\mathbf{B}^{n\geq 1}$ *(resp.* $<\mathbf{B}^{n\geq 1}$*) cannot be primary functor for* $>\mathbf{B}^{n\leq 1}$ *(resp.* $<\mathbf{B}^{n\leq 1}$*).*

NF Constraint 2 ($\mathbf{B}^1$ and $\mathbf{B}^{n\geq 1}$). *The output of* $>\mathbf{B}^{1}$ *(resp.* $<\mathbf{B}^{1}$*) cannot be primary functor for* $>\mathbf{B}^{n\geq 1}$ *(resp.* $<\mathbf{B}^{n\geq 1}$*).*

NF Constraint 3 ($\mathbf{B}^{n>1}$ and $\mathbf{B}^{m>1}$). *The output of* $>\mathbf{B}^{m}$ *(resp.* $<\mathbf{B}^{m}$*) cannot be secondary functor for* $>\mathbf{B}^{n>m}$ *(resp.* $<\mathbf{B}^{n>m}$*).*

4.2 Grammatical type-raising

Eisner NF and type-raising Figure 5 illustrates a spurious ambiguity arising through type-

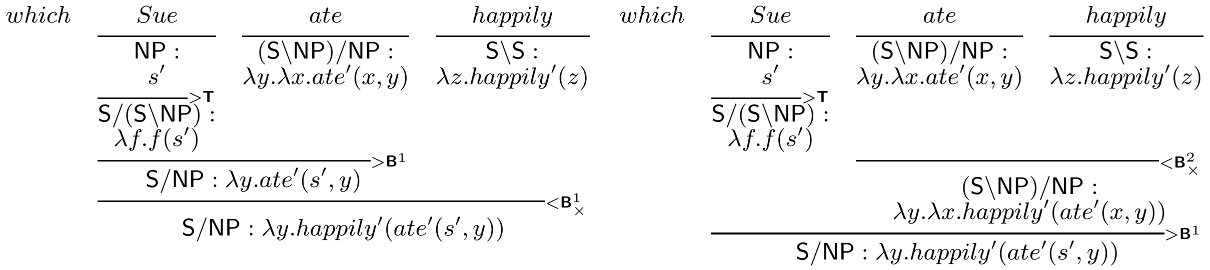
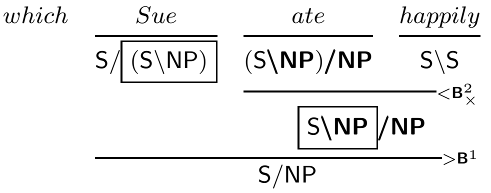


Figure 5: The Eisner NF allows spurious ambiguity arising due to type-raising

raising that the Eisner NF does not exclude.[8] Here two derivations can be obtained because the result of combining the adverb with the subject-verb cluster is no longer the output of a forward composition, and can therefore apply to the object. The derivations are semantically equivalent: although type-raising reverses the syntactic functor-argument relation, a type-raised argument applied to a predicate returns the same interpretation as when the predicate is applied directly to the original. But Eisner treats S/(S\NP) as a category with semantics $\lambda x.\phi(x)$, in which case the derivations yield indeed different scope relations. Eisner's analyis is correct for certain classes of words which have lexical categories that appear like type-raised categories, but have a different interpretation from that of categories obtained by type-raising. These are usually scope-bearing elements, such as the universal quantifer *every* ((S/(S\NP))/N : λPλQ∀xP(x) → Q(x)), and there may not be a single derivation which captures all semantic interpretations. Lexicalized pseudo-type-raising therefore needs to be distinguished from grammatical type-raising.

Our extension of the (modified) Eisner NF In Fig. 5, Eisner NF licenses two derivations. Both contain an instance of composition in which the type-raised argument is the primary component. In the analysis in which this is the second derivation step, the canceled part of this <**B**^{2} composition (boxed) contains a category (\NP) that was part of the argument output of the first >**B**^{1} composition (bold-faced):



Our NF will eliminate derivations of this type and prefer the other, lower-degree derivation. We stipulate that the spurious ambiguities that arise through type-raising and composition can be eliminated through the following rule:

NF Constraint 4 (**T** and **B**$^{n>0}$). *The output of* >**T** *cannot be primary input to* >**B**$^{n>0}$ *if the secondary input is the output of* <**B**$^{m>n}$. *The output of* <**T** *cannot be primary input in* <**B**$^{n>0}$ *if the secondary input is the output of* >**B**$^{m>n}$.

We also stipulate that a type-raised T/(T\X) cannot be used as a functor in application (since T\X could always apply directly to X).

NF Constraint 5 (**T** and **B**^{0}). *The output of forward (or backward) type-raising* >**T** *(resp.* <**T** *) cannot be the functor in application* > *(resp.* <*).*

Additional spurious ambiguities arise through the interaction of type-raising and coordination: Since any category can be coordinated, we can either coordinate X and then type-raise the coordinated X to T/(T\X), or we can first type-raise each conjunct to T/(T\X) and then conjoin. Since nonsymmetric coordinations of an argument-adjunct cluster and a single argument (as in *eats ((pizza for lunch) and pasta)*) require type-raising before coordination, we formulate the following rule to eliminate interactions between type-raising and coordination:

NF Constraint 6 (**T** and Φ). *The result of coordination* Φ *cannot be type-raised.*

[8] We have chosen a slighly unusual adverb category to illustrate a general problem.

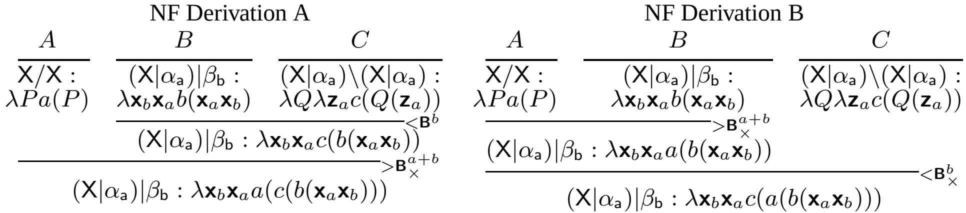

NF Derivation A
A
B
C
$\mathsf{X/X}$: $\lambda P a(P)$
$(\mathsf{X}|\alpha_{\mathsf{a}})|\beta_{\mathsf{b}}$: $\lambda \mathbf{x}_b \mathbf{x}_a b(\mathbf{x}_a \mathbf{x}_b)$
$(\mathsf{X}|\alpha_{\mathsf{a}})\backslash(\mathsf{X}|\alpha_{\mathsf{a}})$: $\lambda Q \lambda \mathbf{z}_a c(Q(\mathbf{z}_a))$
$<\mathbf{B}^b$
$(\mathsf{X}|\alpha_{\mathsf{a}})|\beta_{\mathsf{b}} : \lambda \mathbf{x}_b \mathbf{x}_a c(b(\mathbf{x}_a \mathbf{x}_b))$
$>\mathbf{B}^{a+b}_{\times}$
$(\mathsf{X}|\alpha_{\mathsf{a}})|\beta_{\mathsf{b}} : \lambda \mathbf{x}_b \mathbf{x}_a a(c(b(\mathbf{x}_a \mathbf{x}_b)))$
NF Derivation B
A
B
C
$\mathsf{X/X}$: $\lambda P a(P)$
$(\mathsf{X}|\alpha_{\mathsf{a}})|\beta_{\mathsf{b}}$: $\lambda \mathbf{x}_b \mathbf{x}_a b(\mathbf{x}_a \mathbf{x}_b)$
$(\mathsf{X}|\alpha_{\mathsf{a}})\backslash(\mathsf{X}|\alpha_{\mathsf{a}})$: $\lambda Q \lambda \mathbf{z}_a c(Q(\mathbf{z}_a))$
$>\mathbf{B}^{a+b}_{\times}$
$(\mathsf{X}|\alpha_{\mathsf{a}})|\beta_{\mathsf{b}} : \lambda \mathbf{x}_b \mathbf{x}_a a(b(\mathbf{x}_a \mathbf{x}_b))$
$<\mathbf{B}^b_{\times}$
$(\mathsf{X}|\alpha_{\mathsf{a}})|\beta_{\mathsf{b}} : \lambda \mathbf{x}_b \mathbf{x}_a c(a(b(\mathbf{x}_a \mathbf{x}_b)))$

Figure 6: Constituents with pre- and postmodifiers have two semantically distinct derivations

Punctuation and Type-changing rules Punctuation results in spurious ambiguities, either when a constituent X has both an initial and a final punctuation mark (e.g. a comma), or when it has an initial (final) punctuation mark and a final (initial) modifier. The first case is easy to fix by disallowing the output of , X $\Rightarrow$ X to be the input of X , $\Rightarrow$ X. The latter could be eliminated by disallowing the output X of right-recursive (left-recursive) punctuation rule to be secondary input to any left-recursive (right-recursive) application or composition rule (e.g. X X\X $\Rightarrow$ X).[9]

Implementation Our normal-form constraints can be implemented in a bottom-up parser with items of the form $\langle \mathsf{X}, \mathcal{C}, i, j \rangle$, with

$$\mathcal{C} \in \{>, >\mathbf{B}\,1, >\mathbf{B}\,2, ..., >\mathbf{B}^n; <, <\mathbf{B}\,1, <\mathbf{B}\,2, ..., <\mathbf{B}^n; >\mathbf{T}\,, <\mathbf{T}\,, >\mathsf{Pct}, <\mathsf{Pct}, \Phi>, \Phi<, \mathsf{TCR}\}$$

4.3 Is our normal form safe and complete?

Here we sketch the beginnings of a proof that our algorithm allows one and only one syntactic derivation per semantic interpretation for the version of CCG we consider. We first examine all cases of two adjacent constituents A, B which must combine into a category C:

Functor X/Y and argument Y combine to X The functor must apply to the argument. The argument could type-raise, but then cannot apply.

Functor $\mathsf{X/Y}|\alpha$ and argument Y combine to $\mathsf{X}|\alpha$ The functor cannot apply to the argument. The argument must type-raise to X\(X/Y), and can then backward-compose into the functor.

Functor X/X and X\X can combine to X/X or X\X This is not a spurious ambiguity, since the output categories are different.

Functor X|Y and Y|Z combine to X|Z Our reformulation of Eisner's NF eliminates spurious ambiguities that are due to such associative composition chains. This covers not only argument clusters (which must compose), but also ambiguous cases where one constituent (e.g. Y/Z with $\alpha = \epsilon$) is the argument of the first (X/Y), and either takes the third (Z) as its own argument or is modified by the third Y\Y (there are, of course, other arrangements of such categories which are not ambiguous, e.g. X/Y Z Y\Z.

We now focus our attention on the ternary cases in which one of the constituents is a head (predicate), and the other two are either its arguments or modifiers. The counterexample to Eisner's normal-form algorithm shows that there is at least one additional kind of spurious ambiguity that arises when there are three adjacent constituents A, B, C and both A and C can compose into B. There are three cases: 1) A and C are both modifiers of B, 2) one of A or C is a modifier of B, the other is an argument of B, and 3) A and C are both arguments of B. Only 1) is a real ambiguity, but the other cases are instances of spurious ambiguity which our NF eliminates.

Argument Y, head (X\Y)/Z and argument Z combine to X In the NF derivation, the head applies first to the Z, than to Y. All other derivations are blocked, either because type-raised categories cannot apply, or because the output of composition cannot apply.

Modifier X/X, head $(\mathsf{X}|\alpha)|\beta$ and modifier $(\mathsf{X}|\alpha)\backslash(\mathsf{X}|\alpha)$ **combine to** $(\mathsf{X}|\alpha)|\beta$ (Fig. 4.2). This is the *"intentionally knock twice"* example. The derivations have different semantics.

Argument Y, head $((\mathsf{X}|\alpha)\backslash\mathsf{Y})|\beta$**, and modifier** X\X **combine to** $(\mathsf{X}|\alpha)|\beta$ (Fig. 7). If there is an ambiguity, B must have a category of the form

[9] If punctuation can be used both with X and Y, it also interacts with type-changing rules X $\Rightarrow$ Y. Our current implementation does not deal with this case.

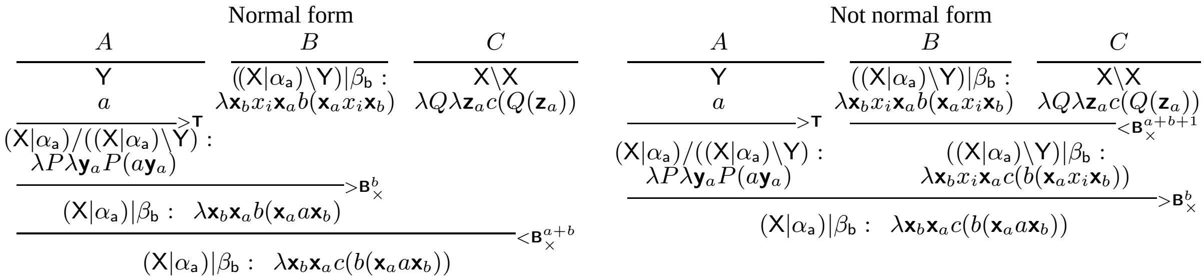

Normal form
A B C
Y a
$((\mathsf{X}|\alpha_\mathsf{a})\backslash\mathsf{Y})|\beta_\mathsf{b}$: $\lambda\mathbf{x}_b x_i \mathbf{x}_a b(\mathbf{x}_a x_i \mathbf{x}_b)$
$\mathsf{X}\backslash\mathsf{X}$ $\lambda Q\lambda\mathbf{z}_a c(Q(\mathbf{z}_a))$
$>\mathbf{T}$
$(\mathsf{X}|\alpha_\mathsf{a})/((\mathsf{X}|\alpha_\mathsf{a})\backslash\mathsf{Y})$: $\lambda P\lambda\mathbf{y}_a P(a\mathbf{y}_a)$
$>\mathbf{B}^b_\times$
$(\mathsf{X}|\alpha_\mathsf{a})|\beta_\mathsf{b}$: $\lambda\mathbf{x}_b\mathbf{x}_a b(\mathbf{x}_a a\mathbf{x}_b)$
$<\mathbf{B}^{a+b}_\times$
$(\mathsf{X}|\alpha_\mathsf{a})|\beta_\mathsf{b}$: $\lambda\mathbf{x}_b\mathbf{x}_a c(b(\mathbf{x}_a a\mathbf{x}_b))$
Not normal form
A B C
Y a
$((\mathsf{X}|\alpha_\mathsf{a})\backslash\mathsf{Y})|\beta_\mathsf{b}$: $\lambda\mathbf{x}_b x_i \mathbf{x}_a b(\mathbf{x}_a x_i \mathbf{x}_b)$
$\mathsf{X}\backslash\mathsf{X}$ $\lambda Q\lambda\mathbf{z}_a c(Q(\mathbf{z}_a))$
$>\mathbf{T}$
$<\mathbf{B}^{a+b+1}_\times$
$(\mathsf{X}|\alpha_\mathsf{a})/((\mathsf{X}|\alpha_\mathsf{a})\backslash\mathsf{Y})$: $\lambda P\lambda\mathbf{y}_a P(a\mathbf{y}_a)$
$((\mathsf{X}|\alpha_\mathsf{a})\backslash\mathsf{Y})|\beta_\mathsf{b}$: $\lambda\mathbf{x}_b x_i\mathbf{x}_a c(b(\mathbf{x}_a x_i\mathbf{x}_b))$
$>\mathbf{B}^b_\times$
$(\mathsf{X}|\alpha_\mathsf{a})|\beta_\mathsf{b}$: $\lambda\mathbf{x}_b\mathbf{x}_a c(b(\mathbf{x}_a a\mathbf{x}_b))$

Figure 7: Argument Y, head $((\mathsf{X}|\alpha_\mathsf{a})\backslash\mathsf{Y})|\beta_\mathsf{b}$, and modifier $\mathsf{X}\backslash\mathsf{X}$ combine to $(\mathsf{X}|\alpha_\mathsf{a})|\beta_\mathsf{b}$

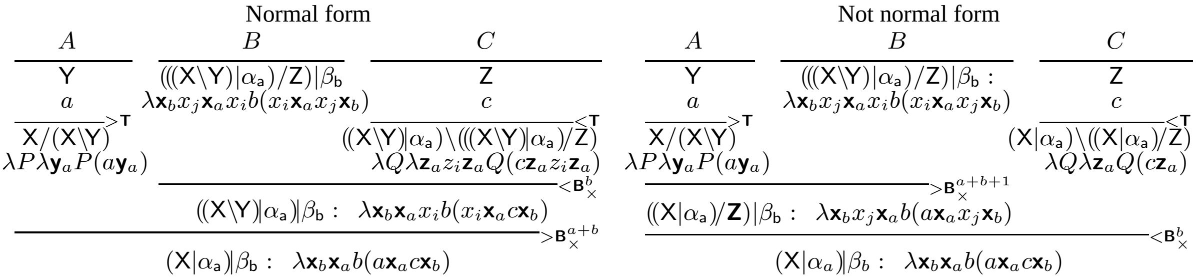

Normal form
A B C
Y a
$(((\mathsf{X}\backslash\mathsf{Y})|\alpha_\mathsf{a})/\mathsf{Z})|\beta_\mathsf{b}$ $\lambda\mathbf{x}_b x_j\mathbf{x}_a x_i b(x_i\mathbf{x}_a x_j\mathbf{x}_b)$
Z c
$>\mathbf{T}$
$<\mathbf{T}$
$\mathsf{X}/(\mathsf{X}\backslash\mathsf{Y})$ $\lambda P\lambda\mathbf{y}_a P(a\mathbf{y}_a)$
$((\mathsf{X}\backslash\mathsf{Y})|\alpha_\mathsf{a})\backslash(((\mathsf{X}\backslash\mathsf{Y})|\alpha_\mathsf{a})/\mathsf{Z})$ $\lambda Q\lambda\mathbf{z}_a z_i\mathbf{z}_a Q(c\mathbf{z}_a z_i\mathbf{z}_a)$
$<\mathbf{B}^b_\times$
$((\mathsf{X}\backslash\mathsf{Y})|\alpha_\mathsf{a})|\beta_\mathsf{b}$: $\lambda\mathbf{x}_b\mathbf{x}_a x_i b(x_i\mathbf{x}_a c\mathbf{x}_b)$
$>\mathbf{B}^{a+b}_\times$
$(\mathsf{X}|\alpha_\mathsf{a})|\beta_\mathsf{b}$: $\lambda\mathbf{x}_b\mathbf{x}_a b(a\mathbf{x}_a c\mathbf{x}_b)$
Not normal form
A B C
Y a
$(((\mathsf{X}\backslash\mathsf{Y})|\alpha_\mathsf{a})/\mathsf{Z})|\beta_\mathsf{b}$: $\lambda\mathbf{x}_b x_j\mathbf{x}_a x_i b(x_i\mathbf{x}_a x_j\mathbf{x}_b)$
Z c
$>\mathbf{T}$
$<\mathbf{T}$
$\mathsf{X}/(\mathsf{X}\backslash\mathsf{Y})$ $\lambda P\lambda\mathbf{y}_a P(a\mathbf{y}_a)$
$(\mathsf{X}|\alpha_\mathsf{a})\backslash((\mathsf{X}|\alpha_\mathsf{a})/\mathsf{Z})$ $\lambda Q\lambda\mathbf{z}_a Q(c\mathbf{z}_a)$
$>\mathbf{B}^{a+b+1}_\times$
$((\mathsf{X}|\alpha_\mathsf{a})/\mathbf{Z})|\beta_\mathsf{b}$: $\lambda\mathbf{x}_b x_j\mathbf{x}_a b(a\mathbf{x}_a x_j\mathbf{x}_b)$
$<\mathbf{B}^b_\times$
$(\mathsf{X}|\alpha_a)|\beta_b$: $\lambda\mathbf{x}_b\mathbf{x}_a b(a\mathbf{x}_a c\mathbf{x}_b)$

Figure 8: Argument Y, head $(((\mathsf{X}\backslash\mathsf{Y})|\alpha)/\mathsf{Z})|\beta$ and argument Z combine to $(\mathsf{X}|\alpha)|\beta$

$((\mathsf{X}|\alpha)\backslash\mathsf{Y}_\mathsf{i})|\beta$ (with X possibly complex and α, β possibly empty), and C must have a category of the form $\mathsf{X}\backslash\mathsf{X}$. We obtain the NF derivation by first combining head and argument, followed by the modifier. The other derivation violates the NF constraints.

Argument Y, head $(((\mathsf{X}\backslash\mathsf{Y})|\alpha)/\mathsf{Z})|\beta$ and argument Z combine to $(\mathsf{X}|\alpha)|\beta$ (Fig. 8) The derivation in which Z composes first is in NF. The derivation in which the Y combines first with the head is blocked.

Arguments Y_A, Y_B, head $(((\mathsf{X}\backslash\mathsf{Y}_1)|\alpha)\backslash\mathsf{Y}_2)|\beta$ combine to $(\mathsf{X}|\alpha)|\beta$ There are two readings: standard (Y_A:=Y_1, Y_B:=Y_2), and scrambled (Y_A:=Y_2, Y_B:=Y_1). If α and β are empty, function application is sufficient for the standard reading, and our NF constraint 1 excludes the 'argument cluster' derivation in which both Y_A and Y_B type-raise, compose and then apply to the head. Otherwise, at least one of the arguments has to type-raise and compose into the head. If both α and β are non-empty, each interpretation has only one derivation in which the type-raised Y_A composes into the output of the composition of the type-raised Y_B with the head. Since the degree of the second composition is lower than the first, this is allowed by our NF constraint 2.

Argument Y_A and heads $(((\mathsf{X}\backslash\mathsf{Y}_1)|\alpha)/\mathsf{Z}$ and $((\mathsf{Z}|\beta)\backslash\mathsf{Y}_2)|\gamma$ combine to $(((\mathsf{X}|\alpha)|\beta)\backslash\mathsf{Y}_2)|\gamma$ or to $(((\mathsf{X}|\backslash\mathsf{Y}_1\alpha)|\beta)|\gamma$ There are two readings: standard (Y_A:=Y_1) or scrambled (Y_A:=Y_2). Depending on the maximal degree n of $\mathbf{B}^n$ allowed by the grammar, the standard reading one can either be obtained by type-raising Y_A and composing into the first head (allowed by our NF) or by first composing the two heads and then composing the type-raised Y_A into the cluster (allowed by Eisner, but not by us). The second reading requires the heads to compose and then Y_A to apply or compose (depending on the arity of γ), which is allowed by our NF constraint 2 because the degree of this second composition is lower than that of the first.

Our NF and the bound $N_\mathbf{T}$ on type-raising If $\mathsf{X}\backslash\mathsf{X}$ in Fig. 7 is replaced with a (non-type-raised) category $\mathsf{Z}\backslash\mathsf{X}$ (for $\mathsf{Z} \neq \mathsf{X}$), the non-NF derivation requires $\mathbf{T}^{|Z|+a}$, whereas the NF-derivation requires $\mathbf{T}^{|X|+a}$. If we stipulate a finite bound $N_\mathbf{T}$ on the degree of type-raising, and if $|X| > |Z|$ and $|X| + a > N_\mathbf{T}$, our NF cannot be derived anymore. If such $\mathsf{Z}\backslash\mathsf{X}$ (with $\mathsf{X} \in \mathcal{C}_{arg}$) can be derived from the lexicon, our NF requires therefore a potentially unbounded degree of type-raising. The $\mathbf{T}$-degree

	Sentence length l=15...30			
	15	20	25	30
No NF (total #derivs)	4.13E6	5.66E8	3.06E11	1.59E14
Eisner **B**	18.92%	9.05%	3.63%	2.14%
Our **B**	18.38%	8.97%	3.60%	2.02%
Our **B** , **T**	2.92%	1.22%	0.37%	0.10%
Our full NF	2.60%	0.93%	0.33%	0.09%

(a) Median % of allowed derivations

	Sentence length l= 30			
	Min	Mean	Median	Max
No NF	5.99E9	8.19E15	1.59E14	2.61E17
Eisner **B**	1.60%	2.68%	2.14%	2.76%
Our **B**	1.57%	2.49%	2.02%	2.69%
Our **B** ,**T**	0.64%	0.07%	0.10%	0.05%
Our full NF	0.53%	0.06%	0.09%	0.05%

(b) Statistics on the % of allowed derivations

Figure 9: Experimental results: the effect of different normal forms on the number of derivations

of the non-NF derivation in Fig. 8 is also one less than that of the NF derivation, but its **B**-degree is increased by one, so for $N_{\mathbf{T}} = N_{\mathbf{B}}$ either both derivations are possible or neither.

What remains to be proven is that we have considered all cases of spurious ambiguity involving three constituents, and that all cases of spurious ambiguity that arise for more than three constituents reduce to these cases.

5 The effects of normal form parsing

We now illustrate the impact of the different normal form variants on a small, restricted, grammar. We define a set of atomic categories, a set of lexical categories (up to a fixed arity N_{Lex}), and compile out all possible rule instantiations (including compositions up to a fixed degree $N_{|B}$) that generate categories up to a fixed arity N_{cat}[10]

The effect of different normal forms This experiment is intended to examine how normal form parsing might reduce spurious ambiguity for actual grammars, e.g. for unsupervised estimation of stochastic CCGs. We created a small English grammar with atomic categories S, NP, N, conj, ., , ; and 47 lexical categories using $N_{Lex} = 3$, $N_{\mathbf{B}} = 3$, $N_{Cat} = 15$. There are two type-changing rules (N ⇒ NP and S/NP ⇒ NP\NP). We accept derivations of S, NP and S\NP. The T|X in **T** has to be a lexical category. Our lexical categories are divided into disjoint sets of adjuncts of the form X|X and (X|X)|Y, head (both atomic and complex), and punctuation and conjunction categories. The comma can act as a conjunction or to set off modifiers (requiring punctuation rules of the form X|X , ⇒ X|X and , X|X ⇒ X|X). We furthermore define coarse-grained parts of speech (noun, verb, function word, conj, other) and decide for each part of speech which lexical categories it can take. We compare different NF settings for sentences of lengths 15–30 from Europarl (Koehn, 2005). At each length, we compare 100 sentences that our grammar can parse. All NFs can parse all sentences the full grammar can parse. Results (Fig. 9(a)) show that our NF reduces the number of derivations significantly over Eisner's NF, even though our (full) grammar only allows a restricted set of type-raising rules. Fig. 9(b) illustrates the combinatorial explosion of spurious derivations as the sentence length increases.

6 Conclusions

We have proposed a modification and extension of Eisner (1996)'s normal form that is more appropriate for commonly used variants of CCG with grammatical type-raising and generalized composition of bounded degree, as well as some non-combinatory extensions to CCG. Our experiments indicate that incorporating normal form constraints to deal with grammatical type-raising drastically reduces the number of derivations. We have sketched the outline of a proof that our normal form is safe and complete for the variant of CCG we consider, althoug we have seen that under certain circumstances, type-raising of unbounded degree may be required. Future work will investigate this issue further, and will also aim to turn our informal arguments about the adequacy of our approach into a full proof, and provide more experiments on a wider range of grammars and languages.

[10] The restriction of categories to a fixed arity means that we could generate cross-serial dependencies $N_1 ... N_n V_1 ... V_n$ only up to $n = A_{cat}$.

References

Boxwell, Stephen, Dennis Mehay, and Chris Brew. 2009. Brutus: A semantic role labeling system incorporating CCG, CFG, and dependency features. In *Proceedings of the 47th ACL/4th IJCNLP*, pages 37–45.

Clark, Stephen and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Curran, James, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and boxer. In *Proceedings of the 45th ACL Companion Volume (Demo and Poster Sessions)*, pages 33–36, Prague, Czech Republic.

Eisner, Jason. 1996. Efficient normal-form parsing for Combinatory Categorial Grammar. In *Proceedings of the 34th ACL*, pages 79–86, Santa Cruz, CA.

Espinosa, Dominic, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of ACL-08: HLT*, pages 183–191, Columbus, Ohio.

Gildea, Daniel and Julia Hockenmaier. 2003. Identifying semantic roles using Combinatory Categorial Grammar. In *Proceedings of EMNLP*, Sapporo, Japan.

Hepple, Mark and Glyn Morrill. 1989. Parsing and derivational equivalence. In *Proceedings of the Fourth EACL*, pages 10–18, Manchester, UK.

Hockenmaier, Julia and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396.

Hockenmaier, Julia. 2003. *Data and models for statistical parsing with Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Hoffman, Beryl. 1995. *Computational Analysis of the Syntax and Interpretation of 'Free' Word-order in Turkish*. Ph.D. thesis, University of Pennsylvania. IRCS Report 95-17.

Hoyt, Frederick and Jason Baldridge. 2008. A logical basis for the D combinator and normal form in CCG. In *Proceedings of ACL-08: HLT*, pages 326–334, Columbus, Ohio.

Karttunen, Lauri. 1989. Radical lexicalism. In Baltin, M.R. and A.S. Kroch, editors, *Alternative Conceptions of Phrase Structure*. Chicago University Press, Chicago.

Koehn, Philipp. 2005. Europarl: A parallel corpus for statistical machine translation. In *10th MT Summit*, pages 79–86, Phuket, Thailand.

Komagata, Nobo. 1997. Generative power of CCGs with generalized type-raised categories. In *ACL35/EACL8 (Student Session)*, pages 513–515.

Komagata, Nobo. 2004. A solution to the spurious ambiguity problem for practical combinatory categorial grammar parsers. *Computer Speech & Language*, 18(1):91 – 103.

Niv, Michael. 1994. A psycholinguistically motivated parser for CCG. In *Proceedings of The 32nd ACL*, Las Cruces, NM, pages 125–132.

Pareschi, Remo and Mark Steedman. 1987. A lazy way to chart parse with categorial grammars. In *Proceedings of the 25th ACL*, pages 81–88, Stanford, CA.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36, July–August.

Steedman, Mark. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Vijay-Shanker, K and David J Weir. 1993. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.

Weir, David. 1988. *Characterising Mildly Context-sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania. Tech. Report CIS-88-74.

Wittenburg, Kent B. 1986. *Natural Language Parsing with Combinatory Categorial Grammar in a Graph-Unification Based Formalism*. Ph.D. thesis, University of Texas at Austin.

Zettlemoyer, Luke S. and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st UAI*, pages 658–666, Edinburgh, UK.

Zettlemoyer, Luke and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of EMNLP-CoNLL*, pages 678–687, Prague, Czech Republic.

Acknowledgements

We would like to thank Mark Steedman for helpful discussions, and Jason Eisner for his very generous feedback which helped to greatly improve this paper. All remaining errors and omissions are our own responsibility. J.H is supported by NSF grant IIS 08- 03603 INT2-Medium.

An Empirical Study on Web Mining of Parallel Data

Gumwon Hong[1], **Chi-Ho Li**[2], **Ming Zhou**[2] **and Hae-Chang Rim**[1]

[1]Department of Computer Science & Engineering, Korea University
{gwhong,rim}@nlp.korea.ac.kr

[2]Natural Language Computing Group, Microsoft Research Asia
{chl,mingzhou}@microsoft.com

Abstract

This paper[1] presents an empirical approach to mining parallel corpora. Conventional approaches use a readily available collection of comparable, non-parallel corpora to extract parallel sentences. This paper attempts the much more challenging task of directly searching for high-quality sentence pairs from the Web. We tackle the problem by formulating good search query using 'Learning to Rank' and by filtering noisy document pairs using IBM Model 1 alignment. End-to-end evaluation shows that the proposed approach significantly improves the performance of statistical machine translation.

1 Introduction

Bilingual corpora are very valuable resources in NLP. They can be used in statistical machine translation (SMT), cross language information retrieval, and paraphrasing. Thus the acquisition of bilingual corpora has received much attention.

Hansards, or parliamentary proceedings in more than one language, are obvious source of bilingual corpora, yet they are about a particular domain and therefore of limited use. Many researchers then explore the Web. Some approach attempts to locate bilingual text within a web page (Jiang et al., 2009); some others attempt to collect web pages in different languages and decide the parallel relationship between the web pages by means of *structural cues,* like existence of a common ancestor web page, similarity between URLs, and similarity between the HTML structures (Chen and Nie, 2000; Resnik and Smith, 2003; Yang and Li, 2003; Shi et al., 2006). The corpora thus obtained are generally of high quality and wide variety in domain, but the amount is still limited, as web pages that exhibit those structural cues are not abundant.

Some other effort is to mine bilingual corpora by *textual means* only. That is, two pieces of text are decided to be parallel merely from the linguistic perspective, without considering any hint from HTML markup or website structure. These approaches (Zhao and Vogel, 2002; Utiyama and Isahara 2003; Fung and Cheung, 2004; Munteanu and Marcu, 2005; Abdul-Rauf and Schwenk, 2009) share roughly the same framework:

Phase 1: Document Pair Retrieval

1) documents in some target language (TL) are stored in some database;
2) each document in some source language (SL) is represented by some TL keywords;
3) the TL keywords in (2) are used to assign some TL documents to a particular SL document, using some information retrieval (IR) technique. For example, Munteanu and Marcu (2005) apply the Lemur IR toolkit, Utiyama and Isahara (2003) use the BM25 similarity measure, and Fung and Cheung (2004) use cosine similarity. Each TL document pairs up with the SL document to form a candidate parallel document pair.

Phase 2: Sentence Pair Extraction

1) sentence pairs can be obtained by running sentence alignment over all candidate document pairs (or a selection of them) (Zhao and Vogel, 2002; Utiyama and Isahara, 2003);
2) sentence pairs can also be selected, by some classifier or reliability measure, from the candidate sentence pairs enumerated from the candidate document pairs (Munteanu and Marcu, 2005).

Note that the primary interest of these approaches is sentence pairs rather than document

[1] This work has been done while the first author was visiting Microsoft Research Asia.

pairs, partially because document pair retrieval is not accurate, and partially because the ultimate purpose of these corpora is SMT training, which is based on sentence pairs. It is found that most of the sentence pairs thus obtained are not truly parallel; rather they are loose translations of each other or they carry partially similar messages. Such bilingual corpora are thus known as *comparable corpora*, while genuinely mutual translations constitute *parallel corpora*.

Note also that all these comparable corpus mining approaches are tested on *closed document collections* only. For example, Zhao and Vogel (2002), Utiyama and Isahara (2003), and Munteanu and Marcu (2005) all acquire their comparable corpora from a collection of news articles which are either downloaded from the Web or archived by LDC. The search of candidate document pairs in such a closed collection is easy in three ways:

1) all the TL documents come from the same news agency and they are not mixed up with similar documents from other news agencies;
2) all the TL documents are news text and they are not mixed up with text of other domains;
3) in fact, the search in these approaches is made easier by applying tricks like date window.

There is no evidence that these methods apply to corpus mining from an open document collection (e.g. the entire Web) without search constraint. The possibility of open-ended text mining is a crucial problem.

This paper focuses on bilingual corpus mining using only textual means. It attempts to answer two questions:

1) Can comparable corpus mining be applied to an open document collection, i.e., the Web?
2) Can comparable corpus mining be adapted to parallel corpus mining?

We give affirmation to both questions. For the first problem, we modify document pair retrieval so that there is no longer a closed set of TL documents. Instead we search for candidate TL documents for a particular SL document from the Web by means of some Web search engine. For the second problem, in Phase 2 we replace the sentence pair classifier by a document pair filter and a sentence alignment module. Based on end-to-end SMT experiments, we will show that 1) high quality bilingual corpora can be mined from the Web; 2) the very first key to Web-mining of bilingual corpus is the formulation of good TL keywords to represent a SL document; 3) a simple document pair filter using IBM Model 1 probabilities is able to identify parallel corpus out of noisy comparable text; and 4) Web-mined parallel corpus, despite its smaller size, improves SMT much more than Web-mined comparable corpus.

2 Problem Setting

Our ultimate goal is to mine from the Web training data for translation from Chinese (SL) to English (TL). As the first step, about 11,000 Chinese web pages of news articles are crawled from some Chinese News sites. Then the task is to search for the English sentences corresponding to those in the selected SL articles. These selected SL news articles all contain cue phrases like “根据外电报道” (*according to foreign media*), as these cue phrases suggest that the Chinese articles are likely to have English counterparts. Moreover, each selected SL article has at least 500 words (empirically determined) since we assume that it is much easier to formulate reliable keywords from a long document than a short one.

3 Document Pair Retrieval

Conventional approaches to comparable corpus mining usually start with document pair retrieval, which assigns to each SL document a set of candidate TL documents. This step is essentially a preliminary search for candidate sentence pairs for further scrutiny in Phase 2. The target is to find document pairs which may contain many good sentence pairs, rather than to discard document pairs which may not contain good sentence pairs. Therefore, *recall is much more emphasized* than precision.

Document pair retrieval in conventional approaches presumes a closed set of TL documents which some IR system can handle easily. In this paper we override this presumption and attempt a much more challenging retrieval task, viz. to search for TL documents among the Web, using the search engines of Google and Yahoo. Therefore we are subject to a much noisier data domain. The correct TL documents may not be indexed by the search engines at all, and even when the target documents are indexed, it requires a more sophisticated formulation of queries to retrieve them.

In response to these challenges, we propose various kinds of queries (elaborated in the following subsections). Moreover, we merge the TL documents found by each query into a big collection, so as to boost up the recall. In case a query fails to retrieve any document, we iteratively drop a keyword in the query until some documents are found. On the other hand, although the document pairs in question are of news domain, we use the general Google/Yahoo web search engines instead of the specific news search engines, because 1) the news search engines keep only a few web pages for all pages about the same news event, and 2) we leave open possibility for correct TL documents to be found in non-news web pages.

3.1 Simple Queries

There are three baseline formulations of queries:

1) Query of translations of SL TF-IDF-ranked keywords ($Q_{SL\text{-}TFIDF}$). This is the method proposed by Munteanu and Marcu (2005). All the words in a SL document are ranked by TF-IDF and the top-N words are selected. Each keyword is then translated into a few TL words by a statistically learned dictionary. In our experiments the dictionary is learned from NIST SMT training data.
2) Query of TF-IDF-ranked machine translated keywords ($Q_{TL\text{-}TFIDF}$). It is assumed that a machine translation (MT) system is better at handling lexical ambiguity than simple dictionary translation. Thus we propose to first translate the SL document into TL and extract the top-N TF-IDF-ranked words as query. In our experiments the MT system used is hierarchical phrase-based system (Chiang, 2007).[2]
3) Query of named entities (Q_{NE}). Another way to tackle the drawback of $Q_{SL\text{-}TFIDF}$ is to focus on named entities (NEs) only, since NEs often provide strong clue for identifying correspondence between two languages. All NEs in a SL document are ranked by TF-IDF, and the top-N NEs are then translated (word by word) by dictionary. In our experiments we identify SL (Chinese) NEs implicitly found by the word segmentation algorithm stated in Gao et al. (2003), and the dictionaries for translating NEs include the same one used for $Q_{SL\text{-}TFIDF}$, and the LDC Chinese/English NE dictionary. For the NEs not covered by our dictionary, we use Google translation service as a back-up.

A small-scale experiment is run to evaluate the merits of these queries. 300 Chinese news web pages in three different periods (each 100) are collected. For each Chinese text, each query (containing 10 keywords) is constructed and submitted to both Google and Yahoo Search, and top-40 returned English web pages for each search are kept. Note that the Chinese news articles are not part of 11,000 pages in section 2. In fact, they do not only satisfy the requirement of length and cue phrases (described in section 2), but they also have another property that they are translated from some English news articles (henceforth target pages) on the Web. Thus they are ideal data for studying the performance of document pair retrieval.

To test the influence of translation quality in document pair retrieval, we also try 'oracle queries', i.e. queries formulated directly from the target pages:

1) OQ_{TFIDF}. This is the query of the top-N TF-IDF-ranked words from the target page.
2) OQ_{NE}. This is the query of the top-N TF-IDF-ranked NEs from the target web page.

We define recall as the proportion of SL documents whose true target pages are found. The comparison between a retrieved page and the target page is done by Longest Common Subsequence (LCS) ratio, defined as the length of the longest common word sequence of two documents divided by the length of the longer of two documents. The threshold 0.7 is adopted as it is strict enough to distinguish parallel document pairs from non-parallel ones.

Table 1 shows the recalls for various queries. It can be seen from Tests 6 and 7 that the largest recall, 85% (within top 40 search results), is achieved when the word distributions in the target web pages are known. In the real scenario where the true English word distribution is not known, the recalls achieved by the simple queries are very unsatisfactory, as shown by Tests 1 to 3. This clearly shows how challenging Web-based mining of bilingual corpora is. Another challenge can be observed in comparing across

[2] We also try online Google translation service, and the performance was roughly the same.

ID	Query	Remote	Near	Recent
1	$Q_{SL\text{-}TFIDF}$	7	6	8
2	$Q_{TL\text{-}TFIDF}$	16	19	32
3	Q_{NE}	16	21	38
4	union(2,3)	27	31	48
5	union(1,2,3)	28	31	48
6	OQ_{TFIDF}	56	66	82
7	OQ_{NE}	62	68	85
8	$Overlap_{TFIDF}$	52	51	74
9	$Overlap_{NE}$	55	62	83

Table 1: Recall (%age) of simple queries. 'Remote' refers to news documents more than a year ago; 'Near' refers to documents about 3 months ago; 'Recent' refers to documents in the last two weeks.

columns, viz. it is much more difficult to retrieve outdated news document pairs. This implies that bilingual news mining must be incrementally carried out.

Comparing Test 1 to Tests 2 and 3, it is obvious that $Q_{SL\text{-}TFIDF}$ is not very useful in document pair retrieval. This confirms our hypothesis that suitable TL keywords are not likely to be obtained by simple dictionary lookup. While the recalls by $Q_{TL\text{-}TFIDF}$ are similar to those by Q_{NE}, the two queries contribute in different ways. Test 4 simply merges the Web search results in Tests 2 and 3. The significantly higher recalls in Test 4 imply that each of the two queries finds substantially different targets than each other. The comparison of Test 5 to Test 4 further confirms the weakness of $Q_{SL\text{-}TFIDF}$.

The huge gap between the three simple queries and the oracle queries shows that the quality of translation of keywords from SL to TL is a major obstacle. There are two problems in translation quality: 1) the MT system or dictionary *cannot produce any translation* for a SL word (let us refer to such TL keywords as 'Utopian translations'); 2) the MT system or dictionary *produces an incorrect translation* for a SL word. We can do very little for the Utopian translations, as the only solution is simply to use a better MT system or a larger dictionary. On the contrary, it seems that the second problem can somewhat be alleviated, if we have a way to distinguish those terms that are likely to be correct translations from those terms that are not. In other words, it may be worthwhile to reorder candidate TL keywords by our confidence in its translation quality.

Tests 8 and 9 in Table 1 show that this hypothesis is promising. In both tests the TF-IDF-based (Test 8) or the NE-based (Test 9) keywords are selected from only those TL words that appear both in the target page and the machine translated text of the source page. In other words, we ensure that the keywords in the query must be correct translations. The recalls (especially the recalls by NE-based query in Test 9) are very close to the recalls by oracle queries. The conclusion is, even though we cannot produce the Utopian translations, document pair retrieval can be improved to a large extent by removing incorrect translations. Even an imperfect MT system or NE dictionary can help us achieve as good document pair retrieval recall as oracle queries.

In the next subsection we will take this insight into our bilingual data mining system, by selecting keywords which are likely to be correct translation.

3.2 Re-ranked Queries

Machine learning is applied to re-rank keywords for a particular document. The re-ranking of keywords is based on two principles. The first one is, of course, the confidence on the translation quality. The more likely a keyword is a correct translation, the higher this keyword should be ranked. The second principle is the representativeness of document. The more representative of the topic of the document where a keyword comes from, the higher this keyword should be ranked. The design of features should incorporate both principles.

The representativeness of document is manifested in the following features for each keyword per each document:

- *TF*: the term frequency.
- *IDF*: the inverted document frequency.
- *TF-IDF*: the product of *TF* and IDF.
- *Title word*: it indicates whether a keyword appears in the title of the document.
- *Bracketed word*: it indicates whether a word is enclosed in a bracket in the source document.
- *Position of first appearance*: the position where a keyword first appears in a document, normalized by number of words in the document.

- *NE types*: it indicates whether a keyword is a person, organization, location, numerical expression, or non NE.

The confidence on translation quality is manifested in the following features:

- *Translation source*: it indicates whether the keyword (in TL) is produced by MT system, dictionary, or by both.
- *Original word*: it indicates whether the keyword is originally written in *English* in the source document. Note that this feature also manifests the representativeness of a document.
- *Dictionary rank*: if the keyword is a NE produced by dictionary, this feature indicates the rank of the NE keyword among all translation options registered in the dictionary.

It is difficult to definitely classify a TL keyword into good or bad translation in absolute sense, and therefore we take the alternative of ranking TL keywords with respect to the two principles. The learning algorithm used is Ranking SVM (Herbrich et al., 2000; Joachims, 2006), which is a state-of-the-art method of the "Learning to rank" framework.

The training dataset of the keyword re-ranker comprises 1,900 Chinese/English news document pairs crawled from the Web[3]. This set is not part of 11,000 pages in section 2. These document pairs share the same properties as those 300 pairs used in Section 3.1. For each English/target document, we build a set T_{ALL}, which contains all words in the English document, and also a set T_{NE}, which is a subset of T_{ALL} such that all words in T_{NE} are NEs in T_{ALL}. The words in both sets are ranked by TFIDF. On the other hand, for each Chinese/source document, we machine-translate it and then store the translated words into a set S, and we also add the dictionary translations of the source NEs into S. Note that S is composed of both good translations (appearing in the target document) and bad translations (not appearing in the target document).

Then there are two ways to assign labels to the words in S. In the first way of labeling (L_{ALL}), the label ***3*** is assigned to those words in S which are ranked among top 5 in T_{ALL}, label ***2*** to those ranked among top 10 but not top 5 in T_{ALL}, ***1*** to those beyond top 10 but still in T_{ALL}, and ***0*** to those words which do not appear in T_{ALL} at all. The second way of labeling, L_{NE}, is done in similar way with respect to T_{NE}. Collecting all training samples over all document pairs, we can train a model, M_{ALL}, based on labeling L_{ALL}, and another model M_{NE}, based on labeling L_{NE}.

The trained models can then be applied to re-rank the keywords of simple queries. In this case, a set S_{TEST} is constructed from the 300 Chinese documents in similar way of constructing S. We repeat the experiment in Section 3.1 with two new queries:

1) $Q_{RANK\text{-}TFIDF}$: the top N keywords from re-ranking S_{TEST} by M_{ALL};
2) $Q_{RANK\text{-}NE}$: the top N keywords from reranking S_{TEST} by M_{NE}.

Again N is chosen as 10.

ID	Query	Remote	Near	Recent
10	$Q_{RANK\text{-}TFIDF}$	18	20	29
11	$Q_{RANK\text{-}NE}$	35	43	54
12	union(10,11)	39	49	63

Table 2: Recall (%age) of re-ranked queries.

The results shown in Table 2 indicate that, while the re-ranked queries still perform much poorer than oracle queries (Tests 6 and 7 in Table 1), they show great improvement over the simple queries (Tests 1 to 5 in Table 1). The results also show that re-ranked queries based on NEs are more reliable than those based on common words.

4 Sentence pair Extraction

The document pairs obtained by the various queries described in Section 3 are used to produce sentence pairs as SMT training data. There are two different methods of extraction for corpora of different nature.

4.1 For Comparable Corpora

Sentence pair extraction for comparable corpus is the same as that elaborated in Munteanu and Marcu (2005). All possible sentence pairs are enumerated from all candidate document pairs produced in Phase 1. These huge number of candidate sentence pairs are first passed to a coarse sentence pair filter, which discards very unlikely candidates by heuristics like sentence

[3] We also attempt to add more training data for re-ranking but the performance remain the same.

length ratio and percentage of word pairs registered in some dictionary.

The remaining candidates are then given to a Maximum Entropy based classifier (Zhang, 2004), which uses features based on alignment patterns produced by some word alignment model. In our experiment we use the HMM alignment model with the NIST SMT training dataset. The sentence pairs which are assigned as positive by the classifier are collected as the mined comparable corpus.

4.2 For Parallel Corpora

The sentence pairs obtained in Section 4.1 are found to be mostly not genuine mutual translations. Often one of the sentences contains some extra phrase or clause, or even conveys different meaning than the other. It is doubtful if the document pairs from Phase 1 are too noisy to be processed by the sentence pair classifier. An alternative way for sentence pair extraction is to further filter the document pairs and discard any pairs that do not look like parallel.

It is hypothesized that the parallel relationship between two documents can be assimilated by the word alignment between them. The document pair filter produces the Viterbi alignment, with the associated probability, of each document pair based on IBM Model 1 (Brown et al., 1993). The word alignment model (i.e. the statistical dictionary used by IBM Model 1) is trained on the NIST SMT training dataset. The probability of the Viterbi alignment of a document pair is the sole basis on which we decide whether the pair is genuinely parallel. That is, an empirically determined threshold is used to distinguish parallel pairs from non-parallel ones. In our experiment, a very strict threshold is selected so as to boost up the precision at the expense of recall.

There are a few important details that enable the document pair filter succeed in identifying parallel text:

1) Function words and other common words occur frequently and so any pair of common word occupies certain probability mass in an alignment model. These common words enable even non-parallel documents achieve high alignment probability. In fact, it is well known that the correct alignment of common words must take into account positional and/or structural factors, and it is beneficial to a simple alignment model like IBM Model 1 to work on data without common words. Therefore, all words on a comprehensive stopword list must be removed from a document pair before word alignment.
2) The alignment probability must be normalized with respect to sentence length, so that the threshold applies to all documents regardless of document length.

Subjective evaluation on selected samples shows that most of the document pairs kept by the filter are genuinely parallel. Thus the document pairs can be broken down into sentence pairs simply by a sentence alignment method. For the sentence alignment, our experiments use the algorithm in Moore (2002).

5 Experiments

It is a difficult task to evaluate the quality of automatically acquired bilingual corpora. As our ultimate purpose of mining bilingual corpora is to provide more and better training data for SMT, we evaluate the parallel and comparable corpora with respect to improvement in Bleu score (Papineni et al., 2002).

5.1 Experiment Setup

Our experiment starts with the 11,000 Chinese documents as described in Section 2. We use various combinations of queries in document pair retrieval (Section 3). Based on the candidate document pairs, we produce both comparable corpora and parallel corpora using sentence pair extraction (Section 4). The corpora are then given to our SMT systems as training data.

The SMT systems are our implementations of phrase-based SMT (Koehn et al., 2003) and hierarchical phrase-based SMT (Chiang, 2007). The two systems employ a 5-gram language model trained from the Xinhua section of the Gigaword corpus. There are many variations of the bilingual training dataset. The B1 section of the NIST SMT training set is selected as the baseline bilingual dataset; its size is of the same order of magnitude as most of the mined corpora so that the comparison is fair. Each of the mined bilingual corpora is compared to that baseline dataset, and we also evaluate the performance of the combination of each mined bilingual corpus with the baseline set.

Bilingual Training Corpus	Phrase-based SMT (PSMT)		Hierarchical PSMT	
	NIST 2005	NIST 2008	NIST 2005	NIST 2008
B1 (baseline)	33.08	21.66	32.85	21.18
B1+comparable(M&M)	33.51(+0.43)	22.71(+1.05)	32.99(+0.14)	22.11(+0.93)
B1+comparable($Q_{\text{RANK-NE}}$)	**34.81(+1.73)**	23.30(+1.64)	34.43(+1.58)	22.85(+1.67)
B1+comparable(all simple)	34.74(+1.66)	**23.48(+1.82)**	34.28(+1.43)	**23.18(+2.00)**
B1+comparable(all ranked)	34.79(+1.71)	**23.48(+1.82)**	34.37(+1.52)	23.06(+1.88)
B1+comparable(all query)	34.74(+1.66)	23.19(+1.53)	**34.46(+1.61)**	23.12(+1.94)
B1+parallel($Q_{\text{RANK-NE}}$)	34.75(+1.67)	23.37(+1.71)	34.24(+1.39)	23.45(+2.27)
B1+parallel(all simple)	34.99(+1.91)	23.96(+2.30)	34.94(+2.09)	23.35(+2.17)
B1+parallel(all ranked)	34.76(+1.68)	23.41(+1.75)	34.54(+1.69)	23.59(+2.41)
B1+parallel(all query)	**35.40(+2.32)**	**23.47(+1.81)**	**35.27(+2.42)**	**23.61(+2.43)**

Table 4: Evaluation of translation quality improvement by mined corpora. The figures inside brackets refer to the improvement over baseline. The bold figures indicate the highest Bleu score in each column for comparable corpora and parallel corpora, respectively.

The SMT systems learn translation knowledge (phrase table and rule table) in standard way. The parameters in the underlying log-linear model are trained by Minimum Error Rate Training (Och, 2003) on the development set of NIST 2003 test set. The quality of translation output is evaluated by case-insensitive BLEU4 on NIST 2005 and NIST 2008 test sets[4].

5.2 Experimental result

Table 3 lists the size of various mined parallel and comparable corpora against the baseline B1 bilingual dataset. It is obvious that for a specific type of query in document pair retrieval, the parallel corpus is significantly smaller than the corresponding comparable corpus.

The apparent explanation is that a lot of document pairs are discarded due to the document pair filter. Note that the big difference in size of the two comparable corpora by single queries, i.e., $Q_{\text{RANK-NE}}$ and M&M, verifies again that re-ranked queries based on NEs are more reliable in sentence pair extraction.

Queries	SP extraction	#SP	#SL words	#TL words
Baseline: B1 in NIST		68K	1.7M	1.9M
M&M	comparable	43K	1.1M	1.2M
$Q_{\text{RANK-NE}}$	comparable	98K	2.7M	2.8M
all simple	comparable	98K	2.6M	2.9M
all ranked	comparable	115K	3.1M	3.3M
all query	comparable	135K	3.6M	4.0M
$Q_{\text{RANK-NE}}$	parallel	66K	1.9M	1.8M
all simple	parallel	52K	1.5M	1.4M
all ranked	parallel	73K	2.1M	2.0M
all query	parallel	90K	2.5M	2.4M

Table 3: Statistics on corpus size. SP means sentence pair. 'all simple', 'all ranked', and 'all query' refer to the merge of the retrieval results of all simple queries, all re-ranked queries, and all simple and re-ranked queries, respectively; M&M (after Munteanu and Marcu (2005)) refers to $Q_{\text{SL-TFIDF}}$.

Table 4 lists the Bleu scores obtained by *augmenting* the baseline bilingual training set *with* the mined corpora. The most important observation is that, despite their smaller size, parallel corpora lead to no less, and often better, improvement in translation quality than comparable corpora. That is especially true for the case where document pair retrieval is based on all five types of query[5]. The superiority of parallel corpora confirms that, in Phase 2 (sentence pair extraction), quality is more important than quantity and thus the filtering of document pair/sentence pair must not be generous.

On the other hand, sentence pair extraction for parallel corpora generally achieves the best result when all queries are applied in document pair retrieval. It is not sufficient to use the more sophisticated re-ranked queries. That means in Phase 1 quantity is more important and we must seek more ways to retrieve as many document pairs as possible. That also confirms the emphasis on recall in document pair retrieval.

Looking into the performance of comparable corpora, it is observed that the M&M query does not effectively apply to Web mining of comparable corpora but the proposed queries do. Any of the proposed query leads to better result than the conventional method, i.e. M&M. Moreover, it can be seen that all four combinations of proposed queries achieve similar per-

[4] It is checked that there is no sentence in the test sets overlapping with any sentences in the mined corpus.

[5] $Q_{\text{SL-TFIDF}}$, $Q_{\text{TL-TFIDF}}$, Q_{NE}, $Q_{\text{RANK-TFIDF}}$, and $Q_{\text{RANK-NE}}$

Bilingual Training Corpus	**Phrase-based SMT**		**Hierarchical PSMT**	
	NIST 2005	NIST 2008	NIST 2005	NIST 2008
B1 (baseline)	33.08	21.66	32.85	21.18
comparable(M&M)	20.84(-12.24)	14.33(-7.33)	20.65(-12.20)	13.73(-7.45)
comparable($Q_{RANK\text{-}NE}$)	26.78(-6.30)	18.54(-3.12)	27.10(-5.75)	18.02(-3.16)
comparable(all simple)	26.39(-6.69)	18.52(-3.14)	26.40(-6.45)	18.22(-2.96)
comparable(all ranked)	27.36(-5.72)	18.89(-2.77)	27.40(-5.45)	18.72(-2.46)
comparable(all query)	**27.96(-5.12)**	**19.27(-2.39)**	**27.83(-5.02)**	**19.46(-1.72)**
parallel($Q_{RANK\text{-}NE}$)	26.37(-6.71)	18.70(-2.96)	26.47(-6.38)	18.51(-2.67)
parallel(all simple)	25.65(-7.43)	18.69(-2.97)	25.28(-7.57)	18.55(-2.63)
parallel(all ranked)	26.86(-6.22)	18.94(-2.72)	27.10(-5.75)	18.78(-2.40)
parallel(all query)	**27.58(-5.50)**	**19.73(-1.93)**	**28.10(-4.75)**	**19.52(-1.66)**

Table 5: Evaluation of translation quality by mined corpora.

formance. This illustrates a particular advantage of using a single re-ranked query, viz. $Q_{RANK\text{-}NE}$, because it significantly reduces the retrieval time and downloading space required for document pair retrieval as it is the main bottleneck of whole process.

Table 5 lists the Bleu scores obtained by *replacing* the baseline bilingual training set *with* the mined corpora. It is easy to note that translation quality drops radically by using mined bilingual corpus alone. That is a natural consequence of the noisy nature of Web mined data. We should not be too pessimistic about Web mined data, however. Comparing the Bleu scores for NIST 2005 test set to those for NIST 2008 test set, it can be seen that the reduction of translation quality for the NIST 2008 set is much smaller than that for the NIST 2005 set. It is not difficult to explain the difference. Both the baseline B1 training set and the NIST 2005 comprise news wire (in-domain) text only. Although the acquisition of bilingual data also targets news text, the noisy mined corpus can never compete with the well prepared B1 dataset. On the contrary, the NIST 2008 test set contains a large portion of out-of-domain text, and so the B1 set does not gain any advantage over Web mined corpora. It might be that better and/or larger Web mined corpus achieves the same performance as manually prepared corpus.

Note also that the reduction in Bleu score by each mined corpus is roughly the same as that by each other, while in general parallel corpora are slightly better than comparable corpora.

6 Conclusion and Future Work

In this paper, we tackle the problem of mining parallel sentences directly from the Web as training data for SMT. The proposed method essentially follows the corpus mining framework by pioneer work like Munteanu and Marcu (2005). However, unlike those conventional approaches, which work on closed document collection only, we propose different ways of formulating queries for discovering parallel documents over Web search engines. Using learning to rank algorithm, we re-rank keywords based on representativeness and translation quality. This new type of query significantly outperforms existing query formulation in retrieving document pairs. We also devise a document pair filter based on IBM model 1 for handling the noisy result from document pair retrieval. Experimental results show that the proposed approach achieves substantial improvement in SMT performance.

For mining news text, in future we plan to apply the proposed approach to other language pairs. Also, we will attempt to use meta-information implied in SL document, such as “publishing date” or “news agency name”, as further clue to the document pair retrieval. Such meta-information may likely to increase the precision of retrieval, which is important to the efficiency of the retrieval process.

An important contribution of this work is to show the possibility of mining text other than news domain from the Web, which is another piece of future work. The difficulty of this task should not be undermined, however. Our success in mining news text from the Web depends on the cue phrases available in news articles. These cue phrases more or less indicate the existence of corresponding articles in another language. Therefore, to mine non-news corpus, we should carefully identify and select cue phrases.

References

Abdul-Rauf, Sadaf and Holger Schwenk. 2009. Exploiting Comparable Corpora with TER and TERp. In *Proceedings of ACL-IJCNLP 2009 workshop on Building and Using Comparable Corpora*, pages 46–54.

Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2): 263-311.

Chen, Jiang and Jian-Yun Nie. 2000. Automatic Construction of Parallel Chinese-English Corpus for Cross-Language Information Retrieval. In *Proceedings of NAACL-ANLP*, pages 21-28.

Chiang, David. 2007. Hierarchical Phrase-based Translation. *Computational Linguistics*, 33(2): 202-228.

Fung, Pascale, and Percy Cheung. 2004. Mining very non-parallel corpora: Parallel sentence and lexicon extraction via bootstrapping and EM. In *Proceedings of 2004 Conference on Empirical Methods in Natural Language Processing*, pages 57-63.

Gao, Jianfeng, Mu Li, and Changning Huang. 2003. Improved Source-Channel Models for Chinese Word Segmentation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 272-279.

Herbrich, Ralf, Thore Graepel, and Klaus Obermayer. 2000. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, Cambridge, MA.

Jiang, Long, Shiquan Yang, Ming Zhou, Xiaohua Liu, and Qingsheng Zhu. 2009. Mining Bilingual Data from the Web with Adaptively Learnt Patterns. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and 4th International Joint Conference on Natural Language Processing*, pages 870-878.

Joachims, Thorsten. 2006. Training Linear SVMs in Linear Time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 217-226.

Koehn, Philipp, Franz Och, and Daniel Marcu. 2003. Statistical Phrase-based Translation. In *Proceedings of conference combining Human Language Technology conference series and the North American Chapter of the Association for Computational Linguistics conference series*, pages 48-54.

Moore, Robert. 2002. Fast and Accurate Sentence Alignment of Bilingual Corpora. In *Proceedings of the 5th conference of the Association for Machine Translation in the Americas*, pages 135–144.

Munteanu, Dragos, and Daniel Marcu. 2005. Improving Machine Translation Performance by Exploiting Non-Parallel Corpora. *Computational Linguistics*, 31(4): 477-504.

Och, Franz J. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160-167.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311-318.

Resnik, Philip, and Noah Smith. 2003. The Web as a Parallel Corpus. *Computational Linguistics*, 29(3): 349-380.

Shi, Lei, Cheng Niu, Ming Zhou, and Jianfeng Gao. 2006. A DOM Tree Alignment Model for Mining Parallel Data from the Web. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 489-496.

Utiyama, Masao, and Hitoshi Isahara. 2003. Reliable Measures for Aligning Japanese-English News Articles and Sentences. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 72-79.

Vogel, Stephan. 2003. Using noisy bilingual data for statistical machine translation. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics*, pages 175-178.

Yang, Christopher C., and Kar Wing Li. 2003. Automatic construction of English/Chinese parallel corpora. *Journal of the American Society for Information Science and Technology*, 54(8):730–742.

Zhang, Le. 2004. Maximum Entropy Modeling Toolkit for Python and C++. http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

Zhao, Bing, and Stephan Vogel. 2002. Adaptive Parallel Sentences Mining from Web Bilingual News Collection. In *Proceedings of IEEE international conference on data mining*, pages 745-750.

Enhancing Cross Document Coreference of Web Documents with Context Similarity and Very Large Scale Text Categorization

Jian Huang
Information Sciences and Technology
Pennsylvania State University
jhuang@ist.psu.edu

Pucktada Treeratpituk
Information Sciences and Technology
Pennsylvania State University
pxt162@ist.psu.edu

Sarah M. Taylor
Lockheed Martin IS&GS
sarah.m.taylor@lmco.com

C. Lee Giles
Information Sciences and Technology
Pennsylvania State University
giles@ist.psu.edu

Abstract

Cross Document Coreference (CDC) is the task of constructing the coreference chain for mentions of a person across a set of documents. This work offers a holistic view of using document-level categories, sub-document level context and extracted entities and relations for the CDC task. We train a categorization component with an efficient flat algorithm using thousands of ODP categories and over a million web documents. We propose to use ranked categories as coreference information, particularly suitable for web documents that are widely different in style and content. An ensemble composite coreference function, amenable to inactive features, combines these three levels of evidence for disambiguation.

A thorough feature importance study is conducted to analyze how these three components contribute to the coreference results. The overall solution is evaluated using the WePS benchmark data and demonstrate superior performance.

1 Introduction

Cross Document Coreference (CDC) is the task to determine whether Named Entities (NE) from different documents refer to the same underlying identity. CDC enables a range of advanced NLP applications such as automated text summarization and question answering (e.g. list-type questions). CDC has mainly been developed from two perspectives.

First, in the Message Understanding Conference (MUC-6), CDC was viewed as an advanced task performed based on a set of Information Extraction (IE) artifacts. IE has been one of the central topics in NLP since the 1970s and gained much success in transforming natural language text to structured text. IE on the Web, however, is inherently very challenging. For one, the Web is comprised of such heterogenous content that IE systems, many of which are developed on tidy and domain-specific corpora, may achieve relatively limited coverage. Also, the content of web documents may not even be in the natural language form. Hence, though IE based features are quite precise, it is rather difficult to achieve good coverage that's necessary to disambiguate person entities on the Web.

Recently, there is significant research interest in a related task called Web Person Search (WePS) (Artiles et al., 2007), which seeks to determine whether two documents refer to the same person given a person name search query. Many systems employed the simple vector space model and word co-occurrence features for this task. Though more robust with better coverage, these methods are more susceptible to irrelevant words with regard to the entity of interest.

Rather than relying solely on IE based or word co-occurrence features, this work adopts a holistic view of the different types of features useful for cross document coreference. Specifically, the main features of our proposed CDC approach are:

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 483–491,
Beijing, August 2010

- The proposed approach covers the entire spectrum of document level, sub-document context level and entity/relation level disambiguation evidence. In particular, we propose to use document categories as robust document level evidence. This comprehensive design naturally combines state-of-the-art categorization, information extraction and IE-driven IR methods and compensates the limitation of each of them.
- The features used in this work are domain independent and thus are particularly suitable for coreferencing web documents.
- The composite pairwise coreference function in this work can readily incorporate a set of heterogenous features that are not always active or are in different ranges, making it easily extensible to additional features. Moreover, we thoroughly study the contribution of each component and its features to gain insight on improving cross document coreference performance.

In this work, three components specialize in generating the aforementioned three levels of features as coreference decisions. Thus we refer to them as *experts*. After reviewing prior work on CDC, we describe the methods of each of these components in detail and present empirical results where appropriate. We then show how these components (and its features) are aggregated to predict pairwise coreference using an ensemble method. We evaluate the contribution of each component and the overall CDC results on a benchmark dataset. Finally, we conclude and discuss future work.

2 Related Work

Compared to the traditional (within-document) coreference resolution problem, cross document coreference is a much harder problem due to the divergence of contents and the lack of consistent discourse information across documents.

(Bagga and Baldwin, 1998b) presented one of the first CDC systems, which relied solely on the contextual words of the named entities. (Gooi and Allan, 2004) used a 55-word window as the context without significant accuracy penalty. As these approaches only considered word co-occurrence, they were more susceptible to genre differences. Recent CDC work has sought Information Extraction (IE) support. Extracted NEs and relationships were considered in (Niu et al., 2004) for improved CDC performance.

Many of these earlier CDC methods were evaluated on small and tidy news articles. CDC for Web documents is even more challenging. (Wan et al., 2005) proposed a web person resolution system called WebHawk, which extracted several attributes such as title, organization, email and phone number using patterns. These features however only covered small amount of disambiguation evidence and certain types of web pages (such as personal home pages). The more recent Web Person Search (WePS) task (Artiles et al., 2007) has created a benchmark dataset which is also used in this work. Different from CDC which aims to resolve mention level NEs, WePS distinguishes *documents* retrieved by a name search query according to the underlying identity. The top-performing system (Chen and Martin, 2007) in this task extracted phrasal contextual and document-level entities as rich features for coreference. Similar IR features are also used by other WePS systems as they are more robust to the variety of web pages (Artiles et al., 2007).

Instead of focusing on local information, (Li et al., 2004) proposed a generative model of entity co-occurrence to capture global document level information. However, inference in generative models is expensive for large scale web data. Our work instead considers document categories/topics that can be efficiently predicted and easily interpretable by users. Hand-tuned weights were used in (Baron and Freedman, 2008) and a linear classifier was used in (Li et al., 2004) to combine the extracted features. Our composite pairwise coreference function is based on an ensemble classifier and is more robust and capable of handling inactive features.

3 Text Categorization Aided CDC

Consider the following scenario for motivation. When a user searches for 'Michael Jordan', the official web page of the basketball player

'Michael Jordan'[1] contains mostly his career statistics, whereas the homepage of 'Michael I. Jordan' the professor[2] contains his titles, contact information and advising students. Neither of these pages contain complete natural language sentences that most IE and NLP tools are designed to process. We propose to use document categories (trained from a very large scale and general purpose taxonomy, Open Directory Project (ODP)) as document level features for CDC. In this example, one can easily differentiate these namesakes by categorizing the former as 'Top/Sports/Basketball/Professional' and the latter as 'Top/Computer/Artificial Intelligence/Machine Learning'. We first introduce the method to categorize Web documents; then we show how to combine these categories for coreferencing.

3.1 Very Large Scale Text Categorization

To handle the web CDC problem, the catagorization component needs to be able to categorize documents of widely different topics. The Open Directory Project (ODP), the largest and most comprehensive human edited directory of the Web[3], contains hundreds of thousands of categories labeled for 2 million Web pages. Leveraging this vast amount of web data and the large Web taxonomy has called for the development of very efficient text categorization methods. There is significant research interest in scaling up to categorize millions of pages to thousands of categories and beyond, called the many class classification setting (Madani and Huang, 2008). Flat classification methods (e.g. (Crammer et al., 2006; Madani and Huang, 2008)), which treat hierarchical categories as flat classes, have been very successful due to their superior scalability and simplicity compared to classical hierarchical one-against-rest categorization. Flat methods also achieve high accuracy that is on par with, or better than, the traditional counterparts.

We adopt a flat multiclass online classification algorithm Passive Aggressive (PA) (Crammer et al., 2006) to predict ranked categories for web documents. For a categorization problem with C categories, PA associates each category k with a weight vector $\mathbf{w}^k$, called its *prototype*. The degree of confidence for predicting category k with respect to an instance $\mathbf{x}$[4] (both in online training and testing) is determined by the similarity between the instance and the prototype — the inner product $\mathbf{w}^k \cdot \mathbf{x}$. PA predicts a ranked list of categories according to this confidence.

PA is a family of online and large-margin based classifiers. Given an instance $(\mathbf{x}_t, y_t)$ during online learning, the multiclass margin $marg$ in PA[5] is the difference between the score of the true category y_t and that of the highest ranked false positive category s, i.e.

$$marg = \mathbf{w}^{y_t} \cdot \mathbf{x_t} - \mathbf{w}^s \cdot \mathbf{x_t} \tag{1}$$

where $s = \arg\max_{s \neq y_t} \mathbf{w}^s \cdot \mathbf{x_t}$.

A positive margin value indicates that the algorithm makes a correct prediction. One is however not only satisfied with a positive margin value, but also seeks to achieve a margin value of at least 1. When this is not satisfied, the online algorithm suffers a multiclass hinge loss:

$$\mathcal{L}_{mc}(\mathbf{w}; (\mathbf{x}_t, y_t)) = \begin{cases} 0 & marg \geq 1 \\ 1 - marg & \text{otherwise} \end{cases}$$

where $\mathbf{w} = (\mathbf{w}^1, .., \mathbf{w}^C)$ denotes the concatenation of the C prototypes (into a vector).

In an online learning step, the PA-II variant updates the category prototype with the solution of this constrained optimization problem,

$$\begin{aligned} \mathbf{w}_{t+1} &= \arg\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|^2 + \mathcal{A}\xi^2 && (2) \\ s.t. \quad & \mathcal{L}_{mc}(\mathbf{w}; (\mathbf{x}_t, y_t)) \leq \xi. && (3) \end{aligned}$$

Essentially, if the margin is met (also implying no misclassification), PA *passively* accepts the current solution. Otherwise, PA *aggressively* learns the new prototype which satisfies the loss constraint and stays as close to the one previously learned as possible. To cope with *label noise*, PA-II introduces a slack variable ξ in the optimization

[1] See *www.nba.com/playerfile/michael_jordan/index.html*

[2] See *www.eecs.berkeley.edu/∼jordan/*

[3] See *http://www.dmoz.org/about.html* for details.

[4] $\mathbf{x}$ is the vector representation of word frequencies of the corresponding document, L_2 normalized.

[5] For brevity of presentation, we consider the *single label* multiclass categorization setting.

for a gentler update, a technique previously employed to derive soft-margin classifiers (Vapnik, 1998). $\mathcal{A}$ is a parameter that controls the *aggressiveness* of the update.

The solution to the above optimization problem amounts to only changing the two prototypes violating the margin in the update step:

$$\mathbf{w}_{t+1}^{y_t} = \mathbf{w}_t^{y_t} + \tau \mathbf{x}_t \qquad \mathbf{w}_{t+1}^{s} = \mathbf{w}_t^{s} - \tau \mathbf{x}_t$$

where $\tau = \frac{\mathcal{L}_{mc}}{\|\mathbf{x}_t\|^2 + \frac{1}{2\mathcal{A}}}$.

To conclude, PA treats the hierarchy as flat categories for multiclass classification. It is similar to Multiclass Perceptron (Crammer and Singer, 2003) but only updates two vectors per iteration and thus is more efficient.

3.2 Categories as Coreference Evidence

Conceptually, the text categorization component can be viewed as a function that maps a document $\mathbf{d}$ to a ranked list of top K categories along with their respective confidence scores, i.e.

$$\phi(\mathbf{d}) = \{< c_1, s_1 >, .., < c_K, s_K >\}$$

We leverage these document categories to measure the pairwise similarity of any two documents, sim$(\phi(\mathbf{d}^u), \phi(\mathbf{d}^v))$, for entity disambiguation. Given a taxonomy $\mathcal{T}$, we first formally define the *affinity* between a category c and one of its ancestor category c' in $\mathcal{T}$ as:

$$\text{affinity}(c; c') = 1 - \frac{len(c, c')}{depth(\mathcal{T})}$$

where len is the length of the shortest path between the two categories and *depth(T)* denotes the depth of the taxonomy. In other words, affinity is the complementary of the normalized path length between c and its ancestor c'.

Using graph theory terminology, LCA(c_1, c_2) denote the *lowest common ancestor* of two categories c_1 and c_2 in $\mathcal{T}$. Given two category lists, $\phi(\mathbf{d}^u) = \{< c_1^u, s_1^u >, .., < c_K^u, s_K^u >\}$ and $\phi(\mathbf{d}^v) = \{< c_1^v, s_1^v >, .., < c_K^v, s_K^v >\}$, we use the $LCA(c_i^u, c_j^v)$ of each category pair c_i^u and c_j^v as the basis to measure similarity. Formally, we transform $\phi(\mathbf{d}^u)$ to a $K \times K$ dimensional vector:

$$\vec{\mathbf{v}}(\mathbf{d}^u) = [\text{affinity}(c_i^u; LCA(c_i^u, c_j^v)) \cdot s_i^u]^T \quad (4)$$

where $i, j = 1..K$. In other words, we project $\phi(\mathbf{d}^u)$ into a vector in the space spanned by the LCAs of category pairs. Using the same bases, we can derive $\vec{\mathbf{v}}(\mathbf{d}^v)$ analogically.

With this transformation, $\phi(\mathbf{d}^u)$ and $\phi(\mathbf{d}^v)$ are expressed in the common bases, i.e. their LCAs. Therefore, the similarity between the top K categories of two documents can be measured by the inner product of these two vectors:

$$\text{sim}(\phi(\mathbf{d}^u), \phi(\mathbf{d}^v)) = \vec{\mathbf{v}}(\mathbf{d}^u) \cdot \vec{\mathbf{v}}(\mathbf{d}^v) \quad (5)$$

3.3 Empirical Studies

To handle the diverse topics of Web documents, we leverage the ODP data to train the many class categorization algorithm. The public ODP data contains 361,621 categories and links to over 2 million pages. We crawled the original web pages from these links, which yielded 1.9 million pages (50GB in size). The taxonomy was condensed to depth three[6] and then very rare categories (having less than 5 instances) were discarded. The data set is created with these categories and the vector representation of the term weights of the extracted raw text. This dataset has 1,889,683 instances and 4,891 categories in total. Finally, stratified 80-20 split was performed on this dataset, i.e. 1.5M pages for training and 377K pages for testing.

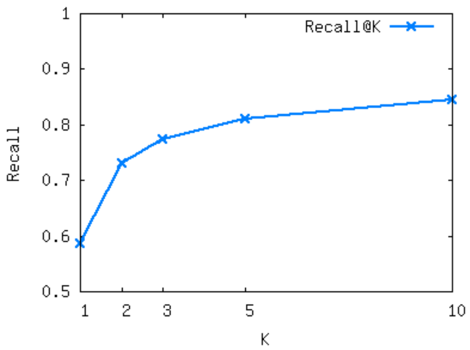


Figure 1: Categorization performance at different positions in the ODP test set.

As we view the taxonomy as a set of flat categories and we are interested in the top K categories, we use the recall at K metric for evaluation. Recall at K is defined as the percentage of instances having their true category ranked

[6] The original taxonomy has average depth 7, which is too deep for the coreference purpose in this work and many categories have too few instances for training.

among the top K slots in the category list. For a single label dataset (most ODP pages have one category) and $K = 1$, this is the accuracy metric in multiclass classification. Note that in the many class setting, recall at 1 is a very strict metric as no credit is given for predicting the parent, children or sibling categories; also, documents may have valid secondary topics not labeled by humans. Figure 1 shows recall at K in the test set. We observe that the algorithm is able to predict the category for 58.7% of the instances in the first rank and more than 77% in top three. There is only diminishing gains when we consider the categories further down the list. Hence we choose to use the similarity of the top 1 and top 3 categories (named `TC1` and `TC3`, respectively) and study their contributions for the CDC task.

3.4 Remarks

In this section, the entire document in the representation of its categories is used as a unit of analysis for CDC. Categorization based CDC works best with namesakes appearing in documents of relatively heterogenous topics, which is usually the case for web documents. Indeed, experienced web searchers would add terms such as 'baseball player' to the name search queries for more relevant results; Wikipedia also (manually) disambiguates namesakes by their professions. Categorization can also be adopted as a robust faceted search system for handling name search queries: users select the interested category/facet to efficiently disambiguate and filter out irrelevant results. The majority of web persons can be readily distinguished by the different underlying categories of the documents where they appear. For more homogeneous corpora or less benevolent cases, the next sections introduce two complementary CDC strategies.

4 Information Extraction for CDC

Consider the following two snippets retrieved with regard to the query 'George Bush':

[Snippet 1]: *"George W. Bush and Bill Clinton are trying to get Congress to allow Haiti to triple the number of exports ..."*

[Snippet 2]: *"George H. W. Bush succeeded Reagan as the 41st U.S. President."*

Using categories alone in this case is insufficient as both will be assigned similar categories such as 'Politics' or 'History/U.S.'. Also, it's not uncommon for these entities to co-occur in the same document and thus making them even more confounding. Properly disambiguating these two mentions requires the usage of local information: for instance, the extraction of full names, the detection of co-occurring NEs and contextual information. We introduce an IE system that extracts precise disambiguation evidence in this section and describe using the extraction context as additional information in the next section.

Our CDC system leverages a state-of-the-art commercial IE system AeroText (Taylor, 2004). The IE system employs manually created knowledge bases with statistically trained models to extract named entities, detect, classify and link relations between NEs. A summary of the most important IE-based features that we use are listed in Table 1. Based on the extracted attributes and relations, we further define their pairwise similarity used as coreference features. This ranges from simple compatibility checking for 'gender', textual soft matching for 'names', to sophisticated semantic matching for 'mentions' and 'locations' using WordNet. (Huang et al., 2009) provides more detailed discussions on the development of these IE based coreference features.

We note that several existing state-of-the-art IE systems are also capable of extracting these features. In particular, Named Entity Recognition (NER) which focuses on a small set of predefined categories of named entities (e.g. persons, organization, location) as well as the detection and tracking of preselected relations have achieved venerable empirical success in practice[7]. Also, within document coreference is a mature and well-studied technology in NLP (e.g. (Ng and Cardie, 2002)). Therefore, our CDC system can readily adopt alternative IE toolkits.

5 Context Matching

As mentioned earlier, achieving high extraction accuracy and coverage for diverse web documents

[7] The Automatic Content Extraction (ACE) evaluation and the Text Analysis Conference (TAC) also have IE-based entity tracking tasks that are relevant to this component.

is still a challenging and open research problem even for the state-of-the-art IE systems. We note that one of the natural outcomes from extraction is the context of the NE of interest, which covers the NE with its surrounding text. For a specific NE, our CDC system uses the context built from the sentences which form the NE's within document coreference chain. The context is then represented as a term vector whose terms are weighted by the TF-IDF weighing scheme. For a pair of NEs, the context matching component measures the cosine similarity of their context term vectors.

Essentially, this component alone is similar to the method presented in the seminal CDC work in (Bagga and Baldwin, 1998b). We however note that simply applying a predetermined threshold on the context similarity for CDC as in this earlier work is not sufficient. First, this method narrowly focuses on the local word occurrence and may miss the *big picture*, i.e. the correlation that exists in the global scope of a document. Also, mere word occurrence is incapable of accounting for the variation of word choices or placing special emphases on evidence such as co-occurring named entities, relations, etc. The categorization and IE components presented earlier in this work overcome these two pitfalls of the simple IR-based approach. We will further showcase the advantage of our comprehensive approach in section 7.2.

6 Composite Pairwise Coreference

In the previous sections, we describe the components to obtain document, sub-document and entity level disambiguation evidence in detail. In this section, we propose to use Random Forest (RF) to combine the experts components into one single composite pairwise similarity score. RF is an ensemble classifier, composed of a collection of randomized decision trees (Breiman, 2001). Each randomized tree is built on a different bootstrap sample of the training data. Randomness is also introduced into the tree construction process: the variable selection for each split is conducted not on the entire feature set, but from a small random subset of features. Gini index is used as the criteria in selecting the best split. Additionally, each tree is unpruned, to keep the prediction bias low. By aggregating many trees that are lowly-correlated (through bootstrap sampling and random variable selection), RF also reduces the prediction variance.

An ensemble method such as Random Forests is very suitable for the CDC task. First, the collection of randomized decision trees is analogous to a panel of different experts, where each makes its decision using different criteria and different features. Previously, RF has been used to aggregate various features in the author disambiguation task (Treeratpituk and Giles, 2009). One of the significant challenges in combining these different features in our CDC setting is that not all of them are always active. For instance, the IE tool may extract an employment relation for one entity and a list relation for another. Also, when the IE tool cannot infer the gender information or when the categorization component does not confidently predict the top K categories (e.g. all with low scores), it's desirable to not supply those features for coreferencing. The traditional technique to impute the missing values, e.g. by replacing them with the mean value, is not suitable in this case. In our work, we specify a special level 'NA' in the decision tree base learner. In our development set, this treatment improves pairwise coreference accuracy by more than 6%.

Figure 2 shows the convergence plot of the composite pairwise coreference function based on Random Forest[8]. We observe that the Out-Of-Bag

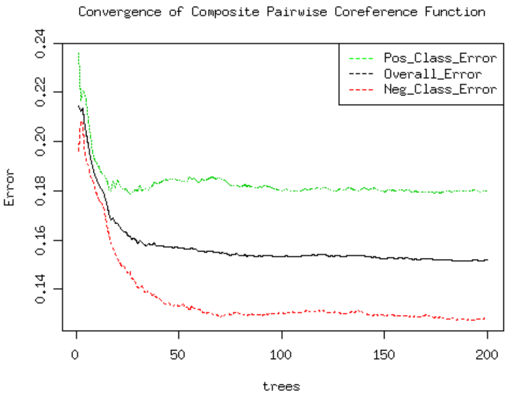


Figure 2: Convergence of OOB errors of the composite pairwise coreference function using the training portion of the WePS dataset.

[8] The R random forest (Liaw and Wiener, 2002) was used.

(OOB) errors [9] drastically decrease with the first 50 trees and then level off (without signs of overfitting). Thus we choose to use the model built with the first 100 trees for prediction. Overall, our model can achieve more than 85% accuracy for pairwise coreference prediction.

7 Experiments

We evaluate our CDC approach with the benchmark dataset from the ACL-2007 SemEval Web Person Search (WePS) evaluation campaign (Artiles et al., 2007). The WePS task is: given a name search query, cluster the search result *documents* according to the underlying referents. Compared to the CDC task which clusters *mention level* entities, a simplifying assumption is made in this task that each document refers to only one identity with respect to the query. The WePS dataset contains the training and test set. The training set contains the top 100 web search results of 49 names from the Web03 corpus (Mann and Yarowsky, 2003), Wikipedia and European Conference on Digital Library (ECDL) participants; the test data are comprised of the top 100 documents of 30 names from Wikipedia, US Census and ACL participants.

Table 1: Expert component and their feature sets.

Feature	Component	Description
TC1	Categorization	Sim. of the top 1 categories
TC3		Sim. of the top 3 categories
CNTX	Context	Sim. of context
NAME	IE (attribute)	Sim. of full/first/last names
MENT		Sim. of mentions
GEND		Sim. of genders
EMP	IE (relation)	Sim. of full/first/last names
LIST		Sim. of co-occurring persons
LOC		Sim. of locations
FAM		Sim. of family members

7.1 Evaluation of Pairwise Coreference

We conduct a thorough study of the importance of the individual expert components and their features with the WePS training set. Table 1 shows the three components of the systems, their main features and descriptions.

The importance of these expert components and their features are illustrated in Figure 3. One of the most important features is CNTX, this confirms that the prior work on CDC (e.g. (Bagga and Baldwin, 1998b)) can achieve good results with the IE-driven context similarity feature (or its variation). The text categorization component also contributes very important features. In particular, TC3 is more significant than TC1 for reducing the Gini index because it recalls more correct categories. On the other hand, TC1 is slightly more important than TC3 for its contribution to accuracy, indicating TC1 is more precise (with less noise categories). For the IE component, attribute features NAME and MENT are the most useful. As aforementioned, the IE component may not always extract the relation features such as EMP, LIST, LOC and FAM, and hence they seemingly have limited effect on model learning (with relatively low reduction in Gini index). These relation features are however very accurate when extracted and are present for prediction. Therefore, they are strong disambiguation evidence and their removal would significantly hamper performance.

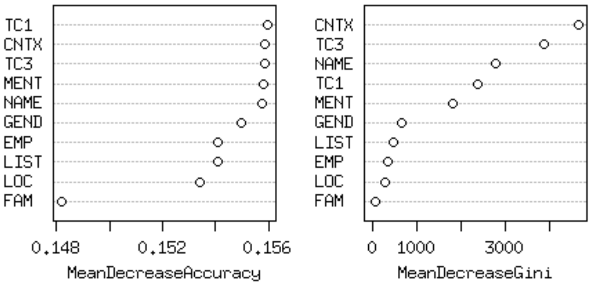


Figure 3: Importance of the expert components and their features found by Random Forest (note the small spread in MeanDecreaseAccuracy).

7.2 Evaluation for Web Person Search

Using the confidence of the pairwise coreference prediction as a distance metric, we adopt a density-based clustering method DBSCAN (Ester et al., 1996) as in (Huang et al., 2006)[10] to induce the person clusters. The final set of evaluation is based on these person clusters generated for the WePS test set.

Two sets of metrics are used to evaluate the overall system. First, we use the B-CUBED

[9] OOB error is an unbiased estimate of test error in RF (Breiman, 2001), computed as the average misclassification rates of each tree with samples not used for its construction.

[10] DBSCAN is a robust and scalable algorithm suitable for clustering relational data. In interest of space, we refer readers to (Ester et al., 1996) for the original algorithm.

Table 2: Cross document coreference performance (I. Pur. denotes inverse purity).

Method	Purity	I. Pur.	*F*	B-CUBED
CDC	0.812	0.796	**0.793**	**0.775**
CNTX	0.863	0.601	0.678	0.675
TC1+3	0.620	0.776	0.660	0.634
OIO	1.000	0.482	0.618	0.618
AIO	0.279	1.000	0.389	0.238

scores designed in (Bagga and Baldwin, 1998a) for evaluating cross document coreference performance. Second, we use the purity, inverse purity and their F score as in WePS (Artiles et al., 2007). Purity penalizes placing noise entities in a cluster, while inverse purity penalizes splitting coreferent entities into separate clusters.

Table 2 shows the performance of the macro-averaged cross document coreference performance on the WePS test sets. Note that though our evaluation is based on the mention level entities, the baselines One-In-One (OIO, placing each entity in a separate cluster) and All-In-One (AIO, putting all entities in one cluster) have almost identical results as those in the evaluation[11]. OIO can yield good performance, indicating that the names in test data are highly ambiguous. As alluded to in the title, context and categories both are very useful disambiguation features. CNTX is essentially very similar to the system presented in (Bagga and Baldwin, 1998b) and is a strong baseline[12] (outperforming $3/4$ of the systems in WePS). Note that CNTX has high purity but inferior inverse purity, indicating that using the context extracted by the IE system alone is unable to link many coreferent entities. Interestingly, we observe that using only the top-K categories (TC1+3) can also achieve competitive F score, though in a very different manner. TC1+3 recalls much more coreferent entities (significantly improving inverse purity), but at the same time also introduces noise.

Finally, adding document categories and using IE results (i.e. using all features in Table 1), our CDC system achieves 22% and 18% relative improvement compared to CNTX in F (purity) and B-CUBED scores, respectively. In particular, inverse purity improves by 46% relatively, implying that the additional evidence significantly improves the recall of coreferent entities (when there is a lack of context similarity in the traditional method). Overall, the comprehensive approach in this work outperforms the top-tiered systems in the WePS evaluation.

8 Conclusion and Future Work

This work proposes a synergy of three levels of analysis for the web cross document coreference task. On the document level, we use text categories, trained from thousands of ODP categories and over a million pages, as a concise representation of the documents. Categorization is a robust strategy for coreferencing web documents with diverse topics, formats and when there is a lack of extraction coverage or word matching. Two types of sub-document level evidence are also used in our approach. First, we apply an information extraction system to extract attributes and relations of named entities from the documents and perform within document coreference. Second, we use the context of the entities, a natural outcome of the IE system as a focused description of the named entity that may miss the extraction process. A CDC system has been implemented based on the IE and the text categorization components to provide a comprehensive solution to the web CDC task. We demonstrate the importance of each component in our system and benchmark our system with the WePS dataset which shows superior CDC performance.

There are a number of interesting directions for future research. Recently, Open IE was proposed in (Etzioni et al., 2008) for Web information extraction. This can be a more powerful alternative to traditional IE toolkits for Web CDC, though measuring the semantic similarity for a vast variety of relations can be another research issue. Employing external background knowledge such as Wikipedia (Han and Zhao, 2009) while maintaining scalability can also be an orthogonal direction for further improvement.

[11] Most person names in this set have only one underlying identity per document; thus the results are comparable despite the simplifying assumption of the WePS evaluation.

[12] We use context similarity 0.2 as the clustering threshold (which has the best performance in training data).

References

Artiles, Javier, Julio Gonzalo, and Satoshi Sekine. 2007. The SemEval-2007 WePS evaluation: Establishing a benchmark for the web people search task. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval)*, pages 64–69.

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

Bagga, Amit and Breck Baldwin. 1998b. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 36th ACL and 17th COLING*, pages 79–85.

Baron, Alex and Marjorie Freedman. 2008. Who is who and what is what: experiments in cross-document co-reference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 274–283.

Breiman, Leo. 2001. Random forests. *Machine Learning*, 45(1):5–32.

Chen, Ying and James Martin. 2007. Towards robust unsupervised personal name disambiguation. In *Proc. of EMNLP and CoNLL*, pages 190–198.

Crammer, Koby and Yoram Singer. 2003. A family of additive online algorithms for category ranking. *J. Machine Learning Research*, 3:1025–1058.

Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*, 7:551–585.

Ester, M., H. Kriegel, J. Sander, and X. Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd KDD Conference*, pages 226 – 231.

Etzioni, Oren, Michele Banko, Stephen Soderland, and Daniel S. Weld. 2008. Open information extraction from the web. *Communications of ACM*, 51(12):68–74.

Gooi, Chung H. and James Allan. 2004. Cross-document coreference on a large scale corpus. In *Proceedings of HLT-NAACL 2004*, pages 9–16.

Han, Xianpei and Jun Zhao. 2009. Named entity disambiguation by leveraging Wikipedia semantic knowledge. In *Proceedings of the 18th Conf. on Information and knowledge management (CIKM)*, pages 215–224.

Huang, Jian, Seyda Ertekin, and C. Lee Giles. 2006. Efficient name disambiguation for large scale databases. In *Proc. of 10th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 536 – 544.

Huang, Jian, Sarah M. Taylor, Jonathan L. Smith, Konstantinos A. Fotiadis, and C. Lee Giles. 2009. Profile based cross-document coreference using kernelized soft relational clustering. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 414–422.

Li, Xin, Paul Morie, and Dan Roth. 2004. Robust reading: Identification and tracing of ambiguous names. In *Proceedings of the Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 17–24.

Liaw, Andy and Matthew Wiener. 2002. Classification and regression by randomforest. *R News*, 2(3).

Madani, Omid and Jian Huang. 2008. On updates that constrain the features' connections during learning. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 515–523.

Mann, Gideon S. and David Yarowsky. 2003. Unsupervised personal name disambiguation. In *Proceedings of the seventh conference on Natural language learning (CoNLL)*, pages 33–40.

Ng, Vincent and Claire Cardie. 2002. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING)*, pages 1–7.

Niu, Cheng, Wei Li, and Rohini K. Srihari. 2004. Weakly supervised learning for cross-document person name disambiguation supported by information extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL)*, pages 598–605.

Taylor, Sarah M. 2004. Information extraction tools: Deciphering human language. *IT Professional*, 6(6):28 – 34.

Treeratpituk, Pucktada and C. Lee Giles. 2009. Disambiguating authors in academic publications using random forests. In *Proceedings of the ACM/IEEE Joint Conference on Digital libraries (JCDL)*, pages 39–48.

Vapnik, V. 1998. *Statistical Learning Theory*. John Wiley and Sons, Inc., New York.

Wan, Xiaojun, Jianfeng Gao, Mu Li, and Binggong Ding. 2005. Person resolution in person search results: WebHawk. In *Proceedings of the 14th ACM International Conference on Information and Knowledge management (CIKM)*, pages 163–170.

Feature-Rich Discriminative Phrase Rescoring for SMT

Fei Huang and **Bing Xiang**

IBM T. J. Watson Research Center
{huangfe, bxiang}@us.ibm.com

Abstract

This paper proposes a new approach to phrase rescoring for statistical machine translation (SMT). A set of novel features capturing the translingual equivalence between a source and a target phrase pair are introduced. These features are combined with linear regression model and neural network to predict the quality score of the phrase translation pair. These phrase scores are used to discriminatively rescore the baseline MT system's phrase library: boost good phrase translations while prune bad ones. This approach not only significantly improves machine translation quality, but also reduces the model size by a considerable margin.

1 Introduction

Statistical Machine Translation (SMT) systems, including phrase-based (Och and Ney 2002; Koehn et. al. 2003), syntax-based (Yamada and Knight 2001; Galley et. al. 2004) or hybrid systems (Chiang 2005; Zollmann and Venugopal 2006), are typically built with bilingual phrase pairs, which are extracted from parallel sentences with word alignment. Due to the noises in the bilingual sentence pairs and errors from automatic word alignment, the extracted phrase pairs may contain errors, such as

- dropping content words
  (the $num countries ,||个:<null>),
- length mismatch
  (along the lines of the || 的:of)
- content irrelevance
  (the next $num years, ||
  水平:level 方面:aspect 所:<null>)

These incorrect phrase pairs compete with correct phrase pairs during the decoding process, and are often selected when their counts are high (if they contain systematic alignment errors) or certain model costs are low (for example, when some source content words are translated into target function words in an incorrect phrase pair, the language model cost of the incorrect pair may be small, making it more likely that the pair will be selected for the final translation). As a result, the translation quality is degraded when these incorrect phrase pairs are selected.

Various approaches have been proposed over the past decade for the purpose of improving the phrase pair quality for SMT. For example, a term weight based model was presented in (Zhao, et al., 2004) to rescore phrase translation pairs. It models the translation probability with similarities between the query (source phrase) and document (target phrase). Significant improvement was obtained in the translation performance. In (Johnson, et al., 2007; Yang and Zheng, 2009), a statistical significance test was used to heavily prune the phrase table and thus achieved higher precision and better MT performance.

In (Deng, et al., 2008), a generic phrase training algorithm was proposed with the focus on phrase extraction. Multiple feature functions are utilized based on information metrics or word alignment. The feature parameters are optimized to directly maximize the end-to-end system performance. Significant improvement was reported for a small MT task. But when the phrase table is large, such as in a large-scale SMT system, the computational cost of tuning with this approach will be high due to many iterations of phrase extraction and re-decoding.

In this paper we attempt to improve the quality of the phrase table using discriminative phrase rescoring method. We develop extensive set of features capturing the equivalence of bilingual

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 492–500,
Beijing, August 2010

phrase pairs. We combine these features using linear and nonlinear models in order to predict the quality of phrase pairs. Finally we boost the score of good phrases while pruning bad phrases. This approach not only significantly improves the translation quality, but also reduces the phrase table size by 16%.

The paper is organized as follows: in section 2 we discuss two regression models for phrase pair quality prediction: linear regression and neural network. In section 3 we introduce the rich set of features. We describe how to obtain the training data for supervised learning of the two models in section 4. Section 5 presents some approaches to discriminative phrase rescoring using these scores, followed by experiments on model regression and machine translation in section 6.

2 Problem Formulation

Our goal is to predict the translation quality of a given bilingual phrase pair based on a set of features capturing their similarities. These features are combined with linear regression model and neural network. The training data for both models are derived from phrase pairs extracted from small amount of parallel sentences with hand alignment and machine alignment. Details are given in section 4.

2.1 Linear regression model

In the linear regression model, the predicted phrase pair quality score is defined as

$$Sco(e,f) = \sum_i \lambda_i f_i(e,f) \qquad (1)$$

where $f_i(e,f)$ is the feature for the phrase pair (*e,f*), as to be defined in section 3. These feature values can be binary (0/1), integers or real values. λ s are the feature weights to be learned from training data. The phrase pair quality score in the training data is defined as the sum of the target phrase's BLEU score (Papineni et. al. 2002) and the source phrase's BLEU score, where the reference translation is obtained from phrase pairs extracted from human alignment. Details about the training data are given in section 4. The linear regression model is trained using a statistical package R[1]. After training, the learned feature weights are applied on a held-out set of phrase pairs with known quality scores to evaluate the model's regression accuracy.

2.2 Neural Network model

A feed-forward back-propagation network (Bryson and Ho, 1969) is created with one hidden layer and 20 nodes. During training, the phrase pair features are fed into the network with their quality scores as expected outputs. After certain iterations of training, the neural net's weights are stable and its mean square error on the training set has been significantly reduced. Then the learned network weights are fixed, and are applied to the test phrase pairs for regression accuracy evaluation. We use MatLab™'s neural net toolkit for training and test.

We will compare both models' prediction accuracy in section 6. We would like to know whether the non-linear regression model outperforms linear regression model in terms of score prediction error, and if fewer regression errors correspond to better translation quality.

3 Feature Description

In this section we will describe the features we use to model the equivalence of a bilingual phrase pair (*e,f*). These features are defined on the phrase pair, its compositional units (words and characters), attributes (POS tags, numbers), co-occurrence frequency, length ratio, coverage ratio and alignment pattern.

- Phrase : $P_p(f \mid e)$, $P_p(e \mid f)$

$$P_p(e \mid f) = \frac{C(e,f)}{C(f)} \qquad (2)$$

where $C(e,f)$ is the co-occurrence frequency of the phrase pair (*e,f*), and C(*f*) is the occurrence frequency of the source phrase *f*. $P_p(f \mid e)$ is defined similarly.

- Word : $P_w(f \mid e)$, $P_w(e \mid f)$

$$P_w(e \mid f) = \prod_i \max_j t(e_i \mid f_j) \qquad (3)$$

where $t(e_i \mid f_j)$ is the lexical translation probability. This is similar to the word-level phrase

[1] http://www.r-project.org/

translation probability, as typically calculated in SMT systems (Brown et. al. 1993). Here we use max instead of sum. $P_w(f \mid e)$ is calculated similarly.

- Character: $P_c(f \mid e), P_c(e \mid f)$

When the source or target words are composed of smaller units, such as characters for Chinese words, or prefix/stem/suffix for Arabic words, we can calculate their translation probability on the sub-unit level. This is helpful for languages where the meaning of a word is closely related to its compositional units, such as Chinese and Arabic.

$$P_c(e \mid f) = \prod_i \max_n t(e_i \mid c_n) \qquad (4)$$

where c_n is the n-th character in the source phrase f (n=1,…,N).

- POS tag: $P_t(f \mid e), P_t(e \mid f)$

In addition to the probabilities estimated at the character, word and phrase levels based on the surface forms, we also compute the POS-based phrase translation probabilities. For each source and target word in a phrase pair, we automatically label their POS tags. Then POS-based probabilities are computed in a way similar to the calculation of the word-level phrase translation probability (formula 3). It is believed that such syntactic information can help to distinguish good phrase pairs from bad ones (for example, when a verb is aligned to a noun, its POS translation probability should be low).

- Length ratio

This feature computes the ratio of the number of content words in the source and target phrases. It is designed to penalize phrases where content words in the source phrase are dropped in the target phrase (or vice versa). The ratio is defined to be 10 if the target phrase has zero content word while the source phrase has non-zero content words. If neither phrase contains a content word, the ratio is defined to be 1.

- Log frequency

This feature takes the logarithm of the co-occurrence frequency of the phrase pair. High frequency phrase pairs are more likely to be correct translations if they are not due to systematic alignment errors.

- Coverage ratio

We propose this novel feature based on the observation that if a phrase pair is a correct translation, it often includes correct sub-phrase pair translations (*decomposition*). Similarly a correct phrase pair will also appear in correct longer phrase pair translations (*composition*) unless it is a very long phrase pair itself. Formally we define the coverage ratio of a phrase pair (*e,f*) as:

$$Cov(e,f) = Cov_d(e,f) + Cov_c(e,f). \quad (5)$$

Here $Cov_d(e,f)$ is the decomposition coverage:

$$Cov_d(e,f) = \sum_{f_i \subseteq f} \frac{\sum_{(e_i,f_i)\in P_L} \Delta(e_i, e)}{\sum_{(*,f_i)\in P_L} 1} \qquad (6)$$

where f_i is a sub-phrase of f, and (e_i, f_i) is a phrase pair in the MT system's bilingual phrase library P_L. $\Delta(e_1, e_2)$ is defined to be 1 if $e_1 \subseteq e_2$, otherwise it is 0. For each source sub-phrase f_i, this formula calculates the ratio that its target translation e_i is also a sub-phrase of the target phrase *e*, then the ratio is summed over all the source sub-phrases.

Similarly the composition coverage is defined as

$$Cov_c(e,f) = \sum_{f \subseteq f^j} \frac{\sum_{(e^j,f^j)\in P_L} \Delta(e, e^j)}{\sum_{(*,f^j)\in P_L} 1} \qquad (7)$$

where f^j is any source phrase containing f and e^j is one of f^j's translations in P_L. We call f^j a super-phrase of *f*. For each source super-phrase f^j, this formula calculates the ratio that its target translation e^j is also a super-phrase of the target phrase *e*, then the ratio is summed over all the source super-phrases.

Short phrase pairs (such as a phrase pair with one source word translating into one target word) have less sub-phrases but more super-phrases (for long phrase pairs, it is the other way around).

Combining the two coverage factors produces balanced coverage ratio, not penalizing too short or too long phrases.

- Number match

During preprocessing of the training data, numbers are mapped into a special token ($num) for better generalization. Typically one number corresponds to one special token. During translation numbers should not be arbitrarily dropped or inserted. Therefore we can check whether the source and target phrases have the right number of $num to be matched. If they are the same the number match feature has value 1, otherwise it is 0.

- Alignment pattern

This feature calculates the number of *unaligned* content words in a given phrase pair, where word alignment is obtained simply based on the maximum lexical translation probability of the source (target) word given all the target (source) words in the phrase pair.

Among the above 13 features, the number match feature is a binary feature, the alignment pattern feature is an integer-value feature, and the rest are real-value features. Also note that most features are positively correlated with the phrase translation quality (the greater the feature value, the more likely it is a correct phrase translation) except the alignment pattern feature, where more unaligned content words corresponds to bad phrase translations.

4 Training Data

The training data for both the linear regression and neural network models are bilingual phrase pairs with the above 13 feature values as well as their expected phrase quality scores. The feature values can be computed according to the description in section 3. The expected translation quality score for the phrase pair (*e*,*f*) is defined as

$$B(e,f)=Bleu(e,e^{*}\mid f)+Bleu(f,f^{*}\mid e) \quad (8)$$

where e^{*} is the human translation of the source phrase *f*, and f^{*} is the human translation of the target phrase *e*. These human translations are obtained from hand alignment of some parallel sentences.

1. Given hand alignment of some bilingual sentence pairs, extract gold phrase translation pairs.
2. Apply automatic word alignment on the same bilingual sentences, and extract phrase pairs. Note that due to the word alignment errors, the extracted phrase pairs are noisy.
3. For each phrase pair (*e*, *f*) in the noisy phrase table, find whether the source phrase *f* also appears in the gold phrase table as (*e**, *f*). If so, use the corresponding target phrase(s) *e** as reference translation(s) to evaluate the BLEU score of the target phrase *e* in the noisy phrase table.
4. Similarly, for each *e* in (*e*, *f*), identify (*e*, *f**) in the gold phrase table and compute the BLEU score of *f* using *f** as the reference.
5. The sum of the above two BLEU scores is the phrase pair's translation quality score.

5 Phrase Rescoring

Given the bilingual phrase pairs' quality score, there are several ways to use them for statistical machine translation.

5.1 Quality score as a decoder feature

A straightforward way is to use the quality scores as an additional feature in the SMT system, combined with other features (phrase scores, word scores, distortion scores, LM scores etc.) for MT hypotheses scoring. The feature weight can be empirically learned using manual tuning or automatic tuning such as MERT (Och 2003). In this situation, all the phrase pairs and their quality scores are stored in the MT system, which is different from the following approach where incorrect phrase translations are pruned.

5.2 Discriminative phrase rescoring

Another approach is to select good and bad phrase pairs based on their predicted quality scores, then discriminatively rescore the phrase pairs in the baseline phrase library. We sort the phrase pairs based on their quality scores in a decreasing order. The bottom *N* phrase pairs are considered as incorrect translations and pruned from the phrase library. The top M phrase pairs P_M are considered as good phrases with correct translations. As identifying correct sub-phrase translation requires accurate word alignment within phrase pairs, which is not easy to obtain due to the lack of rich context information within the phrase pair, we only boost the good phrase pairs' super-phrases in the phrase library. Given a phrase pair (*e*,*f*) with phrase co-occurrence count *C*(*e*,*f*), the weighted co-occurrence count is defined as:

$$C'(e,f) = C(e,f) \prod_{(e_i,f_i)\in(e,f)} b_i \qquad (9)$$

where (e_i, f_i) is a *good* sub-phrase pair of (*e*,*f*) belonging to P_M, with quality score b_i. Note that if (*e*,*f*) contains multiple good sub-phrase pairs, its co-occurrence count will be boosted multiple times. Here the boost factor is defined as the product of quality scores of good sub-phrase pairs. Instead of product, one can also use sum, which did not perform as well in our experiments. The weighted co-occurrence count is used to calculate the new phrase translation scores:

$$P'(e \mid f) = \frac{C'(e,f)}{\sum C'(*,f)} \qquad (10)$$

$$P'(f \mid e) = \frac{C'(e,f)}{\sum C'(e,*)} \qquad (11)$$

which replace the original phrase translation scores in the SMT system. In addition to phrase co-occurrence count rescoring, the quality scores can also be used to rescore word translation lexicons by updating word co-occurrence counts accordingly.

6 Experiments

We conducted several experiments to evaluate the proposed phrase rescoring approach. First we evaluate the two regression models' quality score prediction accuracy. Secondly, we apply the predicted phrase scores on machine translation tasks. We will measure the improvement on translation quality as well as the reduction of model size. Our experiments are on English-Chinese translation.

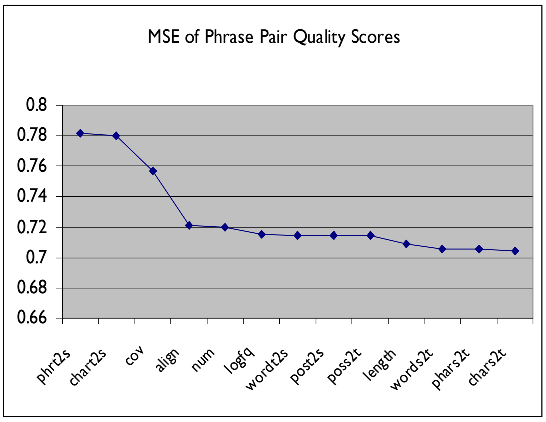


Figure 1. Linear regression model phrase pair prediction MSE curve. Errors are significantly reduced when more features are introduced (phrs2t /phrt2s: phrase source-to-target/target-to-source features; words2t/wordt2s: word-level; chars2t/chart2s: character-level; poss2t/post2s: POS-level; cov: coverage ratio; align: alignment pattern; logfq: log frequency; num: number match; length: length ratio).

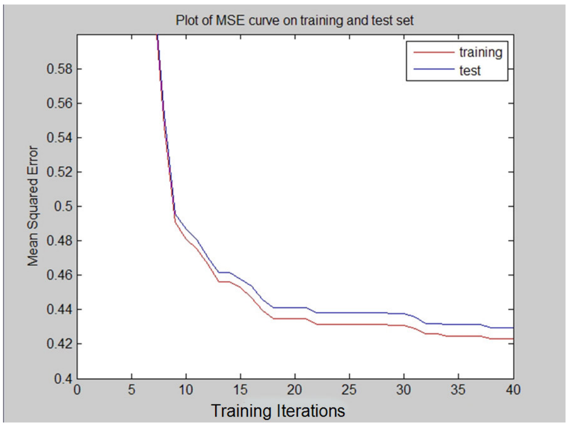


Figure 2. Neural network model phrase pair prediction MSE curve. Errors are significantly reduced with more training iterations.

6.1 Regression model evaluation

We select 10K English-Chinese sentence pairs with both hand alignment and automatic HMM alignment, and extract 106K phrase pairs with true phrase translation quality scores as computed according to formula 8. We choose 53K phrase pairs for regression model training and another 53K phrase pairs for model evaluation. There are 14 parameters to be learned (13 feature weights plus an intercept parameter) for the linear regression model, and 280 weights (13×20

<table>
<tr><th></th><th>Linear Regression</th><th>Neural Network</th></tr>
<tr><td><b>Good</b> phrase pairs</td><td>and|和|5.52327
amount|金额 数量|4.03006
us|, 美 -|3.91992
her husband|她 丈夫|3.85536
the program|节目 , 一|3.81078
the job|了 这 份 工作|3.77406
shrine|; 靖国神社|3.74336
of course ,|, 当然 , 就 是|3.7174
is only|只 能 是 这|3.69426
visit|访问 只|3.67256
facilities and|设施 , 并 在|3.65402</td><td>rights|权利 |6.96817
has become|已 成为 |4.16468
why|为甚么 |3.82629
by armed|受 武装 |3.62988
o|O |3.47795
of drama|在 戏剧 |3.36601
government and|政府 及 |3.27347
introduction|引进 |3.19113
heart disease|心脏 疾病 |3.11829
heads|首脑们 |3.05467
american consumers|美国 消费者 |2.99706</td></tr>
<tr><td><b>Bad</b> phrase pairs</td><td>as well|及 其|1.03234
closed|落下 帷幕|1.01271
she was|梅克尔|0.99011
way|改为 双程|0.955918
of a|出 一 种|0.914717
knowledge|察觉|0.875116
made|出席 "|0.837358
the|保持 联络|0.801142
end|之前|0.769938
held|而 进行 的|0.742588</td><td>letter|致函 贵会 |0.39203
, though|尽管 它 |0.37020
levels of|各 级 落实 |0.34892
- board|面板 |0.32826
number of|批 举报 |0.30499
indonesia|苏马尔佐托 |0.27827
xinhua at|$num |0.24433
provinces|安徽 |0.20281
new .|新鲜 之 处 的 , |0.15430
can|的 不同 |0.09502</td></tr>
</table>

Table 2. Examples of good and bad phrase pairs based on the linear regression model and neural network's predicted quality scores.

for the input weight matrix plus 20×1 for the output weight vector) for the neural network model. In both cases, the training data size is much more than the parameters size, so there is no data sparseness problem.

After the model parameters are learned from the training data, we apply the regression model to the evaluation data set, then compute the phrase quality score prediction mean squared error (MSE, also known as the average residual sum of squares):

$$MSE = \frac{1}{K}\sum_{k}\left[B_p(e_k, f_k) - B_t(e_k, f_k)\right]^2 \quad (12)$$

where B_p is the predicted quality score of the phrase pair (e_k, f_k), while B_t is the true score calculated based on human translations.

Figure 1 shows the reduction of the regression error in the linear regression model trained with different features. One may find that the MSE is significantly reduced (from 0.78 to 0.70) when additional features are added into the regression model.

Similarly, the neural network's MSE curve is shown in Figure 2. It can be seen that the MSE is significantly reduced with more iterations of training (from the initial error of 1.33 to 0.42 after 40 iterations).

Table 2 shows some phrase pairs with high/low quality scores predicted by the linear regression model and the neural network. One can see that both models assign high scores to good phrase translations and low scores to noisy phrase pairs. Although the values of these scores are beyond the range of [0, 2] as defined in formula 8, this is not a problem for our MT tasks, since they are only used as phrase boosting weights or pruning threshold.

6.2 Machine translation evaluation

We test the above phrase rescoring approach on English-Chinese machine translation. The SMT system is a phrase-based decoder similar to the description in (Tillman 2006), where various features are combined within the log-linear framework. These features include source-to-target phrase translation score based on relative frequency, source-to-target and target-to-source word-to-word translation scores, language model score, distortion model scores and word count. The training data for these features are 10M Chi-

	BLEU	NIST	Phrase Table Size
Baseline	38.67	9.3738	3.65M
LR-mtfeat	39.31	9.5356	3.65M
LR-boost (top30k)	39.36	9.5465	3.65M
LR-prune (tail600k)	39.06	9.4890	3.05M
LR-disc (top30K/tail600K)	39.75	9.6388	3.05M
NN-disc (top30K/tail600K)	39.76	9.6547	3.05M
LR-disc tuning	**39.87**	**9.6594**	**3.05M**
Significance-prune	38.96	9.3953	3.01M
Count-Prune	38.65	9.3549	3.05M

Table 3. Translation quality improvements with rescored phrase tables. Best result (1.2 BLEU gain) is obtained with discriminative rescoring by boosting top 30K phrase pairs and pruning bottom 600K phrase pairs, with some weight tuning.

nese-English sentence pairs, mostly newswire and UN corpora released by LDC. The parallel sentences have word alignment automatically generated with HMM and MaxEnt word aligner. Bilingual phrase translations are extracted from these word-aligned parallel corpora. Due to the noise in the bilingual sentence pairs and automatic word alignment errors, the phrase translation library contains many incorrect phrase translations, which lead to inaccurate translations, as seen in Figure 3.

Our evaluation data is NIST MT08 English-Chinese evaluation testset, which includes 1859 sentences from 129 news documents. The automatic metrics are BLEU and NIST scores, as used in the NIST 2008 English-Chinese MT evaluation. Note that as there is no whitespace as Chinese word boundary, the Chinese translations are segmented into characters before scoring in order to reduce the variance and errors caused by automatic word segmentation, which is also done in the NIST MT evaluation.

Table 3 shows the automatic MT scores using the baseline phrase table and rescored phrase tables. When the phrase quality scores from the linear regression model are used as a separate feature in the SMT system (***LR-mtfeat*** as described in section 5.1), the improvement is 0.7 BLEU points (0.16 in terms of NIST scores). By boosting the good phrase pairs (top 30K[2] phrase pairs, ***LR-boost***) from linear regression model, the MT quality is improved by 0.7 BLEU points over the baseline system. Pruning the bad phrase pairs (tail 600K phrase pairs) without using the quality scores as features (***LR-prune***) also improves the MT by 0.4 BLEU points. Combining ***LR-boost*** and ***LR_prune***, a discriminatively rescored phrase table (***LR-disc***) improved the BLEU score by 1.1 BLEU points, and reduce the phrase table size by 16% (from 3.6M to 3.0M phrase pairs). Manually tuning the boosting weights of good phrase pairs leads to additional improvement. Discriminative rescoring using the neural net work scores (***NN-disc***) produced similar improvement**.**

We also experiment with phrase table pruning using Fisher significant test, as proposed in (Johnson et. al. 2007). We tuned the pruning threshold for the best result. It shows that the significance pruning improves over the baseline by 0.3 BLEU pts with 17.5% reduction in phrase table, but is not as good as our proposed phrase rescoring method. In addition, we also show the MT result using a count pruning phrase table (Count-Prune) where 600K phrase translation pairs are pruned based on their co-occurrence counts. The MT performance of such phrase table pruning is slightly worse than the baseline MT system, and significantly worse than the result using the proposed rescored phrase table.

When comparing the linear regression and neural network models, we find rescoring with both models lead to similar MT improvements, even though the neural network model has much fewer regression errors (0.44 vs. 0.7 in terms of MSE). This is due to the rich parameter space of the neural network.

Overall, the discriminative phrase rescoring improves the SMT quality by 1.2 BLEU points and reduces the phrase table size by 16%. With statistical significance test (Zhang and Vogel 2004), all the improvements are statistically significant with p-value < 0.0001.

Figure 3 presents some English sentences, with phrase translation pairs selected in the final translations (the top one is from the baseline MT system and the bottom one is from the LR-disc system).

[2] These thresholds are empirically chosen.

Src	Indonesian bird flu victim contracted virus indirectly:
Baseline	<indonesian bird flu|印尼 禽流感> <virus|病毒> **<victim contracted|感染者>** <indirectly :|间接 :>
PhrResco	<indonesian bird flu|印尼 禽流感> **<victim|受害者>** **<contracted|感染>** <virus|病毒> <indirectly :|间接 :>
Src	The director of Palestinian human rights group Al-Dhamir, Khalil Abu Shammaleh, said he was also opposed to the move.
Baseline	<the director of|署长 的> <palestinian|巴勒斯坦> <human rights group|人权 团体> **<al -|" 基地 " 组织>** <,|,> **<abu|Abu>** <khalil|Khalil> <, said he was|表示 , 他> <also opposed to|也 反对> <the move .|这 项 行动 。>
PhrResco	<the director of|署长 的> <palestinian|巴勒斯坦> <human rights group|人权 团体> **<al -|al ->** <, khalil|, khalil> **<abu|阿布>** <, said he was|说 , 他> <also opposed to|也 反对> <the move .|这 项 行动 。>
Src	A young female tourist and two of her Kashmiri friends were among the victims.
Baseline	<a young female|有 一 名 年轻 女子> **<tourist and|旅游 和>** <$num of her|她 的 $num 个> <kashmiri|克什米尔> <friends were|网友> <among the|之间 的> <victims .|受害者 。>
PhrResco	<a young|一 个 年轻 的> <female|女性> **<tourist and|游客 和>** <$num of her|她 的 $num 个> <kashmiri|克什米尔> <friends were|朋友> <among the|之间 的> <victims .|受害者 。>

Figure 3. Examples of English sentences and their translation, with phrase pairs from baseline system and phrase rescored system. Highlighted text are initial phrase translation errors which are corrected in the PhrResco translations.

We find that incorrect phrase translations in the baseline system (as highlighted with blue bold font) are corrected and better translation results are obtained.

7 Conclusion

We introduced a discriminative phrase rescoring approach, which combined rich features with linear regression and neural network to predict phrase pair translation qualities. Based on these quality scores, we boost good phrase translations while pruning bad phrase translations. This led to statistically significant improvement (1.2 BLEU points) in MT and reduced phrase table size by 16%.

For the future work, we would like to explore other models for quality score prediction, such as SVM. We will want to try other approaches to utilize the phrase pair quality scores, in addition to rescoring the co-occurrence frequency. Finally, we will test this approach in other domain applications and language pairs.

References

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, Robert L. Mercer. 1993. *The Mathematics of Statistical Machine Translation: Parameter Estimation*, Computational Linguistics, v.19 n.2, June 1993.

Arthur Earl Bryson, Yu-Chi Ho. 1969. *Applied Optimal Control: Optimization, Estimation, and Control*. Blaisdell Publishing Company. p481.

David Chiang. 2005. *A Hierarchical Phrase-based Model for Statistical Machine Translation*. 2005. In Proc. of ACL, pp. 263–270.

Yonggang Deng, Jia Xu, and Yuqing Gao. 2008. *Phrase Table Training for Precision and Recall: What Makes a Good Phrase and a Good Phrase Pair*? In Proc. of ACL/HLT, pp. 81-88.

Michel Galley, Mark Hopkins, Kevin Knight, Daniel Marcu. 2004. *What's in a Translation Rule?* In Proc. of NAACL 2004, pp. 273-280.

Howard Johnson, Joel Martin, George Foster, and Roland Kuhn. 2007. *Improving Translation Quality by Discarding Most of the Phrase Table*. In Proc. of EMNLP-CoNLL, pp. 967-975.

Philipp Koehn, Franz Josef Och, Daniel Marcu. 2003. *Statistical Phrase-based Translation*, In Proc. of NAACL, pp. 48-54.

Franz Josef Och and Hermann Ney. 2003. *A Systematic Comparison of Various Statistical Alignment Models*, Computational Linguistics, v.29 n.1, pp.19-51, March 2003

Franz Josef Och. 2003. *Minimum Error Rate Training in Statistical Machine Translation*, In Proc. of ACL, 2003, pp. 160-167.

Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu. 2002. *BLEU: a Method for Automatic Evaluation of Machine Translation*, In Proc. of ACL, pp. 311-318.

Christoph Tillmann. 2006. *Efficient Dynamic Programming Search Algorithms for Phrase-based SMT*. In Proc. of the Workshop CHPSLP at HLT'06.

Kenji Yamada and Kevin Knight. 2001. *A Syntax-based Statistical Translation Model*, In Proc. of ACL, pp.523-530.

Mei Yang and Jing Zheng. 2009. *Toward Smaller, Faster, and Better Hierarchical Phrase-based SMT*. In Proc. of ACL-IJCNLP, pp. 237-240.

Ying Zhang and Stephan Vogel. 2004. *Measuring Confidence Intervals for the Machine Translation Evaluation Metrics*, In Proc. TMI, pp. 4-6.

Bing Zhao, Stephan Vogel, and Alex Waibel. 2004. *Phrase Pair Rescoring with Term Weighting for Statistical Machine Translation*. In Proc. of EMNLP, pp. 206-213.

Andreas Zollmann and Ashish Venugopal. 2006. *Syntax Augmented Machine Translation via Chart Parsing*. In Proc. of NAACL 2006- Workshop on statistical machine translation.

FactRank: Random Walks on a Web of Facts

Alpa Jain
Yahoo! Labs
alpa@yahoo-inc.com

Patrick Pantel
Microsoft Research
ppantel@microsoft.com

Abstract

Fact collections are mostly built using semi-supervised relation extraction techniques and wisdom of the crowds methods, rendering them inherently noisy. In this paper, we propose to validate the resulting facts by leveraging global constraints inherent in large fact collections, observing that correct facts will tend to match their arguments with other facts more often than with incorrect ones. We model this intuition as a graph-ranking problem over a fact graph and explore novel random walk algorithms. We present an empirical study, over a large set of facts extracted from a 500 million document webcrawl, validating the model and showing that it improves fact quality over state-of-the-art methods.

1 Introduction

Fact bases, such as those contained in *Freebase*, *DBpedia*, *KnowItAll*, and *TextRunner*, are increasingly burgeoning on the Internet, in government, in high tech companies and in academic laboratories. Bar the accurate manual curation typified by Cyc (Lenat, 1995), most fact bases are built using either semi-supervised techniques or wisdom of the crowds techniques, rendering them inherently noisy. This paper describes algorithms to validate and re-rank fact bases leveraging global constraints imposed by the semantic arguments predicated by the relations.

Facts are defined as instances of *n*-ary typed relations such as *acted-in*⟨*movie*, *actor*⟩, *director-of*⟨*movie*, *director*⟩, *born-in*⟨*person*, *date*⟩, and *buy*⟨*person*, *product*, *person*⟩. In all but very small fact bases, relations share an argument type, such as *movie* for the relations *acted-in* and *director-of* in the above example. The hypothesis explored in this paper is that when two fact instances from two relations share the same value for a shared argument type, then the validity of both facts should be increased. Conversely, we also hypothesize that an incorrect fact instance will tend to match a shared argument with other facts far less frequently. For example, consider the following four facts from the relations *acted-in*, *director-of*, and *is-actor*:

t_1: *acted-in*⟨Psycho, Anthony Perkins⟩
t_2: **acted-in*⟨Walt Disney Pictures, Johnny Depp⟩
t_3: *director-of*⟨Psycho, Alfred Hitchcock⟩
t_4: *is-actor*⟨Anthony Perkins⟩

Our confidence in the validity of t_1 increases with the knowledge of t_3 and t_4 since the argument *movie* is shared with t_3 and *actor* with t_4. Similarly, t_1 increases our confidence in the validity of t_3 and t_4. For t_2, we expect to find few facts that will match a *movie* argument with *Walt Disney Pictures*. Facts that share the *actor* argument *Johnny Depp* with t_2 will increase its validity, but the lack of matches on its *movie* argument will decrease its validity.

In this paper, we present *FactRank*, which formalizes the above intuitions by constructing a fact graph and running various random walk graph-ranking algorithms over it to re-rank and validate the facts. A collection of facts is modeled in the form of a graph where nodes are fact instances and edges connect nodes that have the same value for a shared argument type (e.g., t_1 would be linked by an edge to both t_3 and t_4.) Given a graph representation of facts, we explore various random walk algorithms to propagate our confidence in individual facts through the web of facts. We explore algorithms such as PageRank (Page et al., 1999) as well as propose novel algorithms that leverage several unique characteristics of fact graphs. Finally, we present an empirical analysis, over a large collection of facts extracted from a 500 mil-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 501–509,
Beijing, August 2010

lion document webcrawl, supporting our model and confirming that global constraints in a fact base can be leveraged to improve the quality of the facts. Our proposed algorithms are agnostic to the sources of a fact base, however our reported experiments were carried over a state-of-the-art semi-supervised extraction system. In summary, the main contributions of this paper are:

- We formalize the notion of ranking facts in a holistic manner by applying graph-based ranking algorithms (Section 2).
- We propose novel ranking algorithms using random walk models on facts (Section 3).
- We establish the effectiveness of our approach through an extensive experimental evaluation over a real-life dataset and show improvements over state-of-the-art ranking methods (Section 4).

2 Fact Validation Revisited

We denote an n-ary ***relation*** r with typed arguments $t_1, t_2, \cdots, t_n$ as $r\langle t_1, t_2, \cdots t_n\rangle$. In this paper, we limit our focus to unary and binary relations. A ***fact*** is an instance of a relation. For example, *acted-in*⟨*Psycho, Anthony Perkins*⟩ is a fact from the *acted-in*$\langle movie, actor\rangle$ relation.

Definition 2.1 [Fact base]: *A* fact base *is a collection of facts from several relations. Textrunner and Freebase are example fact bases (note that they also contain knowledge beyond facts such as entity lists and ontologies.)* □

Definition 2.2 [Fact farm]: *A* fact farm *is a subset of interconnected relations in a fact base that share arguments among them.* □

For example, consider a fact base consisting of facts for relations involving movies, organizations, products, etc., of which the relations *acted-in* and *director-of* could form a MOVIES fact farm.

Real-world fact bases are built in many ways. Semi-supervised relation extraction methods include *KnowItAll* (Etzioni et al., 2005), *TextRunner* (Banko and Etzioni, 2008), and many others such as (Riloff and Jones, 1999; Pantel and Pennacchiotti, 2006; Paşca et al., 2006; Mintz et al., 2009). Wisdom of the crowds methods include DBpedia (Auer et al., 2008) and Freebase which extracts facts from various open knowledge bases and allow users to add or edit its content.

Most semi-supervised relation extraction methods follow (Hearst, 1992). Starting with a relatively small set of seed facts, these extractors iteratively learn patterns that can be instantiated to identify new facts. To reflect their confidence in an extracted fact, extractors assign an *extraction score* with each fact. Methods differ widely in how they define the *extraction score*. Similarly, many extractors assign a *pattern score* to each discovered pattern. In each iteration, the highest scoring patterns and facts are saved, which are used to seed the next iteration. After a fixed number of iterations or when a termination condition is met, the instantiated facts are ranked by their *extraction score*.

Several methods have been proposed to generate such ranked lists (e.g., (Riloff and Jones, 1999; Banko and Etzioni, 2008; Matuszek et al., 2005; Pantel and Pennacchiotti, 2006; Paşca et al., 2006). In this paper, we re-implement the large-scale state-of-the-art method proposed by Paşca et al. (2006). This pattern learning method generates binary facts and computes the extraction scores of a fact based on (a) the scores of the patterns that generated it, and (b) the distributional similarity score between the fact and the seed facts. We computed the distributional similarity between arguments using (Pantel et al., 2009) over a large crawl of the Web (described in Section 4.1). Other implementation details follow (Paşca et al., 2006).

In our experiments, we observed some interesting ranking problems as illustrated by the following example facts for the *acted-in* relation:

id:	Facts (#Rank)
t_1:	*acted-in*⟨Psycho, Anthony Perkins⟩ (#26)
t_2:	**acted-in*⟨Walt Disney Pictures, Johnny Depp⟩ (#9)

Both t_1 and t_2 share similar contexts in documents (e.g., ⟨*movie*⟩ *film starring* ⟨*actor*⟩ *and* ⟨*movie*⟩ *starring* ⟨*actor*⟩), and this, in turn, boosts the pattern-based component of the extraction scores for t_1. Furthermore, due to the ambiguity of the term *psycho*, the distributional similarity-based component of the scores for fact t_2 is also lower than that for t_1.

Relations	id : Facts
acted-in	t_1 : ⟨Psycho, Anthony Perkins⟩
	t_2 : *⟨Walt Disney Pictures, Johnny Depp⟩
director-of	t_3 : ⟨Psycho, Alfred Hitchcock⟩
producer-of	t_4 : ⟨Psycho, Hilton Green⟩
is-actor	t_5 : ⟨Anthony Perkins⟩
	t_6 : ⟨Johnny Depp⟩
is-director	t_7 : ⟨Alfred Hitchcock⟩
is-movie	t_8 : ⟨Psycho⟩

Table 1: Facts share arguments across relations which can be exploited for validation.

Our work in this paper is motivated by the following observation: the ranked list generated by an individual extractor does not leverage any global information that may be available when considering a fact farm in concert. To understand the information available in a fact farm, consider a MOVIES fact farm consisting of relations, such as, *acted-in*, *director-of*, *producer-of*, *is-actor*, *is-movie*, and *is-director*. Table 1 lists sample facts that were generated in our experiments for these relations[1]. In this example, we observe that for t_1 there exist facts in foreign relations, namely, *director-of* and *producer-of* that share the same value for the *Movie* argument, and intuitively, facts t_3 and t_4 add to the validity of t_1. Furthermore, t_1 shares the same value for the *Actor* argument with t_5. Also, t_3, which is expected to boost the validity of t_1, itself shares values for its arguments with facts t_4 and t_7, which again intuitively adds to the validity of t_1. In contrast to this *web of facts* generated for t_1, the fact t_2 shares only one of its argument value with one other fact, i.e., t_6.

The above example underscores an important observation: *How does the web of facts generated by a fact farm impact the overall validity of a fact?* To address this question, we hypothesize that facts that share arguments with many facts are more reliable than those that share arguments with few facts. To capture this hypothesis, we model a web of facts for a farm using a graph-based representation. Then, using graph analysis algorithms, we propagate reliability to a fact using the scores of other facts that recursively connect to it.

Starting with a fact farm, to validate the facts in each consisting relation, we:

(1) Identify arguments common to relations in the farm.
(2) Run extraction methods to generate each relation.
(3) Construct a graph-based representation of the extracted facts using common arguments identified in Step (1) (see Section 3.1 for details on constructing this graph.)
(4) Perform link analysis using random walk algorithms over the generated graph, propagating scores to each fact through the interconnections (see Section 3.2 for details on various proposed random walk algorithms).
(5) Rank facts in each relation using the scores generated in Step (4) or by combining them with the original extraction scores.

For the rest of the paper, we focus on generating better ranked lists than the original rankings proposed by a state-of-the-art extractor.

3 FactRank: Random Walk on Facts

Our approach considers a fact farm holistically, leveraging the global constraints imposed by the semantic arguments of the facts in the farm. We model this idea by constructing a graph representation of the facts in the farm (Section 3.1) over which we run graph-based ranking algorithms. We give a brief overview of one such ranking algorithm (Section 3.2) and present variations of it for fact re-ranking (Section 3.3). Finally, we incorporate the original ranking from the extractor into the ranking produced by our random walk models (Section 3.4).

3.1 Graph Representation of Facts

Definition 3.1 *We define a* fact graph *FG(V, E), with V nodes and E edges, for a fact farm, as a graph containing facts as nodes and a set of edges between these nodes. An edge between nodes v_i and v_j indicates that the facts share the same value for an argument that is common to the relations that v_i and v_j belong to.* □

Figure 1 shows the fact graph for the example in Table 1 centered around the fact t_1.

Note on the representation: The above graph representation is just one of many possible options. For instance, instead of representing facts by nodes, nodes could represent the arguments of facts (e.g., *Psycho*) and nodes could be connected by edges if they occur together in a fact. The task of studying a "best" representation remains a future work direction. However, we believe that our proposed methods can be easily adapted to other such graph representations.

[1] The *is-actor*⟨actor⟩, *is-director*⟨director⟩, and *is-movie*⟨movie⟩ relations are equivalent to the relation *is-a*⟨c-instance, class⟩ where $class \in \{actor, director, movie\}$.

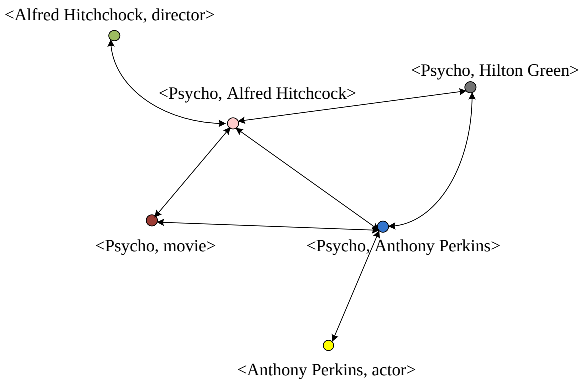


Figure 1: Fact graph centered around t_1 in Table 1.

3.2 The FactRank Hypothesis

We hypothesize that connected facts increase our confidence in those facts. We model this idea by propagating *extraction scores* through the fact graph similarly to how authority is propagated through a hyperlink graph of the Web (used to estimate the importance of a webpage). Several link structure analysis algorithms have been proposed for this goal, of which we explore a particular example, namely, PageRank (Page et al., 1999). The premise behind PageRank is that given the hyperlink structure of the Web, when a page v generates a link to page u, it confers some of its importance to u. Therefore, the importance of a webpage u depends on the number of pages that link to u and furthermore, on the importance of the pages that link to u. More formally, given a directed graph $G = (V, E)$ with V vertices and E edges, let $I(u)$ be the set of nodes that link to a node u and $O(v)$ be the set of nodes linked by v. Then, the importance of a node u is defined as:

$$p(u) = \sum_{v \in I(u)} \frac{p(v)}{|O(v)|} \tag{1}$$

The PageRank algorithm iteratively updates the scores for each node in G and terminates when a convergence threshold is met. To guarantee the algorithm's convergence, G must be irreducible and aperiodic (i.e., a connected graph). The first constraint can be easily met by converting the adjacency matrix for G into a stochastic matrix (i.e., all rows sum up to 1.) To address the issue of periodicity, Page et al. (1999) suggested the following modification to Equation 1:

$$p(u) = \frac{1-d}{|V|} + d \cdot \sum_{v \in I(u)} \frac{p(v)}{|O(v)|} \tag{2}$$

where d is a damping factor between 0 and 1, which is commonly set to 0.85. Intuitively, PageRank can be viewed as modeling a "random walker" on the nodes in G and the score of a node, i.e., PageRank, determines the probability of the walker arriving at this node.

While our method makes use of the PageRank algorithm, we can also use other graph analysis algorithms (e.g., HITS (Kleinberg, 1999)). A particularly important property of the PageRank algorithm is that the stationary scores can be computed for *undirected* graphs in the same manner described above, after replacing each undirected edge by a bi-directed edge. Recall that the edges in a fact graph are bi-directional (see Figure 1).

3.3 Random Walk Models

Below, we explore various random walk models to assign scores to each node in a fact graph *FG*.

3.3.1 Model Implementations

Pln: Our first method applies the traditional PageRank model to *FG* and computes the score of a node u using Equation 2.

Traditional PageRank, as is, does not make use of the *strength* of the links or the nodes connected by an edge. Based on this observation, researchers have proposed several variations of the PageRank algorithm in order to solve their problems. For instance, variations of random walk algorithms have been applied to the task of extracting important words from a document (Hassan et al., 2007), for summarizing documents (Erkan and Radev, 2004), and for ordering user preferences (Liu and Yang, 2008). Following the same idea, we build upon the discussion in Section 3.2 and present random walk models that incorporate the strength of an edge.

Dst: One improvement over ***Pln*** is to distinguish between nodes in *FG* using the extraction scores of the facts associated with them: extraction methods such as our reimplementation of (Paşca et al., 2006) assign scores to each output fact to reflect its confidence in it (see Section 3.2). Intuitively, a higher scoring node that connects to u should increase the importance of u more than a connection from a lower scoring node. Let $I(u)$ be the set of nodes that link to u and $O(v)$ be the set of nodes linked by v. Then, if $w(u)$ is the extraction score for the fact represented by node u, the score for node u is defined:

$$p(u) = \frac{1-d}{|V|} + d \cdot \sum_{v \in I(u)} \frac{w(v) \times p(v)}{|O(v)|} \quad (3)$$

where $w(v)$ is the confidence score for the fact represented by v. Naturally, other (externally derived) extraction scores can also be substituted for $w(v)$.

Avg: We can further extend the idea of determining the strength of an edge by combining the extraction scores of *both* nodes connected by an edge. Specifically,

$$p(u) = \frac{1-d}{|V|} + d \cdot \sum_{v \in I(u)} \frac{avg(u,v) \times p(v)}{|O(v)|} \quad (4)$$

where $avg(u,v)$ is the average of the extraction scores assigned to the facts associated with nodes u and v.

Nde: In addition to using extraction scores, we can also derive the strength of a node depending on the number of *distinct* relations it connects to. For instance, in Figure 1, t_1 is linked to four distinct relations, namely, *director-of*, *producer-of*, *is-actor*, *is-movie*, whereas, t_2 is linked to one relation, namely, *is-actor*. We compute $p(u)$ as:

$$p(u) = \frac{1-d}{|V|} + d \cdot \sum_{v \in I(u)} \frac{(\alpha \cdot w(v) + (1-\alpha) \cdot r(v)) \times p(v)}{|O(v)|} \quad (5)$$

where $w(v)$ is the confidence score for node v and $r(v)$ is the fraction of total number of relations in the farm that contain facts with edges to v.

3.3.2 Dangling nodes

In traditional hyperlink graphs for the Web, dangling nodes (i.e., nodes with no associated edges) are considered to be of low importance which is appropriately represented by the scores computed by the PageRank algorithm. However, an important distinction from this setting is that fact graphs are sparse causing them to have valid facts with no counterpart matching arguments in other relation, thus rendering them dangling. This may be due to several reasons, e.g., extractors often suffer from less than perfect recall and they may miss valid facts. In our experiments, about 10% and 40% of nodes from *acted-in* and *director-of*, respectively, were dangling nodes.

Handling dangling nodes in our extraction-based scenario is a particularly challenging issue: while demoting the validity of dangling nodes could critically hurt the quality of the facts, lack of global information prevents us from systematically introducing them into the re-ranked lists. We address this issue by maintaining the original rank positions when re-ranking dangling nodes.

3.4 Incorporating Extractor Ranks

Our proposed random walk ranking methods ignore the ranking information made available by the original relation extractor (e.g., (Paşca et al., 2006) in our implementation). Below, we propose two ways of combining the ranks suggested by the original ranked list O and the re-ranked list G, generated using the algorithms in Section 3.3.

R-Avg: The first combination method computes the average of the ranks obtained from the two lists. Formally, if $O(i)$ is the original rank for fact i and $G(i)$ is the rank for i in the re-ranked list, the combined rank $M(i)$ is computed as:

$$M(i) = \frac{O(i) + G(i)}{2} \quad (6)$$

R-Wgt: The second method uses a weighted average of the ranks from the individual lists:

$$M(i) = \frac{w_o \cdot O(i) + (1-w_o) \cdot G(i)}{2} \quad (7)$$

In practice, this linear combination can be learned; in our experiments, we set them to $w_o = 0.4$ based on our observations over an independent training set. Several other combination functions could also be applied to this task. For instance, we explored the *min and* max functions but observed little improvements.

4 Experimental Evaluation

4.1 Experimental Setup

Extraction method: For our extraction method, we reimplemented the method described in (Paşca et al., 2006) and further added a validation layer on top of it based on Wikipedia (we boosted the scores of a fact if there exists a Wikipedia page for either of the fact's arguments, which mentions the other argument.) This state-of-the-art method forms a *strong* baseline in our experiments.

Corpus and farms: We ran our extractor over a large Web crawl consisting of 500 million English

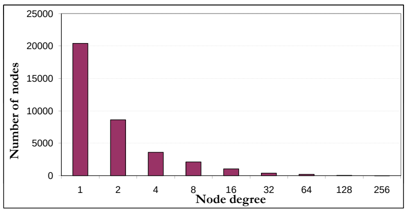


Figure 2: Degree distribution for MOVIES.

webpages crawled by the Yahoo! search engine. We removed paragraphs containing fewer than 50 tokens and then removed all duplicate sentences. The resulting corpus consists of over 5 million sentences. We defined a farm, MOVIES, with relations, *acted-in*, *director-of*, *is-movie*, *is-actor*, and *is-director*.

Evaluation methodology: Using our extraction method over the Web corpus, we generate over 100,000 facts for the above relations. However, to keep our evaluation manageable, we draw a random sample from these facts. Specifically, we first generate a ranked list using the extraction scores output by our extractor. We will refer to this method as ***Org*** (original). We then generate a fact graph over which we will run our methods from Section 3.3 (each of which will re-rank the facts). Figure 2 shows the degree, i.e., number of edges, distribution of the fact graph generated for MOVIES. We ran ***Avg***, ***Dst***, ***Nde***, ***R-Avg***, and ***R-Wgt*** on this fact graph and using the scores we re-rank the facts for each of the relations. In Section 4.2, we will discuss our results for the *acted-in* and *director-of* relations.

Fact Verification: To verify whether a fact is valid or not, we recruit human annotators using the paid service Mechanical Turk. For each fact, two annotations were requested (keeping the total cost under \$100). The annotators were instructed to mark incorrect facts as well as disallow any values that were not "well-behaved." For instance, *acted-in*$\langle$*Godfather, Pacino*$\rangle$ is correct, but *acted-in*$\langle$*The, Al Pacino*$\rangle$ is incorrect. We manually adjudicated 32% of the facts where the judges disagreed.

Evaluation metrics: Using the annotated facts, we construct a goldset S of facts and compute the precision of a list L as: $\frac{|L \cap S|}{|S|}$. To compare the effectiveness of the ranked lists, we use average precision, a standard measure in information retrieval for evaluating ranking algorithms, defined as: $A_p(L) = \frac{\sum_{i=1}^{|L|} P(i) \cdot isrel(i)}{\sum_{i=1}^{|L|} isrel(i)}$, where $P(i)$ is the precision of L at rank i, and *isrel(i)* is 1 if the fact at rank i is in S, and 0 otherwise. We also study the precision values at varying ranks in the list. For robustness, we report the results using 10-fold cross validation.

Method	Average precision		
	30%	50%	100%
Org	0.51	0.39	0.38
Pln	0.44	0.35	0.32
Avg	0.55	0.44	0.42
Dst	0.54	0.44	0.41
Nde	0.53	0.40	0.41
R-Avg	0.58	0.46	**0.45**
R-Wgt	**0.60**	**0.56**	0.44

Table 2: Average precision for *acted-in* for varying proportion of fact graph of MOVIES.

Method	Average precision		
	30%	50%	100%
Org	0.64	0.69	0.66
Pln	0.69	0.67	0.59
Avg	0.69	0.70	0.64
Dst	0.67	0.69	0.64
Nde	0.69	0.69	0.64
R-Avg	0.70	0.70	0.64
R-Wgt	**0.71**	**0.71**	**0.69**

Table 3: Average precision for *director-of* for varying proportion of fact graph of MOVIES.

4.2 Experimental Results

Effectiveness of graph-based ranking: Our first experiment studies the overall quality of the ranked lists generated by each method. Table 2 compares the average precision for *acted-in*, with the maximum scores highlighted for each column. We list results for varying proportions of the original fact graph (30%, 50%, and 100%). Due to our small goldset sizes, these results are not statistically significant over ***Org***, however we consistently observed a positive trend similar to those reported in Table 2 over a variety of evaluation sets generated by randomly building 10-folds of all the facts.

Overall, the ***Avg*** method offers a competitive alternative to the original ranked list generated by the extractor ***Org***: not only are the average precision values for ***Avg*** higher than ***Org***, but as we will see later, the rankings generated by our graph-based methods exhibits some positive unique characteristics. These experiments also

R	*Org*	*Pln*	*Avg*	*Dst*	*Nde*	*R-Avg*	*R-Wgt*
5	0.44	0.40	0.52	0.48	0.40	0.52	**0.56**
10	0.36	0.36	**0.42**	0.38	0.36	0.36	0.36
15	0.287	0.24	**0.30**	0.28	0.26	**0.30**	**0.30**
20	0.26	0.26	0.26	0.26	0.26	**0.27**	**0.27**
21	0.27	0.27	0.27	0.27	0.27	0.27	0.27

Table 4: Precision at varying ranks for the *acted-in* relation (R stands for Ranks).

R	*Org*	*Pln*	*Avg*	*Dst*	*Nde*	*R-Avg*	*R-Wgt*
5	0.58	0.68	**0.70**	0.68	0.64	0.66	**0.70**
10	0.60	0.57	0.59	0.58	0.59	0.6	**0.69**
15	0.57	0.53	**0.58**	0.56	0.56	0.56	**0.60**
20	0.57	0.57	0.58	0.58	0.58	0.58	**0.60**
25	**0.60**	0.54	0.56	0.57	0.56	0.57	0.57
30	0.57	0.57	0.57	0.57	0.57	0.58	**0.59**
33	0.56	0.56	0.56	0.56	0.56	0.56	0.56

Table 5: Precision at varying ranks for the *director-of* relation (R stands for Ranks).

confirm our initial observations: using traditional PageRank (***Pln***) is not desirable for the task of re-ranking facts (see Section 3.3). Our modifications to the PageRank algorithm (e.g., ***Avg***, ***Dst***, ***Nde***) consistently outperform the traditional PageRank algorithm (***Pln***). The results also underscore the benefit of combining the original extractor ranks with those generated by our graph-based ranking algorithms with ***R-Wgt*** consistently leading to highest or close to the highest average precision scores.

In Table 3, we show the average precision values for *director-of.* In this case, the summary statistic, average precision, does not show many differences between the methods. To take a finer look into the quality of these rankings, we investigated the precision scores at varying ranks across the methods. Table 4 and Table 5 show the precision at varying ranks for *acted-in* and *director-of* respectively. The maximum precision values for each rank are highlighted.

For *acted-in* again we see that ***Avg***, ***R-Avg***, ***R-Wgt*** outperform ***Org*** and ***Pln*** at *all* ranks, and ***Dst*** outperforms ***Org*** at two ranks. While the method ***Nde*** outperforms ***Org*** for a few cases, we expected it to perform better. Error analysis revealed that the sparsity of our fact graph was the problem. In our MOVIES fact graph, we observed very few nodes that are linked to *all* possible relation types, and the scores used by ***Nde*** rely on being able to identify nodes that link to numerous relation types. This problem can be alleviated by reducing the sparsity of the fact graphs (e.g., by allowing edges between nodes that are "similar enough"), which we plan to explore as future work. For *director-of,* Table 5 now shows that for small ranks (less than 15), a small (but consistent in our 10-folds) improvement is observed when comparing our random walk algorithms over ***Org***.

#Relation	*Avg*	*Dst*	*Nde*
2	0.35	0.34	0.33
3	0.35	0.35	0.34
4	0.37	0.36	0.35
5	0.38	0.38	0.37
6	0.42	0.41	0.41

Table 6: Average precision for *acted-in* for varying number of relations in the MOVIES fact farm.

While our proposed algorithms show a consistent improvement for *acted-in*, the case of *director-of* needs further discussion. For both average precision and precision vs. rank values, ***Avg***, ***R-Avg***, and ***R-Wgt*** are similar or slightly better than ***Org***. We observed that the graph-based algorithms tend to bring together "clusters" of noisy facts that may be spread out in the original ranked list of facts. To illustrate this point, we show the ten lowest scoring facts for the *director-of* relation. Table 7 shows these ten facts for ***Org*** as well as ***Avg***. These examples highlight the ability of our graph-based algorithms to demote noisy facts.

Effect of number of relations: To understand the effect of the number of relations in a farm (and hence connectivity in a fact graph), we verified the re-ranking quality of our proposed methods on various subsets of the MOVIES fact farm. We generated five different subsets, one with 2 relations, another with 3 relations, and three more with four, five, and six relations (note that although we have 5 relations in the farm, *is-movie* can be used in combination with both *acted-in* and *director-of,* thus yielding six relations to ablate.) Table 6 shows the results for *acted-in.* Overall, performance improves as we introduce more relations (i.e., more connectivity). Once again, we observe that the performance deteriorates for sparse graphs: using very few relations results in degenerating the average precision of the original ranked list. The issue of identifying the "right" characteristics of the fact graph (e.g., number of relations, degree distribution, etc.) remains future work.

Org	***Avg***
⟨david mamet, bob rafelson⟩	⟨ drama, nicholas ray⟩
⟨cinderella, wayne sleep⟩	⟨ drama, mitch teplitsky official⟩
⟨mozartdie zauberflte, julie taymor⟩	⟨ hollywood, marta bautis⟩
⟨matthew gross, julie taymor⟩	⟨ hollywood, marek stacharski⟩
⟨steel magnolias, theater project⟩	⟨ drama, kirk shannon-butts⟩
⟨rosie o'donnell, john badham⟩	⟨ drama, john pietrowski⟩
⟨my brotherkeeper, john badham⟩	⟨ drama, john madden starring⟩
⟨goldie hawn, john badham⟩	⟨ drama, jan svankmajer⟩
⟨miramaxbad santa, terry zwigoff⟩	⟨ drama, frankie sooknanan⟩
⟨premonition, alan rudolph⟩	⟨ drama, dalia hager⟩

Table 7: Sample facts for *director-of* at the bottom of the ranked list generated by (a) ***Org*** and (b) ***Avg***.

Evaluation conclusion: We demonstrated the effectiveness of our graph-based algorithms for reranking facts. In general, ***Avg*** outperforms ***Org*** and ***Pln***, and we can further improve the performance by using a combination-based ranking algorithm such as ***R-Wgt***. We also studied the impact of the size of the fact graphs on the quality of the ranked lists and showed that increasing the density of the fact farms improves the ranking using our methods.

5 Related Work

Information extraction from text has received significant attention in the recent years (Cohen and McCallum, 2003). Earlier approaches relied on hand-crafted extraction rules such as (Hearst, 1992), but recent efforts have developed supervised and semi-supervised extraction techniques (Riloff and Jones, 1999; Agichtein and Gravano, 2000; Matuszek et al., 2005; Pantel and Pennacchiotti, 2006; Paşca et al., 2006; Yan et al., 2009) as well as unsupervised techniques (Davidov and Rappoport, 2008; Mintz et al., 2009). Most common methods today use semi-supervised pattern-based learning approaches that follow (Hearst, 1992), as discussed in Section 2. Recent work has also explored extraction-related issues such as, *scalability* (Paşca et al., 2006; Ravichandran and Hovy, 2002; Pantel et al., 2004; Etzioni et al., 2004), *learning extraction schemas* (Cafarella et al., 2007a; Banko et al., 2007), and *organizing extracted facts* (Cafarella et al., 2007b). There is also a lot of work on deriving extraction scores for facts (Agichtein and Gravano, 2000; Downey et al., 2005; Etzioni et al., 2004; Pantel and Pennacchiotti, 2006).

These extraction methods are complementary to our general task of fact re-ranking. Since our proposd re-ranking algorithms are agnostic to the methods of generating the initial facts and since they do not rely on having available corpus statistics, we can use any of the available extractors in combination with any of the scoring methods. In this paper, we used Paşca et al.'s (2006) state-of-the-art extractor to learn a large set of ranked facts.

Graph-based ranking algorithms have been explored for a variety of text-centric tasks. Random walk models have been built for document summarization (Erkan and Radev, 2004), keyword extraction (Hassan et al., 2007), and collaborative filtering (Liu and Yang, 2008). Closest to our work is that of Talukdar et al. (2008) who proposed random walk algorithms for learning instances of semantic classes from unstructured and structured text. The focus of our work is on random walk models over fact graphs in order to rerank collections of facts.

6 Conclusion

In this paper, we show how information available in a farm of facts can be exploited for reranking facts. As a key contribution of the paper, we modeled fact ranking as a graph ranking problem. We proposed random walk models that determine the validity of a fact based on (a) the number of facts that "vote" for it, (b) the validity of the voting facts, and (c) the extractor's confidence in these voting facts. Our experimental results demonstrated the effectiveness of our algorithms, thus establishing a stepping stone towards exploring graph-based frameworks for fact validation. While this paper forms the basis of employing random walk models for fact re-ranking, it also suggests several interesting directions for future work. We use and build upon PageRank, however, several alternative algorithms from the link analysis literature could be adapted for ranking facts. Similarly, we employ a single (simple) graph-based representation that treats all edges the same and exploring richer graphs that distinguish between edges supporting different arguments of a fact remains future work.

References

[Agichtein and Gravano2000] Agichtein, Eugene and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *DL-00*.

[Auer et al.2008] Auer, S., C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. 2008. Dbpedia: A nucleus for a web of open data. In *ISWC+ASWC 2007*.

[Banko and Etzioni2008] Banko, Michele and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *ACL-08*.

[Banko et al.2007] Banko, Michele, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of IJCAI-07*.

[Cafarella et al.2007a] Cafarella, Michael, Dan Suciu, and Oren Etzioni. 2007a. Navigating extracted data with schema discovery. In *Proceedings of WWW-07*.

[Cafarella et al.2007b] Cafarella, Michael J., Christopher Re, Dan Suciu, Oren Etzioni, and Michele Banko. 2007b. Structured querying of web text: A technical challenge. In *Proceedings of CIDR-07*.

[Cohen and McCallum2003] Cohen, William and Andrew McCallum. 2003. Information extraction from the World Wide Web (tutorial). In *KDD*.

[Davidov and Rappoport2008] Davidov, Dmitry and Ari Rappoport. 2008. Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated sat analogy questions. In *ACL-08*.

[Downey et al.2005] Downey, Doug, Oren Etzioni, and Stephen Soderland. 2005. A probabilistic model of redundancy in information extraction. In *Proceedings of IJCAI-05*.

[Erkan and Radev2004] Erkan, Güneş and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *JAIR*, 22:457–479.

[Etzioni et al.2004] Etzioni, Oren, Michael J. Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-scale information extraction in KnowItAll. In *Proceedings of WWW-04*.

[Etzioni et al.2005] Etzioni, Oren, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: an experimental study. *Artif. Intell.*, 165:91–134.

[Hassan et al.2007] Hassan, Samer, Rada Mihalcea, and Carmen Banea. 2007. Random-walk term weighting for improved text classification. *ICSC*.

[Hearst1992] Hearst, Marti A. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING-92*.

[Kleinberg1999] Kleinberg, Jon Michael. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

[Lenat1995] Lenat, Douglas B. 1995. Cyc: a large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11).

[Liu and Yang2008] Liu, Nathan and Qiang Yang. 2008. Eigenrank: a ranking-oriented approach to collaborative filtering. In *SIGIR 2008*.

[Matuszek et al.2005] Matuszek, Cynthia, Michael Witbrock, Robert C. Kahlert, John Cabral, Dave Schneider, Purvesh Shah, and Doug Lenat. 2005. Searching for common sense: Populating cyc from the web. In *AAAI-05*.

[Mintz et al.2009] Mintz, Mike, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL-09*.

[Paşca et al.2006] Paşca, Marius, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In *Proceedings of AAAI-06*.

[Page et al.1999] Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank citation ranking: Bringing order to the Web. Technical Report 1999/66, Stanford University, Computer Science Department.

[Pantel and Pennacchiotti2006] Pantel, Patrick and Marco Pennacchiotti. 2006. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *ACL/COLING-06*.

[Pantel et al.2004] Pantel, Patrick, Deepak Ravichandran, and Eduard Hovy. 2004. Towards terascale knowledge acquisition. In *COLING-04*.

[Pantel et al.2009] Pantel, Patrick, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *EMNLP-09*.

[Ravichandran and Hovy2002] Ravichandran, Deepak and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL-08*, pages 41–47. Association for Computational Linguistics.

[Riloff and Jones1999] Riloff, Ellen and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of AAAI-99*.

[Talukdar et al.2008] Talukdar, Partha Pratim, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of EMNLP-08*.

[Yan et al.2009] Yan, Yulan, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. 2009. Unsupervised relation extraction by mining wikipedia texts with support from web corpus. In *ACL-09*.

Open Entity Extraction from Web Search Query Logs

Alpa Jain
Yahoo! Labs
alpa@yahoo-inc.com

Marco Pennacchiotti
Yahoo! Labs
pennac@yahoo-inc.com

Abstract

In this paper we propose a completely unsupervised method for open-domain entity extraction and clustering over query logs. The underlying hypothesis is that classes defined by mining search user activity may significantly differ from those typically considered over web documents, in that they better model the *user space*, i.e. users' perception and interests. We show that our method outperforms state of the art (semi-)supervised systems based either on web documents or on query logs (16% gain on the clustering task). We also report evidence that our method successfully supports a real world application, namely keyword generation for sponsored search.

1 Introduction

Search engines are increasingly moving beyond the traditional keyword-in document-out paradigm, and are improving user experience by focusing on user-oriented tasks such as query suggestions and search personalization. A fundamental building block of these applications is recognizing structured information, such as, *entities* (e.g., mentions of people, organizations, or locations) or relations among entities (Cao et al., 2008; Hu et al., 2009). For this, search engines typically rely on large collections of entities and relations built using information extraction (IE) techniques (Chaudhuri et al., 2009).

Commonly used IE techniques follow two main assumptions: (1) IE focuses on extracting information from syntactically and semantically "well-formed" pieces of texts, such as, news corpora and web documents (Pennacchiotti and Pantel, 2009); (2) extraction processes are bootstrapped with some pre-existing knowledge of the target domain (e.g entities are typically extracted for pre-defined categories, such as Actors, Manufacturers, Persons, Locations (Grishman and Sundheim, 1996)). Prior work (Banko et al., 2007), has looked into relaxing the second assumption and proposed open information extraction (OIE), a domain-independent and scalable extraction paradigm, which however focuses mostly on web corpora.

In this paper, we argue that for user-oriented applications discussed earlier, IE techniques should go beyond the traditional approach of using "well-formed" text documents. With this in mind, we explore the utility of search query logs, a rich source of user behaviors and perception, and build techniques for open entity extraction and clustering over query logs. We hypothesize that web documents and query logs model two different spaces: web documents model the *web space*, i.e. general knowledge about entities and concepts in an objective and generic way; search query logs model the *user space*, i.e. the users' view and perception of the world in a more specific fashion, where available information directly expresses users' needs and intents. For example, in a web space, 'britney spears' will tend to be similar and be clustered with other singers, such as 'celine dion' and 'bruce springsteen'. On the contrary, in the users' space, she is highly similar and clustered with other gossiped celebrities like 'paris hilton' and 'serena williams': the users' space better models the users' perception of that person; such a space is then highly valuable for all those applications where users' perceptions matters.

To computationally model our hypothesis for OIE over search query logs, we present a two phase approach to OIE for search query logs. The first phase (*entity extraction*) extracts entities from the search query logs using an unsupervised approach, by applying pattern-based heuristics and statistical measures. The second phase (*entity clustering*) induces classes over these entities by applying clustering techniques. In summary, our main contribu-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 510–518,
Beijing, August 2010

tions are: (1) We propose and instantiate a novel model for open information extraction over web search query logs; and we apply it to the task of entity extraction and clustering. (2) We show how we characterize each extracted entity to capture the 'user space', and induce classes over the entities. (3) We present an extensive evaluation over real-life datasets showing that query logs is a rich source for domain-independent user-oriented extraction tasks (Section 3). We also show the practicality of our approach by incorporating it into a real-world application, namely keyword suggestions for sponsored search (Section 4).

2 Open Entity Extraction on Query Log

In this section, we present our method for open entity extraction from query logs. We first describe our heuristic method for extracting entities (Section 2.1), and then three different feature 'user spaces' to cluster the entities (Section 2.2).

2.1 Entity Extraction

In our setting, entities correspond to Named Entities. i.e. they are defined using the standard named entity types described in (Sekine et al., 2002)[1]. In this paper, we use a set of entities extracted from query log, obtained by applying a simple algorithm (any other query log entity extraction method would apply here, e.g. (Pasca, 2007b)). The algorithm is based on the observation that oftentimes users construct their search query by copy-pasting phrases from existing texts. Due to this phenomenon, user queries often carry over surface-level properties such as capitalization and tokenization information. Our approach realizes this observation by identifying contiguous capitalized words from a user query. (In our experiments, we observed that 42% of the queries had at least one upper-case character.) Specifically, given a query $Q = q_1\ q_2\ q_3 \cdots\ q_n$, we define a candidate entity $E = e_1\ e_2\ \cdots\ e_m$ as the maximal sequence of words (i.e., alpha-numeric characters) in the query such that each word e_i in the entity begins with an uppercase character. The set of candidate entities is then cleaned by applying a set of heuristics, thus producing the final set of entities. In particular, for each extracted entity, we assign two confidence scores: a Web-based *representation score* and a query-log-based *standalone score*. The representation score checks if the case-sensitive representation observed for E in Q, is the most likely representation for E, as observed on a Web corpus (e.g., 'DOor HANGing TIps' is assigned a low representation score). The standalone score is based on the observation that a candidate E should often occur in a standalone form among the search query logs, in order to get the status of a proper named entity as defined in (Sekine et al., 2002; Grishman and Sundheim, 1996). In practice, among the query logs we must find queries of the form $Q == E$, capturing the fact that users are looking to learn more about the given entity[2].

2.2 Entity Clustering

The clustering phase takes as input any of the feature spaces presented in the rest of this section, and groups the entities according to the similarity of their vectors in the space. The desiderata for a clustering algorithm for the task of open-domain information extraction are the following: (1) The algorithm must be highly scalable, efficient, and able to handle high dimensionality, since the number of queries and the size of the feature vectors can be large; (2) We do not know in advance the number of clusters; therefore, the algorithm needs not to require a pre-defined number of clusters.

Any clustering algorithm fulfilling the above requirements would fit here. In our experiments, we adopt a highly scalable Map-Reduce implementation of the hard-clustering version of Clustering by Committee (CBC), a state-of-the-art clustering algorithm presented in (Pantel and Lin, 2002).

Context Feature Space. The basic hypothesis for the *context feature space*, is that an entity can be effectively represented by the set of contexts in which it appears in queries. This allows to capture the users' view of the entity, i.e. what people query, and want to know about the entity. This is similar to that proposed by Pasca (2007b; 2007a), i.e. that queries provide good semantics cues for modeling named entities.

Our query log feature space may significantly differ from a classical contextual feature space com-

[1] We exclude 'Time' and 'Numerical Expressions', which are out of the scope of our study.

[2] We refer the readers to (Jain and Pennacchiotti, 2010) for details on the entity extraction algorithms.

puted over a Web corpus, since the same entity can be differently perceived and described in the two corpora (query log and Web). Consider for example the entity 'galapagos islands'. Typical contexts on the Web and query log for this entity are:

web:	`endemic birds`
web:	`big turtles`
web:	`charles darwin foundation`
web:	`sensitive water`
qlog:	`trip to`
qlog:	`diving`
qlog:	`where are the`
qlog:	`travel package`

The difference between the two representations implies that entities that are similar on the Web, are not necessarily similar on query logs. For example, on the Web 'galapagos islands' is very similar to other countries such as 'tasmania', 'guinea' and 'luxemburg'; while on query log is similar to other sea-side travel destination and related concepts, such as 'greek isle', 'kauai snorkeling' and 'south america cruise'. Our new similarity computed over query log, is potentially useful for those applications in which is more important to represent users' intents, than an objective description of entities (e.g. in query suggestion and intent modeling).

To obtain our contextual representation we proceed as follows. For each entity e, we identify all queries in the query log, in which e appears. Then, we collect the set of all suffixes and postfixes of the entity in those queries. For example, given the entity 'galapagos islands' and the query 'summer 2008 galapagos islands tour', the contexts are: 'summer 2008' and 'tour'.

Once the set of all contexts of all entities has been collected, we discard contexts appearing less than τ-times in the query log, so to avoid statistical biases due to data sparseness (in the reported experiments we set $\tau = 200$). We then compute the corrected pointwise mutual information (*cpmi*) (Pantel and Ravichandran, 2004) between each instance and each context c as:

$$cpmi(e,c) = log_2 \frac{f(e,c) \cdot f(*,*)}{f(e) \cdot f(c)} \cdot M \qquad (1)$$

where $f(e,c)$ is the number of times e and c occur in the same query; $f(e)$ and $f(c)$ is the count of the entity and the context in the query log; $f(*,*)$ the overall count of all co-occurrences between contexts and entities; and M is the correction factor presented in (Pantel and Ravichandran, 2004), that eases the pmi's bias towards infrequent entities/features. Each instance is then represented in the feature space of all contexts, by the computed pmi values. Note that our method does not use any NLP parsing, since queries rarely present syntactic structure. This guarantees the method to be computationally inexpensive and easily adaptable to languages other than English.

Clickthrough Feature Space. During a search session, users issue a search query for which the search engine presents a list of result urls. Of the search results, users choose those urls that are representative of their intent. This interaction is captured by means of a click, which is logged by most search engines as *click-through data*. For instance, a search log may contain the following clicked urls for a query 'flv converter', for different users:

$user_1$:	`www.flv-converter.com`
$user_2$:	`www.videoconverterdownload.com/flv/`
$user_3$:	`www.ripzor.com/flv.html`

Our main motivation behind clustering entities based on past user click behavior is that non-identical queries that generate clicks on the same urls capture similar user intent. Thus, grouping entities that were issued as a query and generated user clicks on the same url may be considered similar. For instance, the query 'convert flv' may also generate clicks on one of the above urls, thus hinting that the two entities are similar. We observed that websites tend to dedicate a url per entity. Therefore, grouping by click urls can lead to clusters with synonyms (i.e., different ways of representing the same entity) or variants (e.g., spelling errors). To get more relevant clusters, instead of grouping entities by the click urls, we use the base urls. For instance, the url `www.ripzor.com/flv.html` is generalized to `www.ripzor.com`.

With the advent of encyclopedic websites such as, `www.wikipedia.org` and `wwww.youtube.com`, naively clustering entities by the clickthrough data can led to non-similar entities to be placed in the same cluster. For instance, we observed the most frequently clicked base url for both 'gold retriever' and 'abraham lincoln' is `www.wikipedia.org`. To address this issue, in our experiments we employed a stop-list by eliminating top-5 urls based on their inverse document frequency, where an entity is intended as the 'document'.

In practice, each extracted entity e is represented by a feature vector of size equal to the number of distinct base urls in the click-through data, across all users. Each dimension in the vector represents a url in the click-through information. The value f of an entity e for the dimension associated with url j is computed as:

$$f(e,j) = \begin{cases} \frac{w(e,j)}{\sqrt{\sum_i^{|\mathcal{U}|} w(e,i)^2}} & \text{if url } j \text{ clicked for query } e; \\ 0 & \text{otherwise.} \end{cases}$$

where $\mathcal{U}$ is the set of base urls found in click-through data when entity e was issued as a query; and $w(e,i)$ is the number of time the base url i was clicked when e was a query.

Hybrid Feature Space. We also experiment a hybrid feature space, which is composed by the normalized union of the two feature spaces above (i.e. context and clickthrough). Though more complex hybrid models could be applied, such as one based on ensemble clustering, we here opt for a simple solution which allows to better read and compare to other methods.

3 Experimental Evaluation

In this section, we report experiments on our clustering method. The goal of the experiment is two-fold: (1) evaluate the intrinsic quality of the clustering methods, i.e. if two entities in the same cluster are similar or related from a web user's perspective; (2) verify if our initial hypothesis holds, i.e. if query log based features spaces capture different properties than Web based feature spaces (i.e. the 'user space'). In Section 3.1 we describe our experimental setup; and, in 3.2 we provide the results. We couple this intrinsic evaluation with an extrinsic application-driven one in Section 4.

3.1 Experimental Settings

In the experiments we use the following datasets:

Query log: A random sample of 100 million, fully anonymized queries collected by the Yahoo! search engine in the first 3 months of 2009, along with their frequency. This dataset is used to generate both the context and the clickthrough feature spaces for the clustering step.

Web documents: A collection of 500 million web pages crawled by a Yahoo! search engine crawl. This data set is used to implement a web-based feature space that we will compare to in Section 3.2.

Entity set: A collection of 2,067,385 entities, extracted with the method described in 2.1, which shows a precision of 0.705 ± 0.044. Details on the evaluation of such method are available in (Jain and Pennacchiotti, 2010), where a full comparison with state-of-the-art systems such as (Pasca, 2007b) and (Banko et al., 2007) are also reported.

Evaluation methodology: Many clustering evaluation metrics have been proposed, ranging from Purity to Rand-statistics and F-Measure. We first select from the original 2M entity set, a random set of n entities biased by their frequency in query logs, so to keep the experiment more realistic (more frequent entities have more chances to be picked in the sample). For each entity e in the sample set, we derived a random list of k entities that are clustered with e. In our experiments, we set $n = 10$ and $k = 20$. We then present to a pool of paid editors, each entity e along with the list of co-clustered entities. Editors are requested to classify each co-clustered entity e_i as *correct* or *incorrect*. An entity e_i is deemed as correct, if it is similar or related to e from a web user's perspective: to capture this intuition, the editor is asked the question: 'If you were interested in e, would you be also interested in e_i in any intent?'.[3] Annotators' agreement over a random set of 30 entities is $kappa = 0.64$ (Marques De Sá, 2003), corresponding to substantial agreement. Additionally, we ask editors to indicate the *relation type* between e and e_i (synonyms, siblings, parent-child, topically related).

Compared methods:

CL-CTX: A CBC run, based on the query log context feature space (Section 2.2).

CL-CLK: A CBC run, based on the clickthrough feature space (Section 2.2).

[3] For example, if someone is interested in 'hasbro', he could be probably also be interested in 'lego', when the intent is buying a toy. The complete set of annotation guidelines is reported in (Jain and Pennacchiotti, 2010).

method	# cluster	avg cluster size
CL-Web	1,601	240
CL-CTX	875	1,182
CL-CLK	4,385	173
CL-HYB	1,580	478

Table 1: Statistics on the clustering results.

CL-HYB: A CBC run, based on the hybrid space that combines CL-CTXand CL-CLK(Section 2.2).
CL-Web: A state-of-the-art open domain method based on features extracted from the Web documents data set (Pantel et al., 2009). This method runs CBC over a space where features are the contexts in which an entity appears (noun chunks preceding and following the target entity); and feature value is the *pmi* between the entity and the chunks.

Evaluation metrics: We evaluate each method using **accuracy**, intended as the percentage of correct judgments.

3.2 Experimental Results

Table 3 reports accuracy results. CL-HYB is the best performing method, achieving 0.85 accuracy, respectively +4% and +11% above CL-CLK and CL-Web. CL-CTX shows the lowest performance. Our results suggest that query log spaces are more suitable to model the 'user space' wrt web features. Specifically, clickthrough information are most useful confirming our hypothesis that queries that generate clicks on the same urls capture similar user intents.

To have an anecdotal and practical intuition on the results, in Table 2 we report some entities and examples of other entities from the same clusters, as obtained from the CL-HYB and CL-Web methods. The examples show that CL-HYB builds clusters according to a variety of relations, while CL-Web mostly capture sibling-like relations.

One relevant of such relations is topicality. For example, for 'aaa insurance' the CL-HYB cluster mostly contains entities that are topically related to the American Automobile Association, while the CL-Web cluster contains generic business companies. In this case, the CL-HYB approach simply chose to group together entities having clicks to 'aaa.com' and appearing in contexts as 'auto club'. On the contrary, CL-Web grouped according to contexts such as 'selling' and 'company'. The entity 'hip osteoarthritis' shows a similar behavior: CL-HYB groups entities topically related to orthopedic issues, since most of the entities are sharing contexts such as 'treatment' and 'recovery' and, at the same time, clicks to urls such as 'orthoinfo.aaos.org' and 'arthirtis.about.com'.

entity	CL-HYB	CL-Web
aaa insurance	roadside assistance personal liability insurance international driving permits aaa minnesota travelers checks	loanmax pilot car service localnet fibermark country companies insurance
paris hilton	brenda costa adriana sklenarikova kelly clarkson anja rubik federica ridolfi	julia roberts brad pitt nicole kidman al pacino tom hanks
goldie hawn	bonnie hunt brad pitt tony curtis nicole kidman nicholas cage	julia roberts brad pitt nicole kidman al pacino tom hanks
basic algebra	numerical analysis discrete math lattice theory nonlinear physics ramsey theory	math tables trigonometry help mathtutor surface area formula multiplying fractions
hip osteoarthritis	atherosclerosis pneumonia hip fracture breast cancer anorexia nervosa	wrist arthritis disc replacement rotator cuff tears shoulder replacement american orthopedic society
acer america	acer aspire accessories aspireone acer monitors acer customer service acer usa	microsoft casio computer borland software sony nortel networks

Table 2: Sample of the generated entity clusters.

Another interesting observation regards entities referring to people. The 'paris hilton' and 'goldie hawn' examples show that the CL-Web approach groups famous people according to their category – i.e. profession in most cases. On the contrary, query log approaches tend to group people according to their social attitude, when this prevails over the profession. In the example, CL-HYB clusters the actress 'goldie hawn' with other actors, while 'paris hilton' is grouped with an heterogeneous set of celebrities that web users tend to query and click in a same manner: In this case, the social persona of 'paris hilton' prevails over its profession (actress/singer). This aspect is important in many applications, e.g. in query suggestion, where one wants to propose to the user entities that have been similarly queried and clicked.

In order to check if the above observations are not anecdotal, we studied the *relation type* annotation provided by the editors (Table 4). Table shows

method	Precision
CL-Web	0.735
CL-CTX	0.460
CL-CLK	0.815 †
CL-HYB	**0.850** †

Table 3: Precision of various clustering methods († indicates statistical-significant better than the CL-Web method, using t-test).

class	**method**			
	CL-Web	CL-CTX	CL-CLK	CL-HYB
topic	0.27	0.46	0.46	0.40
sibling	0.72	0.43	0.29	0.32
parent	-	0.09	0.13	0.09
child	0.01	-	0.01	0.02
synonym	0.01	0.03	0.12	0.16

Table 4: Fraction of entities that have been classified by editors in the different relation types.

method	**labelled clusters**			
	CL-CTX	CL-CLK	CL-HYB	CL-Web
CL-CTX	-	0.2	0.53	0.29
CL-CLK	0.21	-	0.54	0.34
CL-HYB	0.53	0.51	-	0.31
CL-Web	0.33	0.35	0.41	-

Table 5: Purity of clusters for each method using clusters from other methods as “labelled” data.

that query log based methods are more varied in the type of clusters they build. Table 5 shows the difference between the clustering obtained using the different methods and the overlap between the produced clusters. For example, 40% of the relations for the CL-HYB system are topical, while 32% are sibiling ones. On the contrary, the CL-Web method is highly biased towards sibling relations.

As regard a more attentive analysis of the different query log based methods, CL-CTX has the lowest performance. This is mainly due to the fact that contextual data are sometimes too sparse and generic. For example ‘mozilla firefox’ is clustered with ‘movie program’ and ‘astro reading’ because they share only some very generic contexts such as ‘free downloads’. In order to get more data, one option is to relax the τ threshold (see Section 2) so to include more contexts in the semantic space. Unfortunately, this would have a strong drawback, in that low-frequency context tend to be idiosyncratic and spurious. A typical case regards recurring queries submitted by robots for research purposes, such as ‘who is X’, ‘biography of X’, or ‘how to X’. These queries tend to build too generic clusters containing people or objects. Another relevant problem of the CL-CTX method is that even when using a high τ cut, clusters still tend to be too big and generic, as statistics in Table 4 shows.

CL-CTX, despite the low performance, is very useful when combined with CL-CLK. Indeed the CL-HYB system improves +4% over the CL-CLK system alone. This is because the CL-HYB method is able to recover some misleading or incomplete evidence coming from the CL-CLK using features provided by CL-CLK. For example, editors judged as incorrect 11 out of 20 entities co-clustered with the entity ‘goldie hawn’ by CL-CLK. Most of these errors are movies (e.g. ‘beverly hills cops’) soap operas (e.g. ‘sortilegio’) and directors, because all have clicks to ‘imdb.com’ and ‘movies.yahoo.com’. CL-HYB recovers these errors by including features coming from CL-CTX such as ‘actress’.

In summary, query log spaces group together entities that are similar by web users (this being topical similarity or social attitude), thus constituting a practical model of the ‘user space’ to be leveraged by web applications.

4 Keywords for Sponsored Search

In this section we explore the use of our methods for keyword generation for sponsored search. In sponsored search, a search company opens an auction, where on-line advertisers bid on specific keywords (called *bidterms*). The winner is allowed to put its ad and link on the search result page of the search company, when the bidterm is queried. Companies such as Google and Yahoo are investing efforts for improving their bidding platforms, so to attract more advertisers in the auctions. Bidterm suggestion tools (adWords, 2009; yahooTool, 2009) are used to help advertiser in selecting bidterms: the advertisers enters a seed keyword (*seed*) expressing the intent of its ad, and the tool returns a list of suggested keywords (*suggestions*) that it can use for bidding – e.g for the seed ‘mp3 player’, a suggestion could be ‘ipod nano’. The task of generating bid suggestions (i.e. *keyword generation*) is typically automatic, and has received a growing attention in the search community for its impact on search company revenue. The main problem of existing methods for suggestion (adWords, 2009; yahooTool, 2009; wordTracker, 2009) is that they produce only suggestions that contain the initial seed (e.g. ‘belkin mp3 player’ for the seed ‘mp3 player’), while nonobvious (and potentially less expensive) suggestions not containing the seed are neglected (e.g. ‘ipod nano’ for ‘mp3 player’). For example for ‘galapagos islands’, a typical production system suggests ‘galapagos islands tour’ which cost almost 5$ per click; while the less obvious ‘isla santa cruz’ would cost only 0.35$. Below we show our method to discover such nonobvious suggestions, by retrieving entities in the same cluster of a given seed.

4.1 Experimental Setting

We evaluate the quality of the suggestions proposed by different methods for a set of seed bidterms., adopting the evaluation schema in (Joshi and Motwani, 2006)

Dataset Creation. To create the set of seeds, we use Google skTool[4]. The tool provides a list of popular bid terms, organized in a taxonomy of advertisement topics. We select 3 common topics: tourism, vehicles and consumer-electronics. For each topic, we randomly pick 5 seeds among the 800 most popular bid terms, which also appear in our entity set described in Section 3.1.[5]. We evaluate a system by collecting all its suggestions for the 15 seeds, and then extracting a random sample of 20 suggestions per seed.

Evaluation and Metrics. We use precision and Nonobviousness. **Precision** is computed by asking two experienced human experts to classify each suggestion of a given seed, as *relevant* or *irrelevant*. A suggestion is deemed as relevant if any advertiser would likely choose to bid for the suggestion, having as intent the seed. Annotator agreement, evaluated on a subset of 120 suggestions is $kappa = 0.72$ (substantial agreement). Precision is computed as the percentage of suggestions judged as relevant. **Nonobviousness** is a metric introduced in (Joshi and Motwani, 2006), capturing how nonobvious the suggestions are. It simply counts how many suggestions for a given seed do not contain the seed itself (or any of its variants): this metric is computed automatically using string matching and a simple stemmer.

Comparisons. We compare the suggestions proposed by `CL-CTX`, `CL-CLK`, and `CL-HYB`, against `Web` and two reference state-of-the-art production systems: Google AdWords (`GOO`) and Yahoo Search Marketing Tool (`YAH`). As concerns our methods, we extract as suggestions the entities that occur in the same cluster of a given seed. For the production systems, we rely on the suggestions proposed on the website of the tools.

4.2 Experimental Results

Precision results are reported in the second column of Table 6. Both `CL-CLK` and `CL-HYB` outperform `Web` in precision, `CL-HYB` being close to the upper-bound of the two production systems. As expected, production systems show a very high precision but their suggestions are very obvious. Our results are fairly in line with those obtained on a similar dataset, by Joshi and Motwani (2006).

A closer look at the results shows that most of the errors for `CL-CTX` are caused by the same problem outlined in Section 3.2: Some entities are wrongly assigned to a cluster, because they have some high *cpmi* context feature which is shared with the cluster centroid, but which is not very characteristic for the entity itself. This is particularly evident for some of the low frequency entities, where *cpmi* values could not reflect the actual semantics of the entity. For example the entity ‘nickelodeon’ (a kids tv channel in UK) is assigned to the cluster of ‘galapagos islands’, because of the feature ‘cruise’: indeed, some people query about ‘nickelodeon cruise’ because the tv channel organizes some kids cruises. Other mistakes are due to feature ambiguity. For example, the entity ’centurion boats’ is assigned to the cluster of ‘obertauern’ (a ski resort in Austria), because they share the ambiguous feature ‘ski’ (meaning either winter-ski or water-ski). As for the `CL-CLK` system, some of the errors are caused by the fact that some base url can refer to very different types of entities. For example the entity ‘color copier’ is suggested for the the camera ‘canon rebel xti’, since they both share clicks to the Canon website. The `CL-HYB` system achieves a higher preci-

[4] http://www.google.com/sktool

[5] The final set of 15 bid terms is: *tourism:*galapagos islands,holiday insurance,hotel booking,obertauern,wagrain; *vehicles:*audi q7,bmw z4,bmw dealers,suzuki grand vitara,yamaha banshee; *consumer electr:*canon rebel xti,divx converter,gtalk,pdf reader,flv converter.

method	Precision	Nonobviousness
GOO	0.982	0.174
YAH	0.966	0.195
Web	0.814	0.827
CL-CTX	0.547	0.963
CL-CLK	0.827	0.630
CL-HYB	0.946	0.567

Table 6: Results for keyword generation.

sion wrt CL-CTX and CL-CLK: the combination of the two spaces decreases the impact of misleading features –e.g. for 'yamaha bunshee', all CL-HYB 's suggestions are correct, while almost all CL-CLK 's suggestions are incorrect: the hybrid system recovered the negative effect of the misleading feature ebay.com, by backing up on features from the contextual subspace (e.g. 'custom', 'specs', 'used parts').

Nonobviousness results are reported in column three of Table 6. All our systems return a high number of nonobvious suggestions (all above 50%).[6] On the contrary, GOO and YAH show low performance, as both systems are heavily based on the substring matching technique. This strongly motivates the use of semantic approaches as those we propose, that guarantee at the same time both a higher linguistic variety and an equally high precision wrt the production systems. For example, for the seeds 'galapagos islands', GOO returns simple suggestions such as 'galapagos islands vacations' and 'galapagos islands map'; while CL-HYB returns 'caribbean mexico' and 'pacific dawn', two terms that are semantically related but dissimilar from the seed. Remember that these letter terms are related to the seed because they are similar in the user space, i.e. users looking at 'galapagos islands' tend to similarly look for 'caribbean mexico' and 'pacific dawn'. These suggestions would then be very valuable for tourism advertisers willing to improve their visibility through a non-trivial and possibly less expensive set of bid terms.

5 Related Work

While literature abounds with works on entity extraction from web documents (e.g. (Banko et al., 2007; Chaudhuri et al., 2009; Pennacchiotti and Pantel, 2009)), the extraction of classes of entities over query logs is a pretty new task, recently introduced in (Pasca, 2007b). Pasca's system extracts entities of pre-defined classes in a semi-supervised fashion, starting with an input class represented by a set of seeds, which are used to induce typical query-contexts for the class. Contexts are then used to extract and select new candidate instances for the class. A similar approach is also adopted in (Sekine and Suzuki, 2007). Pasca shows an improvement of about 20% accuracy, compared to existing Web-based systems. Our extraction algorithm differs from Pasca's work in that it is completely unsupervised. Also, Pasca's cannot be applied to OIE, i.e. it only works for pre-defined classes. Our clustering approach is related to Lin and Wu's work (Lin and Wu, 2009). Authors propose a semi-supervised algorithm for query classification. First, they extract a large set of 20M phrases from a query log, as those unique queries appearing more than 100 times in a Web corpus. Then, they cluster the phrases using the K-means algorithm, where features are the phrases' bag-of-words contexts computed over a web corpus. Finally, they classify queries using a logistic regression algorithm. Our work differs from Lin and Wu, as we focus on entities instead of phrases. Also, the features we use for clustering are from query logs and click data, not web contexts.

6 Conclusions

We presented an open entity extraction approach over query logs that goes beyond the traditional web corpus, with the goal of modeling a 'user-space' as opposed to an established 'web-space'. We showed that the clusters generated by query logs substantially differ from those by a Web corpus; and that our method is able to induce state-of-the-art quality classes on a user-oriented evaluation on the real world task of keyword generation for sponsored search. As future work we plan to: (i) experiment different clustering algorithms and feature models, e.g. soft-clustering for handling ambiguous entities; (ii) integrate the Web space and the query log spaces; (iii) embed our methods in in existing tools for intent modeling, query suggestion and similia, to check its impact in production systems.

[6] Note that very high values for CL-CTX may be misleading, as many of the suggestions proposed by this system are incorrect (see precision results) and hence non-obvious (e.g., 'derek lewis' for 'galapagos islands').

References

adWords. 2009. Google adwords. adwords.google.com/select/keywordtoolexternal.

Banko, Michele, Michael Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of IJCAI*.

Cao, Huanhuan, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware query suggestion by mining click-through and session data. In *Proceedings of KDD-08*.

Chaudhuri, Surajit, Venkatesh Ganti, and Dong Xin. 2009. Exploiting web search to generate synonyms for entities. In *Proceedings of WWW-09*.

Grishman, R. and B. Sundheim. 1996. Message understanding conference- 6: A brief history. In *Proceedings of COLING*.

Hu, Jian, Gang Wang, Fred Lochovsky, Jian tao Sun, and Zheng Chen. 2009. Understanding user's query intent with Wikipedia. In *Proceedings of WWW-09*.

Jain, Alpa and Marco Pennacchiotti. 2010. Open Information Extraction from Web Search Query Logs. Technical Report YL-2010-003, Yahoo! Labs.

Joshi, Amruta and Rajeev Motwani. 2006. Keyword generation for search engine advertising. In *Proceedings of Sixth IEEE-ICDM*.

Lin, Dekang and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of ACL-IJCNLP-2009*.

Marques De Sá, Joaquim P. 2003. *Applied Statistics*. Springer Verlag.

Pantel, Patrick and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings KDD-02*.

Pantel, Patrick and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *Proceeding of HLT-NAACL-2004*.

Pantel, Patrick, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of EMNLP-09*.

Pasca, Marius. 2007a. Organizing and searching the world wide web of facts - step two: Harnessing the wisdom of the crowds. In *Proceedings of the WWW-2007*.

Pasca, Marius. 2007b. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of CIKM-2007*.

Pennacchiotti, Marco and Patrick Pantel. 2009. Entity extraction via ensemble semantics. In *Proceedings of EMNLP-2009*.

Sekine, Satoshi and Hisami Suzuki. 2007. Acquiring ontological knowledge from query logs. In *Proceedings of WWW-07*.

Sekine, Satoshi, Kiyoshi Sudo, and Chikashi Nobata. 2002. Extended named entity hierarchy. In *Proceedings of LREC-2002*.

wordTracker. 2009. Word tracker. www.wordtracker.com.

yahooTool. 2009. Yahoo search marketing. searchmarketing.yahoo.com.

Reranking Models in Fine-grained Opinion Analysis

Richard Johansson and **Alessandro Moschitti**
University of Trento
{johansson, moschitti}@disi.unitn.it

Abstract

We describe the implementation of reranking models for fine-grained opinion analysis – marking up opinion expressions and extracting opinion holders. The reranking approach makes it possible to model complex relations between multiple opinions in a sentence, allowing us to represent how opinions *interact* through the syntactic and semantic structure. We carried out evaluations on the MPQA corpus, and the experiments showed significant improvements over a conventional system that only uses local information: for both tasks, our system saw recall boosts of over 10 points.

1 Introduction

Recent years have seen a surge of interest in the automatic processing of *subjective language*. The technologies emerging from this research have obvious practical uses, either as stand-alone applications or supporting other NLP tools such as information retrieval or question answering systems. While early efforts in subjectivity analysis focused on coarse-grained tasks such as retrieving the subjective documents from a collection, most recent work on this topic has focused on fine-grained tasks such as determining the attitude of a particular person on a particular topic. The development and evaluation of such systems has been made possible by the release of manually annotated resources using fairly fine-grained representations to describe the structure of subjectivity in language, for instance the MPQA corpus (Wiebe et al., 2005).

A central task in the automatic analysis of subjective language is the indentification of *subjective expressions*: the text pieces that allow us to draw the conclusion that someone has a particular feeling about something. This is necessary for further analysis, such as the determination of opinion holder and the polarity of the opinion. The MPQA corpus defines two types of subjective expressions: *direct subjective expressions* (DSEs), which are explicit mentions of attitude, and *expressive subjective elements* (ESEs), which signal the attitude of the speaker by the choice of words. The prototypical example of a DSE would be a verb of statement or categorization such as *praise* or *disgust*, and the opinion holder would typically be a direct semantic argument of this verb. ESEs, on the other hand, are less easy to categorize syntactically; prototypical examples would include value-expressing adjectives such as *beautiful* and strongly charged words like *appeasement*, while the relation between the expression and the opinion holder is typically less clear-cut than for DSEs. In addition to DSEs and ESEs, the MPQA corpus also contains annotation for non-subjective statements, which are referred to as *objective speech events* (OSEs).

Examples (1) and (2) show two sentences from the MPQA corpus where DSEs and ESEs have been manually annotated.

(1) He [made such charges]$_{DSE}$ [despite the fact]$_{ESE}$ that women's political, social and cultural participation is [not less than that]$_{ESE}$ of men.

(2) [However]$_{ESE}$, it is becoming [rather fashionable]$_{ESE}$ to [exchange harsh words]$_{DSE}$ with each other [like kids]$_{ESE}$.

The task of marking up these expressions has usually been approached using straightforward sequence labeling techniques using using simple features in a small contextual window (Choi et al., 2006; Breck et al., 2007). However, due to

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 519–527,
Beijing, August 2010

the simplicity of the feature sets, this approach fails to take into account the fact that the semantic and pragmatic interpretation of sentences is not only determined by words but also by syntactic and shallow-semantic *relations*. Crucially, taking grammatical relations into account allows us to model how expressions *interact* in various ways that influence their interpretation as subjective or not. Consider, for instance, the word *said* in examples (3) and (4) below, where the interpretation as a DSE or an OSE is influenced by the subjective content of the enclosed statement.

(3) "We will identify the [culprits]$_{ESE}$ of these clashes and [punish]$_{ESE}$ them," he [said]$_{DSE}$.

(4) On Monday, 80 Libyan soldiers disembarked from an Antonov transport plane carrying military equipment, an African diplomat [said]$_{OSE}$.

In addition, the various opinions expressed in a sentence are very interdependent when it comes to the resolution of their *holders*, i.e. determining the entity that harbors the sentiment manifested textually in the opinion expression. Clearly, the structure of the sentence is influential also for this task: an ESE will be quite likely to be linked to the same opinion holder as a DSE directly above it in the syntactic tree.

In this paper, we demonstrate how syntactic and semantic structural information can be used to improve the detection of opinion expressions and the extraction of opinion holders. While this feature model makes it impossible to use the standard sequence labeling method, we show that with a simple strategy based on *reranking*, incorporating structural features results in a significant improvement. In an evaluation on the MPQA corpus, the best system we evaluated, a reranker using the Passive–Aggressive learning algorithm, achieved a 10-point absolute improvement in soft recall, and a 5-point improvement in F-measure, over the baseline sequence labeler. Similarly, the recall is boosted by almost 11 points for the holder extraction (3 points in F-measure) by modeling the interaction of opinion expressions with respect to holders.

2 Related Work

Since the most significant body of work in subjectivity analysis has been dedicated to coarse-grained tasks such as document polarity classification, most approaches to analysing the sentiment of natural-language text have relied fundamentally on purely lexical information (see (Pang et al., 2002; Yu and Hatzivassiloglou, 2003), *inter alia*) or low-level grammatical information such as part-of-speech tags and functional words (Wiebe et al., 1999). This is not unexpected since these problems have typically been formulated as text categorization problems, and it has long been agreed in the information retrieval community that very little can be gained by complex linguistic processing for tasks such as text categorization and search (Moschitti and Basili, 2004).

As the field moves towards increasingly sophisticated tasks requiring a detailed analysis of the text, the benefit of syntactic and semantic analysis becomes more clear. For the task of subjective expression detection, Choi et al. (2006) and Breck et al. (2007) used syntactic features in a sequence model. In addition, syntactic and shallow-semantic relations have repeatedly proven useful for subtasks of subjectivity analysis that are inherently *relational*, above all for determining the holder or topic of a given opinion. Choi et al. (2006) is notable for the use of a global model based on hand-crafted constraints and an integer linear programming optimization step to ensure a globally consistent set of opinions and holders.

Works using syntactic features to extract topics and holders of opinions are numerous (Bethard et al., 2005; Kobayashi et al., 2007; Joshi and Penstein-Rosé, 2009; Wu et al., 2009). Semantic role analysis has also proven useful: Kim and Hovy (2006) used a FrameNet-based semantic role labeler to determine holder and topic of opinions. Similarly, Choi et al. (2006) successfully used a PropBank-based semantic role labeler for opinion holder extraction. Ruppenhofer et al. (2008) argued that semantic role techniques are useful but not completely sufficient for holder and topic identification, and that other linguistic phenomena must be studied as well. One such linguistic pheonomenon is the *discourse* structure, which has recently attracted some attention in the subjectivity analysis community (Somasundaran et al., 2009).

3 Modeling Interaction over Syntactic and Semantic Structure

Previous systems for opinion expression markup have typically used simple feature sets which have allowed the use of efficient off-the-shelf sequence labeling methods based on Viterbi search (Choi et al., 2006; Breck et al., 2007). This is not possible in our case since we would like to extract structural, relational features that involve *pairs* of opinion expressions and may apply over an arbitrarily long distance in the sentence.

While it is possible that search algorithms for exact or approximate inference can be constructured for the $\arg\max$ problem in this model, we sidestepped this issue by using a *reranking* decomposition of the problem:

- Apply a standard Viterbi-based sequence labeler based on local context features but no structural interaction features. Generate a small candidate set of size k.
- Generate opinion holders for every proposed opinion expression.
- Apply a complex model using interaction features to pick the top candidate from the candidate set.

The advantages of a reranking approach compared to more complex approaches requiring advanced search techniques are mainly simplicity and efficiency: this approach is conceptually simple and fairly easy to implement provided that k-best output can be generated efficiently, and features can be arbitrarily complex – we don't have to think about how the features affect the algorithmic complexity of the inference step. A common objection to reranking is that the candidate set may not be diverse enough to allow for much improvement unless it is very large; the candidates may be trivial variations that are all very similar to the top-scoring candidate.

3.1 Syntactic and Semantic Structures

We used the syntactic–semantic parser by Johansson and Nugues (2008) to annnotate the sentences with dependency syntax (Mel'čuk, 1988) and shallow semantic structures in the PropBank (Palmer et al., 2005) and NomBank (Meyers et al., 2004) frameworks. Figure 1 shows an example of the annotation: The sentence *they called him a liar*, where *called* is a DSE and *liar* is an ESE, has been annotated with dependency syntax (above the text) and PropBank-based semantic role structure (below the text). The predicate *called*, which is an instance of the PropBank frame call.01, has three semantic arguments: the Agent (A0), the Theme (A1), and the Predicate (A2), which are realized on the surface-syntactic level as a subject, a direct object, and an object predicative complement, respectively.

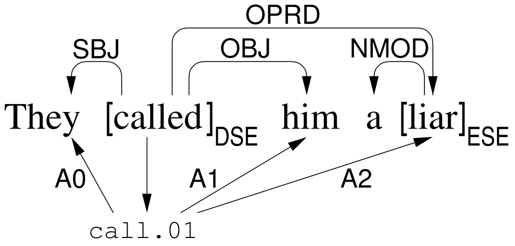


Figure 1: Syntactic and shallow semantic structure.

3.2 Base Sequence Labeling Model

To solve the first subtask, we implemented a standard sequence labeler for subjective expression markup, similar to the approach by Breck et al. (2007). We encoded the opinionated expression brackets using the IOB2 encoding scheme (Tjong Kim Sang and Veenstra, 1999) and trained the model using the metod by Collins (2002).

The sequence labeler used word, POS tag, and lemma features in a window of size 3. In addition, we used prior polarity and intensity features derived from the lexicon created by Wilson et al. (2005). It is important to note that prior subjectivity does not always imply subjectivity in a particular context; this is why contextual features are essential for this task.

This sequence labeler was used to generate the candidate set for the reranker. To generate reranking training data, we carried out a 5-fold hold-out procedure: We split the training set into 5 pieces, trained a sequence labeler on pieces 1 to 4, applied it to piece 5 and so on.

3.3 Base Opinion Holder Extractor

For every opinion expression, we extracted *opinion holders*, i.e. mentions of the entity holding the opinion denoted by the opinion expression. Since the problem of holder extraction is in many ways similar to semantic argument detection – when the opinion expression is a verb, finding the holder typically entails finding a SPEAKER argument – we approached this problem using methods inspired by semantic role labeling. We thus trained support vector machines using the LIBLINEAR software (Fan et al., 2008), and applied them to the noun phrases in the same sentence as the holder. Separate classifiers were trained to extract holders for DSEs, ESEs, and OSEs. The classifiers used the following feature set:

- **SYNTACTIC PATH**. Similarly to the path feature widely used in SRL, we extract a feature representing the path in the dependency tree between the expression and the holder (Johansson and Nugues, 2008). For instance, the path from the DSE *called* to the holder *They* is `SBJ↓`.
- **SHALLOW-SEMANTIC RELATION**. If there is a direct shallow-semantic relation between the expression and the holder, use a feature representing its semantic role, such as `A0` for *They* with respect to *called*.
- **EXPRESSION HEAD WORD AND POS**.
- **HOLDER HEAD WORD AND POS**.
- **DOMINATING EXPRESSION TYPE**.
- **CONTEXT WORDS AND POS FOR HOLDER**.
- **EXPRESSION VERB VOICE**.

However, there are also differences compared to typical argument extraction in SRL. First, it is important to note that the MPQA corpus does not annotate direct links from opinions to a holders, but from opinions to *holder coreference chains*. To handle this issue, we created positive training instances for *all* members of the coreference chain in the same sentence as the opinion, and negative instances for the other noun phrases.

Secondly, an opinion may be linked not to an overt noun phrase in a sentence, but to an *implicit* holder; a special case of implicit holder is the *writer* of the text. We trained separate classifiers to detect these situations. These classifiers did not use the features requiring a holder phrase.

Finally, there is a restriction that every expression may have at most one holder, so at test time we select only the highest-scoring opinion holder candidate.

3.4 Opinion Expression Reranker Features

The rerankers use two types of structural features: syntactic features extracted from the dependency tree, and semantic features extracted from the predicate–argument (semantic role) graph.

The syntactic features are based on paths through the dependency tree. This creates a small complication for multiword opinion expressions; we select the shortest possible path in such cases. For instance, in example (1) above, the path will be computed between *made* and *despite*, and in (2) between *fashionable* and *exchange*.

We used the following syntactic interaction features:

- **SYNTACTIC PATH**. Given a pair opinion expressions, we use a feature representing the labels of the two expressions and the path between them through the syntactic tree. For instance, for the DSE *called* and the ESE *liar* in Figure 1, we represent the syntactic configuration using the feature `DSE:OPRD↓:ESE`, meaning that the path from the DSE to the ESE follows an OPRD link downward.
- **LEXICALIZED PATH**. Same as above, but with lexical information attached: `DSE/called:OPRD↓:ESE/liar`.
- **DOMINANCE**. In addition to the features based on syntactic paths, we created a more generic feature template describing dominance relations between expressions. For instance, from the graph in Figure 1, we extract the feature `DSE/called→ESE/liar`, meaning that a DSE with the word *called* dominates an ESE with the word *liar*.

The semantic features were the following:

Predicate Sense Label. For every predicate found inside an opinion expression, we add a feature consisting of the expression label and the predicate sense identifier. For instance, the verb *call* which is also a DSE is represented with the feature `DSE/call.01`.

Predicate and Argument Label. For every argument of a predicate inside an opinion expression, we also create a feature representing the predicate–argument pair: `DSE/call.01:A0`.

Connecting Argument Label. When a predicate inside some opinion expression is connected to some argument inside another opinion expression, we use a feature consisting of the two expression labels and the argument label. For instance, the ESE *liar* is connected to the DSE *call* via an A2 label, and we represent this using a feature `DSE:A2:ESE`.

Apart from the syntactic and semantic features, we also used the score output from the base sequence labeler as a feature. We normalized the scores over the k candidates so that their exponentials summed to 1.

3.5 Opinion Holder Reranker Features

In addition, we modeled the interaction between different opinions with respect to their holders. We used the following two features to represent this interaction:

Shared Holders. A feature representing whether or not two opinion expressions have the same holder. For instance, if a DSE dominates an ESE and they have the same holder as in Figure 1 where the holder is *They*, we represent this by the feature `DSE:ESE:true`.

Holder types + Path. A feature representing the types of the holders, combined with the syntactic path between the expressions. The types take the following possible values: explicit, implicit, writer. In Figure 1, we would thus extract the feature `DSE/Expl:OPRD↓:ESE/Expl`.

Similar to base model feature for the expression detection, we also used a feature for the output score from the holder extraction classifier.

3.6 Training the Reranker

We trained the reranker using the method employed by many rerankers following Collins (2002), which learns a scoring function that is trained to maximize performance on the reranking task. While there are batch learning algorithms that work in this setting (Tsochantaridis et al., 2005), online learning methods have been more popular for performance reasons. We investigated two online learning algorithms: the popular *structured perceptron* (Collins, 2002) and the Passive–Aggressive (PA) algorithm (Crammer et al., 2006). To increase robustness, we used an averaged implementation (Freund and Schapire, 1999) of both algorithms.

The difference between the two algorithms is the way the weight vector is incremented in each step. In the perceptron, for a given input x, we update based on the difference between the correct output y and the predicted output $\hat{y}$, where Φ is the feature representation function:

$$\begin{aligned}&\hat{y} \leftarrow \arg\max_h w \cdot \Phi(x,h)\\ &w \leftarrow w + \Phi(x,y) - \Phi(x,\hat{y})\end{aligned}$$

In the PA algorithm, which is based on the theory of large-margin learning, we instead find the $\hat{y}$ that violates the margin constraints maximally. The update step length τ is computed based on the margin; this update is bounded by a regularization constant C:

$$\begin{aligned}&\hat{y} \leftarrow \arg\max_h w \cdot \Phi(x,h) + \sqrt{\rho(y,h)}\\ &\tau \leftarrow \min\left(C, \frac{w(\Phi(x,\hat{y})-\Phi(x,y))+\sqrt{\rho(y,\hat{y})}}{\|\Phi(x,\hat{y})-\Phi(x,y)\|^2}\right)\\ &w \leftarrow w + \tau(\Phi(x,y) - \Phi(x,\hat{y}))\end{aligned}$$

The algorithm uses a cost function ρ. We used the function $\rho(y,\hat{y}) = 1 - F(y,\hat{y})$, where F is the soft F-measure described in Section 4.1. With this approach, the learning algorithm thus directly optimizes the measure we are interested in, i.e. the F-measure.

4 Experiments

We carried out the experiments on version 2 of the MPQA corpus (Wiebe et al., 2005), which we split into a test set (150 documents, 3,743 sentences) and a training set (541 documents, 12,010 sentences).

4.1 Evaluation Metrics

Since expression boundaries are hard to define exactly in annotation guidelines (Wiebe et al., 2005), we used soft precision and recall measures to score the quality of the system output. To derive the soft precision and recall, we first define the *span coverage* c of a span s with respect to another span s', which measures h ow well s' is covered by s:

$$c(s, s') = \frac{|s \cap s'|}{|s'|}$$

In this formula, the operator $|\cdot|$ counts tokens, and the intersection $\cap$ gives the set of tokens tha t two spans have in common. Since our evaluation takes span labels (DSE, ESE, OSE) into account, we set $c(s, s')$ to zero if the labels associated with s and s' are different.

Using the span coverage, we define the *span set coverage* C of a set of spans $\boldsymbol{S}$ with respect to a set $\boldsymbol{S'}$:

$$C(\boldsymbol{S}, \boldsymbol{S'}) = \sum_{s_j \in \boldsymbol{S}} \sum_{s'_k \in \boldsymbol{S'}} c(s_j, s'_k)$$

We now define the soft precision P and recall R of a proposed set of spans $\hat{\boldsymbol{S}}$ with respect to a gold standard set $\boldsymbol{S}$ as follows:

$$P(\boldsymbol{S}, \hat{\boldsymbol{S}}) = \frac{C(\boldsymbol{S}, \hat{\boldsymbol{S}})}{|\hat{\boldsymbol{S}}|} \quad R(\boldsymbol{S}, \hat{\boldsymbol{S}}) = \frac{C(\hat{\boldsymbol{S}}, \boldsymbol{S})}{|\boldsymbol{S}|}$$

Note that the operator $|\cdot|$ counts spans in this formula.

Conventionally, when measuring the quality of a system for an information extraction task, a predicted entity is counted as correct if it exactly matches the boundaries of a corresponding entity in the gold standard; there is thus no reward for close matches. However, since the boundaries of the spans annotated in the MPQA corpus are not strictly defined in the annotation guidelines (Wiebe et al., 2005), measuring precision and recall using exact boundary scoring will result in figures that are too low to be indicative of the usefulness of the system. Therefore, most work using this corpus instead use overlap-based precision and recall measures, where a span is counted as correctly detected if it *overlaps* with a span in the gold standard (Choi et al., 2006; Breck et al., 2007). As pointed out by Breck et al. (2007), this is problematic since it will tend to reward long spans – for instance, a span covering the whole sentence will always be counted as correct if the gold standard contains any span for that sentence.

The precision and recall measures proposed here correct the problem with overlap-based measures: If the system proposes a span covering the whole sentence, the span coverage will be low and result in a low soft precision. Note that our measures are bounded below by the exact measures and above by the overlap-based measures: replacing $c(s, s')$ with $\lfloor c(s, s') \rfloor$ gives the exact measures and replacing $c(s, s')$ with $\lceil c(s, s') \rceil$ the overlap-based measures.

To score the extraction of opinion holders, we started from the same basic approach. However, the evaluation of this task is more complex because a) we only want to give credit for holders for correctly extracted opinion expressions; b) the gold standard links opinion expressions to coreference chains rather than individual mentions of holders; c) the holder may be the writer or implicit (see 3.3). We therefore used the following method: Given a holder h linked to an expression e, we first located the expression e' in the gold standard that most closely corresponds to e, that is $e' = \arg\max_x c(x, e)$, regardless of the labels of e and e'. We then located the gold standard holder h' by finding the closest corresponding holder in the coreference chain H linked to e': $h' = \arg\max_{x \in H} c(x, h)$. If h is proposed as the writer, we score it as perfectly detected (coverage 1) if the coreference chain H contains the writer, and a full error (coverage 0) otherwise, and similar if h is implicit.

4.2 Machine Learning Methods

We compared the machine learning methods described in Section 3. In these experiments, we used a candidate set size k of 8. Table 1 shows the results of the evaluations using the precision and recall measures described above. The baseline is the result of taking the top-scoring labeling from the base model.

System	P	R	F
Baseline	63.36	46.77	53.82
Perceptron	62.84	48.13	54.51
PA	63.50	51.79	57.04

Table 1: Evaluation of reranking learning methods.

We note that the best performance was obtained using the PA algorithm. While these results are satisfactory, it is possible that they could be improved further if we would use a batch learning method such as SVM$^{\text{struct}}$ (Tsochantaridis et al., 2005) instead of the online learning methods used here.

4.3 Candidate Set Size

In any method based on reranking, it is important to study the influence of the candidate set size on the quality of the reranked output. In addition, an interesting question is what the upper bound on reranker performance is – the *oracle* performance. Table 2 shows the result of an experiment that investigates these questions. We used the reranker based on the Passive–Aggressive method in this experiment since this reranker gave the best results in the previous experiment.

	Reranked			Oracle		
k	P	R	F	P	R	F
1	63.36	46.77	53.82	63.36	46.77	53.82
2	63.70	48.17	54.86	72.66	55.18	62.72
4	63.57	49.78	55.84	79.12	62.24	69.68
8	63.50	51.79	57.04	83.72	68.14	75.13
16	63.00	52.94	57.54	86.92	72.79	79.23
32	62.15	54.50	58.07	89.18	76.76	82.51
64	61.02	55.67	58.22	91.08	80.19	85.28
128	60.22	56.45	58.27	92.63	83.00	87.55
256	59.87	57.22	58.51	94.01	85.27	89.43

Table 2: Oracle and reranker performance as a function of candidate set size.

As is common in reranking tasks, the reranker can exploit only a fraction of the potential improvement – the reduction of the F-measure error is between 10 and 15 percent of the oracle error reduction for all candidate set sizes.

The most visible effect of the reranker is that the recall is greatly improved. However, this does not seem to have an adverse effect on the precision until the candidate set size goes above 16 – in fact, the precision actually improves over the baseline for small candidate set sizes. After the size goes above 16, the recall (and the F-measure) still rises, but at the cost of decreased precision.

4.4 Syntactic and Semantic Features

We studied the impact of syntactic and semantic structural features on the performance of the reranker. Table 3 shows the result of the investigation for syntactic features. Using all the syntactic features (and no semantic features) gives an F-measure roughly 4 points above the baseline, using the PA reranker with a k of 64. We then measured the F-measure obtained when each one of the three syntactic features has been removed. It is clear that the unlexicalized syntactic path is the most important syntactic feature; the effect of the two lexicalized features seems to be negligible.

System	P	R	F
Baseline	63.36	46.77	53.82
All syntactic	62.45	53.19	57.45
No SYN PATH	64.40	48.69	55.46
No LEX PATH	62.62	53.19	57.52
No DOMINANCE	62.32	52.92	57.24

Table 3: Effect of syntactic features.

A similar result was obtained when studying the semantic features (Table 4). Removing the connecting labels feature, which is unlexicalized, has a greater effect than removing the other two semantic features, which are lexicalized.

System	P	R	F
Baseline	63.36	46.77	53.82
All semantic	61.26	53.85	57.31
No PREDICATE SL	61.28	53.81	57.30
No PRED+ARGLBL	60.96	53.61	57.05
No CONN ARGLBL	60.73	50.47	55.12

Table 4: Effect of semantic features.

4.5 Opinion Holder Extraction

Table 5 shows the performance of the opinion holder extractor. The baseline applies the holder classifier (3.3) to the opinions extracted by the base sequence labeler (3.2), without modeling any interactions between opinions. A large performance boost is then achieved simply by applying the opinion expression reranker ($k = 64$); this is simply the consequence of improved expression detection, since a correct expression is required to get credit for a holder).

However, we can improve on this by adding the holder interaction features: both the SHARED HOLDERS and HOLDER TYPES + PATH features contribute to improving the recall even further.

System	P	R	F
Baseline	57.66	45.14	50.64
Reranked expressions	52.35	52.54	52.45
SHARED HOLDERS	52.43	55.21	53.78
HTYPES + PATH	52.22	54.41	53.30
Both	52.28	55.99	54.07

Table 5: Opinion holder extraction experiments.

5 Conclusion

We have shown that features derived from grammatical and semantic role structure can be used to improve two fundamental tasks in fine-grained opinion analysis: the detection of opinionated expressions in subjectivity analysis, and the extraction of opinion holders. Our feature sets are based on interaction between opinions, which makes exact inference intractable. To overcome this issue, we used an implementation based on reranking: we first generated opinion expression sequence candidates using a simple sequence labeler similar to the approach by Breck et al. (2007). We then applied SRL-inspired opinion holder extraction classifiers, and finally a global model applying to all opinions and holders.

Our experiments show that the interaction-based models result in drastic improvements. Significantly, we see significant boosts in recall (10 points for both tasks) while the precision decreases only slightly, resulting in clear F-measure improvements. This result compares favorably with previously published results, which have been precision-oriented and scored quite low on recall.

We analyzed the impact of the syntactic and semantic features and saw that the best model is the one that makes use of both types of features. The most effective features we have found are purely structural, i.e. based on tree fragments in a syntactic or semantic tree. Features involving words did not seem to have the same impact.

There are multiple opportunities for future work in this area. An important issue that we have left open is the coreference problem for holder extraction, which has been studied by Stoyanov and Cardie (2006). Similarly, recent work has tried to incorporate complex, high-level linguistic structure such as discourse representations (Somasundaran et al., 2009); it is clear that these structures are very relevant for explaining the way humans organize their expressions of opinions rhetorically. However, theoretical depth does not necessarily guarantee practical applicability, and the challenge is as usual to find a middle ground that balances our goals: explanatory power in theory, significant performance gains in practice, computational tractability, and robustness in difficult circumstances.

6 Acknowledgements

The research described in this paper has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant 231126: LivingKnowledge – Facts, Opinions and Bias in Time, and under grant 247758: Trustworthy Eternal Systems via Evolving Software, Data and Knowledge (EternalS).

References

Bethard, Steven, Hong Yu, Ashley Thornton, Vasileios Hatzivassiloglou, and Dan Jurafsky. 2005. Extracting opinion propositions and opinion holders using syntactic and lexical cues. In Shanahan, James G., Yan Qu, and Janyce Wiebe, editors, *Computing Attitude and Affect in Text: Theory and Applications*.

Breck, Eric, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In *Proceedings of IJCAI-2007*, Hyderabad, India.

Choi, Yejin, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings of EMNLP 2006*.

Collins, Michael. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 1–8.

Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Schwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 2006(7):551–585.

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Freund, Yoav and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Johansson, Richard and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *CoNLL 2008: Proceedings of the Twelfth Conference on Natural Language Learning*, pages 183–187, Manchester, United Kingdom.

Joshi, Mahesh and Carolyn Penstein-Rosé. 2009. Generalizing dependency features for opinion mining. In *Proceedings of ACL/IJCNLP 2009, Short Papers Track*.

Kim, Soo-Min and Eduard Hovy. 2006. Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proceedings of ACL/COLING Workshop on Sentiment and Subjectivity in Text*.

Kobayashi, Nozomi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting aspect-evaluation and aspect-of relations in opinion mining. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-CoNLL-2007)*.

Mel'čuk, Igor A. 1988. *Dependency Syntax: Theory and Practice*. State University Press of New York, Albany.

Meyers, Adam, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The NomBank project: An interim report. In *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, United States.

Moschitti, Alessandro and Roberto Basili. 2004. Complex linguistic features for text classification: A comprehensive study. In *Proceedings of ECIR*.

Palmer, Martha, Dan Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105.

Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of EMNLP*.

Ruppenhofer, Josef, Swapna Somasundaran, and Janyce Wiebe. 2008. Finding the sources and targets of subjective expressions. In *Proceedings of LREC*.

Somasundaran, Swapna, Galileo Namata, Janyce Wiebe, and Lise Getoor. 2009. Supervised and unsupervised methods in employing discourse relations for improving opinion polarity classification. In *Proceedings of EMNLP*.

Stoyanov, Veselin and Claire Cardie. 2006. Partially supervised coreference resolution for opinion summarization through structured rule learning. In *Proceedings of EMNLP 2006*.

Tjong Kim Sang, Erik F. and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of EACL99*, pages 173–179, Bergen, Norway.

Tsochantaridis, Iannis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(Sep):1453–1484.

Wiebe, Janyce, Rebecca Bruce, and Thomas O'Hara. 1999. Development and use of a gold standard data set for subjectivity classifications. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.

Wiebe, Janyce, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2-3):165–210.

Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP 2005*.

Wu, Yuanbin, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proceedings of EMNLP*.

Yu, Hong and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2003)*, pages 129–136, Sapporo, Japan.

Unsupervised phonemic Chinese word segmentation using Adaptor Grammars

Mark Johnson
Department of Computing
Macquarie University
Mark.Johnson@mq.edu.au

Katherine Demuth
Department of Linguistics
Macquarie University
Katherine.Demuth@mq.edu.au

Abstract

Adaptor grammars are a framework for expressing and performing inference over a variety of non-parametric linguistic models. These models currently provide state-of-the-art performance on unsupervised word segmentation from phonemic representations of child-directed unsegmented English utterances. This paper investigates the applicability of these models to unsupervised word segmentation of Mandarin. We investigate a wide variety of different segmentation models, and show that the best segmentation accuracy is obtained from models that capture inter-word "collocational" dependencies. Surprisingly, enhancing the models to exploit syllable structure regularities and to capture tone information does improve overall word segmentation accuracy, perhaps because the information these elements convey is redundant when compared to the inter-word dependencies.

1 Introduction and previous work

The word-segmentation task is an abstraction of part of the problem facing a child learning its native language. Fluent speech, even the speech directed at children, doesn't come with silence or pauses delineating acoustic words the way that spaces separate orthographic words in writing systems like that of English. Instead, as most people listening to a language they don't understand can attest, words in fluent speech "run together", and a language user needs to learn how to segment utterances of the language they are learning into words. This kind of word segmentation is presumably an important first step in acquiring a language. It is scientifically interesting to know what information might be useful for word segmentation, and just how this information might be used. These scientific questions have motivated a body of research on computational models of word segmentation. Since as far as we can tell any child can learn any human language, our goal is to develop a single model that can learn to perform accurate word segmentation given input from any human language, rather than a model that specialised to perform well on a single language. This paper extends the previous work on word segmentation by investigating whether one class of models that work very well with English input also work with Chinese input. These models will permit us to study the role that syllable structure constraints and tone in Chinese might play in word segmentation.

While learners and fluent speakers undoubtedly use a wide variety of cues to perform word segmentation, computational models since Elman (1990) have tended to focus on the use of phonotactic constraints (e.g., syllable-structure constrains) and distributional information. Brent and Cartwright (1996) introduced the standard form of the word segmentation task still studied today. They extracted the orthographic representations of child-directed speech from the Bernstein-Ratner corpus (Bernstein-Ratner, 1987) and "phonologised" them by looking up each word in a pronouncing dictionary. For example, the orthographic utterance *you want to see the book* is mapped to the sequence of pronunciations *yu want tu si D6 bUk*, (the pronunciations are in an

ASCII encoding of the International Phonetic Alphabet representation of English phonemes). The input to the learner is obtained by concatenating together the phonemic representations of each utterance's words. The learner's task is to identify the locations of the word boundaries in this sequence, and hence identify the words (up to homophony). Brent and Cartwright (1996) pointed out the importance of both distributional information and phonotactic (e.g., syllable-structure) constraints for word segmentation (see also Swingley (2005) and Fleck (2008)).

Recently there has been considerable interest in applying Bayesian inference techniques for non-parametric models to this problem. Here the term "non-parametric" does not mean that the models have no parameters, rather, it is used to distinguish these models from the usual "parametric models" that have a fixed finite vector of parameters.

Goldwater et al. (2006) introduced two non-parametric Bayesian models of word segmentation, which are discussed in more detail in (Goldwater et al., 2009). The *unigram model*, which assumes that each word is generated independently to form a sentence, turned out to be equivalent to a model originally proposed by Brent (1999). The *bigram model* improves word segmentation accuracy by modelling bigram inter-word contextual dependencies, "explaining away" inter-word dependencies that would otherwise cause the unigram model to under-segment. Mochihashi et al. (2009) showed that segmentation accuracy could be improved by using a more sophisticated "base distribution" and a dynamic programming sampling algorithm very similar to the one used with the adaptor grammars below. They also applied their algorithm to Japanese and Chinese word segmentation, albeit from orthographic rather than phonemic forms, so unfortunately their results are not comparable with ours.

Johnson et al. (2007) introduced *adaptor grammars* as a grammar-based framework for expressing a variety of non-parametric models, and provided a dynamic programming Markov Chain Monte Carlo (MCMC) sampling algorithm for performing Bayesian inference on these models. For example, the unigram model can be expressed as a simple adaptor grammar as shown below, and the generic adaptor grammar inference procedure provides a dynamic programming sampling algorithm for this model. Johnson (2008b) showed how a variety of different word segmentation models can be expressed as adaptor grammars, and Johnson and Goldwater (2009) described a number of extensions and specialisations to the adaptor grammar framework that improve inference speed and accuracy (we use these techniques in our work below).

Previous work on unsupervised word segmentation from phonemic input has tended to concentrate on English. However, presumably children the world over segment their first language input in the same (innately-specified) way, so a correct procedure should work for all possible human languages. However, as far as we are aware there has been relatively little work on word segmentation from phonemic input except on English. Johnson (2008a) investigated whether the adaptor grammars models that do very well on English also apply to Sesotho (a Bantu language spoken in southern Africa with rich agglutinating morphology). He found that the models in general do very poorly (presumably because the adaptor grammars used cannot model the complex morphology found in Sesotho) and that the best segmentation accuracy was considerably worse than that obtained for English, even when that model incorporated some Bantu-specific information about morphology. Of course it may also be that the Sesotho and English corpora are not really comparable: the Bernstein-Ratner corpus that Brent and other researchers have used for English was spoken to pre-linguistic 1-year olds, while most non-English corpora are of child-directed speech to older children who are capable of talking back, and hence these corpora are presumably more complex. We discuss this issue in more detail in section 4 below.

2 A Chinese word segmentation corpus

Our goal here is to prepare a Chinese corpus of child-directed speech that parallels the English one used by Brent and other researchers. That corpus was in broad phonemic form, obtained by looking each word up in a pronouncing dictionary. Here instead we make use of a corpus in Pinyin format, which we translate into a broad phonemic IPA format using the freely-available Pinyin-to-IPA translation program “Pinyin to IPA Conversion Tools” version 2.1 available on http://sourceforge.net/projects/py2ipa.

We used the “Beijing” corpus (Tardif, 1993) available from the publicly-distributed Childes collection of corpora (MacWhinney and Snow, 1985). We are interested in child-directed speech (rather than children’s speech), so we removed all utterances from participants with an Id containing “Child”. (Tardif (1993) points out that Chinese-speaking children typically have a much richer social environment involving multiple adult caregivers than middle-class English-speaking children do, so we cannot simply collect only the mother’s utterances, as was done for the English corpus). We also ignored all utterances with codes $INTERJ, $UNINT, $VOC and $PRMPT, as these are not always linguistic utterances. In addition, we deleted all words that could not be analysed as a sequence of syllables, such as “xxx” and “hmm”, and also deleted “cluck”. The first few utterances of the corpus in Pinyin format are:

zen3me gei3 ta1 bei1 shang4 lai2 (1.) ?
ta1: (.) a1yi2 gei3 de (.) ta1 gei3 de .
hen3 jian3dan1 .

We then fed these into the Pinyin-to-IPA translation program, producing output of the following format:

tsən^{214}mɤ kei^{214} t^{h}a^{55} pei^{55} ʂɑŋ^{51} lai^{35}
t^{h}a^{55} a^{55}i^{35} kei^{214} tɤ t^{h}a^{55} kei^{214} tɤ
xən^{214} tɕiɛn^{214}tan^{55}

In the IPA format, the superscript indices indicate the tone patterns associated with syllables; these appear at the end of each syllable, as is standard. While we believe there are good linguistic reasons to analyse tones as associated with syllables, we moved all the tones so they immediately followed the final vowel in each syllable. We did this because we thought that locating tones after the syllable-final consonant might give our models a strong cue as to the location of syllable boundaries, and since words often end at syllable boundaries, this would make the word segmentation problem artificially easier. (Our models take a sequence of symbols as input, so the tones must be located somewhere in the sequence. However, the linguistically “correct” solution would probably be to extend the models so they could process input in an auto-segmental format (Goldsmith, 1990) where tones would be on a separate tier and unordered with respect to the segments within a syllable.)

In order to evaluate the importance of tone for our word-segmentation models we also constructed a version of our corpus in which all tones were removed. We present results for all of our models on two versions of the corpus, one that contains tones following the vowels, and another that contains no tones at all. These two corpora constitute the “gold standard” against which our word segmentation models will be evaluated. These corpora contain 50,118 utterances, consisting of 187,533 word tokens.

The training data provided to the word segmentation models is obtained by segmenting the gold data at all possible boundary locations. Consonant clusters, diphthongs and tones (if present) are treated as single units, so the training data appears as follows:

ts ə ^{214} n m ɤ k e i ^{214} t^{h} a ^{55} p e i ^{55} ʂ ɑ ^{51} ŋ l ai ^{35}
t^{h} a ^{55} a ^{55} i ^{35} k e i ^{214} t ɤ t^{h} a ^{55} k e i ^{214} t ɤ
x ə ^{214} n tɕ iɛ ^{214} n t a ^{55} n

The task of a word-segmentation model is to identify which of these possible boundary locations correspond to actual word boundaries. The training corpus without tones contains 531,384 segments, while the training corpus with tones contains 712,318 segments.

3 Adaptor grammars for word segmentation

Adaptor grammars were first introduced by Johnson et al. (2007) as a grammar-based framework for specifying hierarchical non-parametric Bayesian models, and Johnson and Goldwater (2009) describes a number of implementation details that significantly improve performance; the interested reader should consult those papers for a full technical introduction. Johnson (2008b) proposed a number of adaptor grammars for English word segmentation, which we review and minimally modify here so they can perform Chinese word segmentation as well. In section 4 we evaluate these adaptor grammars on the Chinese corpus just described.

The grammars vary along two orthogonal dimensions, which correspond to the kinds of generalisations that the model can learn. The simplest grammar is the unigram adaptor grammar, which generates an utterance as an i.i.d. sequences of words, where each word is a sequence of phonemes. The collocation adaptor grammars capture dependencies above the word level by generating collocations, or groups of words, as memoized units. The syllable adaptor grammars capture dependencies below the word level by generating words as sequences of syllables rather than phonemes.

3.1 Unigram adaptor grammars

In order to motivate adaptor grammars as an extension to Probabilistic Context-Free Grammars (PCFGs), consider an attempt to perform unsupervised word segmentation with a PCFG containing the following rules (ignore the underlining of the Word non-terminal for now).

$$\begin{array}{l} \text{Words} \rightarrow \text{Words } \underline{\text{Word}} \\ \text{Words} \rightarrow \underline{\text{Word}} \\ \underline{\text{Word}} \rightarrow \text{Phons} \\ \text{Phons} \rightarrow \text{Phon} \\ \text{Phons} \rightarrow \text{Phons Phon} \\ \text{Phons} \rightarrow \text{Phons Tone} \\ \text{Phon} \rightarrow \text{ai} \mid \text{o} \mid \ldots \mid \text{ʂ} \mid \text{tʂ}^{h} \mid \ldots \\ \text{Tone} \rightarrow 35 \mid 55 \mid 214 \mid \ldots \end{array} \qquad (1)$$

In this grammar, Phon expands to all the phonemes appearing in the phonemic training data, and Tone expands to all of the tone patterns. (In this and all of the other grammars in this paper, the start symbol is the non-terminal symbol of the first rule in the grammar. This grammar, like all others in this paper, is crafted so that a Word subtree can never begin with a Tone, so the presence of tones does not make the segmentation problem harder).

The trees generated by this grammar are sufficiently expressive to *represent* any possible segmentation of any sequence of phonemes into words (including the true segmentation); a typical segmentation is shown in Figure 1. However, it should also be clear that no matter how we vary the probabilities on the rules of this grammar, *the grammar itself cannot encode the subset of trees that correspond to words of the language*. In order to do this, a model would need to memorise the probabilities of entire Word subtrees, since these are the units that correspond to individual words, but this PCFG simply is not expressive enough to do this.

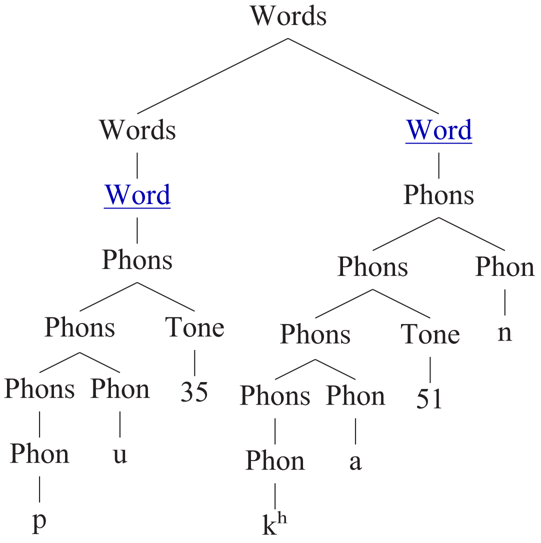


Figure 1: A parse tree generated by the unigram grammar, where adapted and non-adapted non-terminals are shown. It depicts a possible segmentation of p u ^{35} k^{h} a ^{51} n.

Adaptor grammars learn the probabilities of subtrees in just this way. An adaptor grammar is specified via a set of rules or productions, just like a CFG, and the set of trees that an adaptor grammar generates is exactly the same as the CFG with those rules.

However, an adaptor grammar defines probability distributions over trees in a completely different fashion to a PCFG: for simplicity we focus here on the sampling or predictive distribution, which defines the probability of generating an entire corpus of trees. In a PCFG, the probability of each non-terminal expanding using a given rule is determined by the probability of that rule, and is independent of the expansions of the other non-terminals in the tree. In an adaptor grammar a subset of the non-terminals are designated as *adapted*. We indicate adapted non-terminals by underlining them, so $\underline{\text{Word}}$ is the only adapted non-terminal in (1). Unadapted non-terminals expand just as in a PCFG: a production is chosen according to the production probabilities. An adapted non-terminal can expand in two different ways. With probability proportional to $n(t) - a_A$ an adapted non-terminal A expands to a tree t rooted in A that has been previously generated, while with probability proportional to $m(A)a_A + b_A$ the adapted non-terminal A expands using some grammar rule, just as in a PCFG. Here $n(t)$ is the number of times tree t has been previously generated, $m(A)$ is the number of trees rooted in A that have been previously generated using grammar rules, and $0 \leqslant a_A \leqslant 1$ and $b_A > 0$ are adjustable parameters associated with the adapted non-terminal A.

Technically, this is known as a *Pitman-Yor Process* (PYP) with *concentration parameters* a_A and b_A, where the PCFG rules define the *base distribution* of the process. (The PYP is a generalisation of the Chinese Restaurant Process (CRP); a CRP is a PYP with parameter $a = 0$). Rather than setting the concentration parameters by hand (there are two for each adapted non-terminal in the grammar) we follow Johnson and Goldwater (2009) and put uniform Beta and vague Gamma priors on each of these parameters, and use sampling to explore their posterior values.

Because the probability of selecting a tree t is proportional to $n(t)$, an adaptor grammar is a kind of "rich-get-richer" process that generates power-law distributions. Depending on the values of a_A and b_A, most of the probability mass can wind up concentrated on just a few trees. An adaptor grammar is a kind of "cache" model, in which previously generated subtrees are stored and more likely to be reused in later sentences. That is, while an adapted non-terminal A can expand to any tree rooted in A that can be constructed with the grammar rules, in practice it is increasingly likely to reuse the same trees over and over again. It can be viewed as a kind of tree substitution grammar (Joshi, 2003), but where the tree fragments (as well as their probabilities) are learnt from the data.

The unigram grammar is the simplest of the word segmentation models we investigate in this paper (it is equivalent to the unigram model investigated in Goldwater et al. (2009)). Because the grammars we present below rapidly become long and complicated to read if each grammar rule is explicitly stated, we adopt the following conventions. We use regular expressions to abbreviate our grammars, with the understanding that the regular expressions are always expanded produce a left-recursive structure. For example, the unigram grammar in (1) is abbreviated as:

$$\begin{aligned}
&\text{Words} \rightarrow \underline{\text{Word}}^{+} \\
&\underline{\text{Word}} \rightarrow \text{Phon (Phon} \mid \text{Tone)}^{\star} \\
&\text{Phon} \rightarrow \text{ai} \mid \text{o} \mid \ldots \mid \text{ʂ} \mid \text{tʂ}^{\text{h}} \mid \ldots \\
&\text{Tone} \rightarrow 35 \mid 55 \mid 214 \mid \ldots
\end{aligned} \qquad (2)$$

3.2 Collocation adaptor grammars

Goldwater et al. (2006) and Goldwater et al. (2009) demonstrated the importance of contextual dependencies for word segmentation, and proposed a bigram model in order to capture some of these. It turns out that while the bigram model cannot be expressed as an adaptor grammar, a *collocation model*, which captures similar kinds of contextual dependencies, can be expressed as an adaptor grammar (Johnson et al., 2007). In a collocation grammar there are two different adapted non-terminals; $\underline{\text{Word}}$ and $\underline{\text{Colloc}}$; $\underline{\text{Word}}$ expands exactly as in the unigram grammar (2), so it is not repeated here.

$$\begin{aligned}
&\text{Collocs} \rightarrow \underline{\text{Colloc}}^{+} \\
&\underline{\text{Colloc}} \rightarrow \text{Words} \\
&\text{Words} \rightarrow \underline{\text{Word}}^{+}
\end{aligned} \qquad (3)$$

A collocation adaptor grammar caches both words and collocations (which are sequences of words). An utterance is generated by generating one or more collocations. The PYP associated with collocations either regenerates a previously generated collocation or else generates a "fresh" collocation by generating a sequence of words according to the PYP model explained above.

The idea of aggregating words into collocations can be reapplied at a more abstract level by aggregating collocations into "super-collocations", which are sequences of collocations. This involves adding the following additional rules to the grammar in (3):

$$\begin{array}{l}\text{Colloc2s} \rightarrow \text{Colloc2}^{+}\\ \text{Colloc2} \rightarrow \text{Collocs}^{+}\end{array} \tag{4}$$

There are three PYPs in a grammar with 2 levels of collocations, arranged in a strict Bayesian hierarchy. It should be clear that this process can be repeated indefinitely; we investigate grammars with up to three levels of collocations below. (It should be possible to use Bayesian techniques to learn the appropriate number of levels in the hierarchy, but we leave this for future work).

3.3 Syllable structure adaptor grammars

Brent and Cartwright (1996) and others emphasise the role that syllable-structure and other phonotactic constraints might play in word segmentation. Johnson (2008b) pointed out that adaptor grammars can learn at least some of these kinds of generalisations. It's not unreasonable to assume that language learners can learn to group phonemes into syllables, and that they can exploit this syllabic structure to perform word segmentation. The syllable-structure grammars we describe below assume that word boundaries are always aligned with syllable boundaries; this is not universally true, but it is reliable enough to dramatically improve unsupervised word segmentation in English.

There is considerable cross-linguistic variation in the syllable-structure and phonotactic constraints operative in the languages of the world, so we'd like to avoid "building in" language-specific constraints into our model. We therefore make the relatively conservative assumption that the child can distinguish vowels from consonants, and that the child knows that syllables consist of Onsets, Nuclei and Codas, that Onsets and Codas consist of arbitrary sequences of consonants while Nuclei are arbitrary sequences of vowels and tones, and that Onsets and Codas are optional. Notice that syllable structure in both English and Chinese is considerably more constrained than this; we use this simple model here because it has proved successful for English word segmentation.

The syllable-structure adaptor grammars replace the rules expanding Words with the following rules:

$$\begin{array}{l}\text{Word} \rightarrow \text{Syll}\\ \text{Word} \rightarrow \text{Syll Syll}\\ \text{Word} \rightarrow \text{Syll Syll Syll}\\ \text{Word} \rightarrow \text{Syll Syll Syll Syll}\\ \text{Syll} \rightarrow (\text{Onset})^{?}\ \text{Rhy}\\ \text{Onset} \rightarrow \text{C}^{+}\\ \text{Rhy} \rightarrow \text{Nucleus}\ (\text{Coda})^{?}\\ \text{Nucleus} \rightarrow \text{V}\ (\text{V} \mid \text{Tone})^{\star}\\ \text{Coda} \rightarrow \text{C}^{+}\\ \text{C} \rightarrow \text{ʂ} \mid \text{tʂ}^{\text{h}} \mid \ldots\\ \text{V} \rightarrow \text{ai} \mid \text{o} \mid \ldots\end{array} \tag{5}$$

In these rules the superscript "?" indicates optionality. We used the relatively cumbersome mechanism of enumerating each possible number of syllables per word (we permit words to consist of from 1 to 4 syllables, although ideally this number would not be hard-wired into the grammar) because a relatively trivial modification of this grammar can distinguish word-initial and word-final consonant clusters from word-internal clusters. Johnson (2008b) demonstrated that this significantly improves English word segmentation accuracy. We do not expect this to improve Chinese word segmentation because Chinese clusters do not vary depending on their location within the word, but it will be interesting to see if the additional cluster flexibility that is useful for English segmentation hurts Chinese segmentation.

In this version of the syllable-structure grammar, we replace the Word rules in the syllable adaptor grammar with the following:

$$\begin{array}{l}\text{Word} \rightarrow \text{SyllIF}\\ \text{Word} \rightarrow \text{SyllI SyllF}\\ \text{Word} \rightarrow \text{SyllI Syll SyllF}\\ \text{Word} \rightarrow \text{SyllI Syll Syll SyllF}\end{array} \tag{6}$$

and add the following rules expanding the new kinds of syllables to the rules in (5).

$$\begin{array}{l}\text{SyllIF} \rightarrow (\text{OnsetI})^{?}\ \text{RhyF}\\ \text{SyllI} \rightarrow (\text{OnsetI})^{?}\ \text{Rhy}\\ \text{SyllF} \rightarrow (\text{OnsetI})^{?}\ \text{RhyF}\\ \text{Syll} \rightarrow (\text{Onset})^{?}\ \text{Rhy}\\ \text{OnsetI} \rightarrow \text{C}^{+}\\ \text{RhyF} \rightarrow \text{Nucleus}\ (\text{CodaF})^{?}\\ \text{CodaF} \rightarrow \text{C}^{+}\end{array} \tag{7}$$

	Syllables		
	None	General	Specialised
Unigram	0.57	0.50	0.50
Colloc	0.69	0.67	0.67
Colloc^2	0.72	0.75	0.75
Colloc^3	*0.64*	**0.77**	**0.77**

Table 1: F-score accuracies of word segmentations produced by the adaptor grammar models on the Chinese corpus *with tones*.

	Syllables		
	None	General	Specialised
Unigram	0.56	0.46	0.46
Colloc	0.70	0.65	0.65
Colloc^2	0.74	0.74	0.73
Colloc^3	0.75	0.76	**0.77**

Table 2: F-score accuracies of word segmentations produced by the adaptor grammar models on the Chinese corpus *without tones*.

These rules distinguish syllable onsets in word-initial position and syllable codas in word-final position; the standard adaptor grammar machinery will then learn distributions over onsets and codas in these positions that possibly differ from those in word-internal positions.

4 Results on Chinese word segmentation

The previous section described two dimensions along which adaptor grammars for word segmentation can independently vary. Above the Word level, there can be from zero to three levels of collocations, yielding four different values for this dimension. Below the Word level, phonemes can either be treated as independent entities, or else they can be grouped into onset, nuclei and coda clusters, and these can vary depending on where they appear within a word. Thus there are three different values for the syllable dimension, so there are twelve different adaptor grammars overall. In addition, we ran all of these grammars on two versions of the corpus, one with tones and one without tones, so we report results for 24 different runs here.

The adaptor grammar inference procedure we used is the one described in Johnson and Goldwater (2009). We ran 1,000 iterations of 8 MCMC chains for each run, and we discarded all but last 200 iterations in order to "burn-in" the sampler. The segmentation we predict is the one that occurs the most frequently in the samples that were not discarded. As is standard, we evaluate the models in terms of token f-score; the results are presented in Tables 1 and 2.

In these tables, "None" indicates that the grammar does not model syllable structure, "General" indicates that the grammar does not distinguish word-peripheral from word-internal clusters, while "Specialised" indicates that it does. "Unigram" indicates that the grammar does not model collocational structure, otherwise the superscript indicates the number of collocational levels that the grammar captures.

Broadly speaking, the results are consistent with the English word segmentation results using adaptor grammars presented by Johnson (2008b). The unigram grammar segmentation accuracy is similar to that obtained for English, but the results for the other models are lower than the results for the corresponding adaptor grammars on English.

We see a general improvement in segmentation accuracy as the number of collocation levels increases, just as for English. However, we do not see any general improvements associated with modelling syllables; indeed, it seems modelling syllables causes accuracy to decrease unless collocational structure is also modelled. This is somewhat surprising, as Chinese has a very regular syllabic structure. It is not surprising that distinguishing word-peripheral and word-medial clusters does not improve segmentation accuracy, as Chinese does not distinguish these kinds of clusters. There is also no sign of the "synergies" when modelling collocations and syllables together that Johnson (2008b) reported.

It is also surprising that tones seem to make little difference to the segmentation accuracy, since they are crucial for disambiguating lexical items. The segmentation accuracy of the models that capture little or no inter-word dependencies (e.g., Unigram, Colloc) improved slightly when the input contains tones, but the best-performing models that capture a more complex set of inter-word dependencies do equally well on the corpus without tones as they do on the corpus with tones. Because these models capture rich inter-word context (they model three levels of collocational structure), it is possible that this context provides sufficient information to segment words even in the absence of tone information, i.e., the tonal information is redundant given the richer inter-word dependencies that these models capture. It is also possible that word segmentation may simply require less information than lexical disambiguation.

One surprising result is the relatively poor performance of the Colloc^3 model without syllables but with tones; we have no explanation for this. However, all 8 of the MCMC chains in this run produced lower f-scores, so it unlikely to be simply a random fluctuation produced by a single outlier.

Note that one should be cautious when comparing the absolute f-scores from these experiments with those of the English study, as the English and Chinese corpora differ in many ways. As Tardif (1993) (the creator of the Chinese corpus) emphasises, this corpus was collected in a much more diverse linguistic environment with child-directed speech from multiple caregivers. The children involved in the Chinese corpus were also older than the children in the English corpus, which may also have affected the nature of the corpus.

5 Conclusion

This paper applied adaptor grammar models of phonemic word segmentation originally developed for English to Chinese data. While the Chinese data was prepared in a very different way to the English data, the adaptor grammars used to perform Chinese word segmentation were very similar to those used for the English word segmentation. They also achieved quite respectable f-score accuracies, which suggests that the same models can do well on both languages.

One puzzling result is that incorporating syllable structure phonotactic constraints, which enhances English word segmentation accuracy considerably, doesn't seem to improve Chinese word segmentation to a similar extent. This may reflect the fact that the word segmentation adaptor grammars were originally designed and tuned for English, and perhaps differently formulated syllable-structure constraints would work well for Chinese. But even if one can "tune" the adaptor grammars to improve performance on Chinese, the challenge is doing this in a way that improves performance on all languages, rather than just one.

Acknowledgments

The authors would like to thank the US NSF for support for this research, which was begun while they were on the faculty at Brown University in the USA. The NSF supported this work through NSF awards 0544127 and 0631667 to Mark Johnson and Katherine Demuth.

The adaptor grammar software is freely available for download from http://web.science.mq.edu.au/˜mjohnson, and the Chinese data was obtained from the Childes archive.

References

Bernstein-Ratner, N. 1987. The phonology of parent-child speech. In Nelson, K. and A. van Kleeck, editors, *Children's Language*, volume 6. Erlbaum, Hillsdale, NJ.

Brent, M. and T. Cartwright. 1996. Distributional regularity and phonotactic constraints are useful for segmentation. *Cognition*, 61:93–125.

Brent, M. 1999. An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34:71–105.

Elman, Jeffrey. 1990. Finding structure in time. *Cognitive Science*, 14:197–211.

Fleck, Margaret M. 2008. Lexicalized phonotactic word segmentation. In *Proceedings of ACL-08: HLT*, pages 130–138, Columbus, Ohio, June. Association for Computational Linguistics.

Goldsmith, John A. 1990. *Autosegmental and Metrical Phonology*. Basil Blackwell, Oxford, England.

Goldwater, Sharon, Thomas L. Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 673–680, Sydney, Australia. Association for Computational Linguistics.

Goldwater, Sharon, Thomas L. Griffiths, and Mark Johnson. 2009. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1):21 – 54.

Johnson, Mark and Sharon Goldwater. 2009. Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325, Boulder, Colorado, June. Association for Computational Linguistics.

Johnson, Mark, Thomas L. Griffiths, and Sharon Goldwater. 2007. Adaptor Grammars: A framework for specifying compositional nonparametric Bayesian models. In Schölkopf, B., J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 641–648. MIT Press, Cambridge, MA.

Johnson, Mark. 2008a. Unsupervised word segmentation for Sesotho using adaptor grammars. In *Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology*, pages 20–27, Columbus, Ohio, June. Association for Computational Linguistics.

Johnson, Mark. 2008b. Using adaptor grammars to identifying synergies in the unsupervised acquisition of linguistic structure. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, Columbus, Ohio. Association for Computational Linguistics.

Joshi, Aravind. 2003. Tree adjoining grammars. In Mikkov, Ruslan, editor, *The Oxford Handbook of Computational Linguistics*, pages 483–501. Oxford University Press, Oxford, England.

MacWhinney, Brian and Catherine Snow. 1985. The child language data exchange system. *Journal of Child Language*, 12:271–296.

Mochihashi, Daichi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 100–108, Suntec, Singapore, August. Association for Computational Linguistics.

Swingley, Dan. 2005. Statistical clustering and the contents of the infant vocabulary. *Cognitive Psychology*, 50:86–132.

Tardif, Twila. 1993. *Adult-to-child speech and language acquisition in Mandarin Chinese*. Ph.D. thesis, Yale University.

Data-Driven Parsing with Probabilistic Linear Context-Free Rewriting Systems

Laura Kallmeyer and **Wolfgang Maier**
SFB 833, University of Tübingen
{lk,wmaier}@sfs.uni-tuebingen.de

Abstract

This paper presents a first efficient implementation of a weighted deductive CYK parser for Probabilistic Linear Context-Free Rewriting Systems (PLCFRS), together with context-summary estimates for parse items used to speed up parsing. LCFRS, an extension of CFG, can describe discontinuities both in constituency and dependency structures in a straightforward way and is therefore a natural candidate to be used for data-driven parsing. We evaluate our parser with a grammar extracted from the German NeGra treebank. Our experiments show that data-driven LCFRS parsing is feasible with a reasonable speed and yields output of competitive quality.

1 Introduction

Data-driven parsing has largely been dominated by Probabilistic Context-Free Grammar (PCFG). The use of PCFG is tied to the annotation principles of popular treebanks, such as the Penn Treebank (PTB) (Marcus et al., 1994), which are used as a data source for grammar extraction. Their annotation generally relies on the use of trees without crossing branches, augmented with a mechanism that accounts for non-local dependencies. In the PTB, e.g., labeling conventions and trace nodes are used which establish additional implicit edges in the tree beyond the overt phrase structure. In contrast, some other treebanks, such as the German NeGra and TIGER treebanks allow annotation with crossing branches (Skut et al., 1997). Non-local dependencies can then be expressed directly by grouping all dependent elements under a single node.

However, given the expressivity restrictions of PCFG, work on data-driven parsing has mostly excluded non-local dependencies. When using treebanks with PTB-like annotation, labeling conventions and trace nodes are often discarded, while in NeGra, resp. TIGER, tree transformations are applied which resolve the crossing branches (Kübler, 2005; Boyd, 2007, e.g.). Especially for these treebanks, such a transformation is questionable, since it is non-reversible and implies information loss.

Some research has gone into incorporating non-local information into data-driven parsing. Levy and Manning (2004) distinguish three approaches: 1. Non-local information can be incorporated directly into the PCFG model (Collins, 1999), or can be reconstructed in a post-processing step after PCFG parsing (Johnson, 2002; Levy and Manning, 2004). 2. Non-local information can be incorporated into complex labels (Hockenmaier, 2003). 3. A formalism can be used which accommodates the direct encoding of non-local information (Plaehn, 2004). This paper pursues the third approach.

Our work is motivated by the following recent developments: Linear Context-Free Rewriting Systems (LCFRS) (Vijay-Shanker et al., 1987) have been established as a candidate for modeling both discontinuous constituents and non-projective dependency trees as they occur in treebanks (Kuhlmann and Satta, 2009; Maier and Lichte, 2009). LCFRS extend CFG such that non-terminals can span tuples of possibly non-

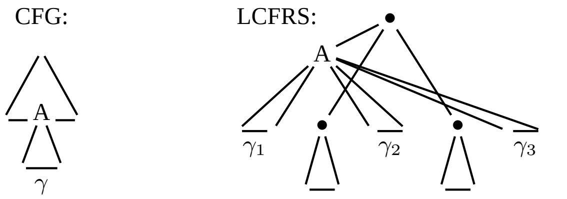


Figure 1: Different domains of locality

adjacent strings (see Fig. 1). PCFG techniques, such as Best-First Parsing (Charniak and Caraballo, 1998), Weighted Deductive Parsing (Nederhof, 2003) and A* parsing (Klein and Manning, 2003a), can be transferred to LCFRS. Finally, German has attracted the interest of the parsing community due to the challenges arising from its frequent discontinuous constituents (Kübler and Penn, 2008).

We bring together these developments by presenting a parser for probabilistic LCFRS. While parsers for subclasses of PLCFRS have been presented before (Kato et al., 2006), to our knowledge, our parser is the first for the entire class of PLCFRS. We have already presented an application of the parser on constituency and dependency treebanks together with an extensive evaluation (Maier, 2010; Maier and Kallmeyer, 2010). This article is mainly dedicated to the presentation of several methods for context summary estimation of parse items, and to an experimental evaluation of their usefulness. The estimates either act as figures-of-merit in a best-first parsing context or as estimates for A* parsing. Our evaluation shows that while our parser achieves a reasonable speed already without estimates, the estimates lead to a great reduction of the number of produced items, all while preserving the output quality.

Sect. 2 and 3 of the paper introduce probabilistic LCFRS and the parsing algorithm. Sect. 4 presents different context summary estimates. In Sect. 5, the implementation and evaluation of the work is discussed.

2 Probabilistic LCFRS

LCFRS are an extension of CFG where the nonterminals can span not only single strings but, instead, tuples of strings. We will notate LCFRS with the syntax of *simple Range Concatenation Grammars* (SRCG) (Boullier, 1998), a formalism that is equivalent to LCFRS.

A LCFRS (Vijay-Shanker et al., 1987) is a tuple $\langle N, T, V, P, S \rangle$ where a) N is a finite set of non-terminals with a function dim: $N \to \mathbb{N}$ that determines the *fan-out* of each $A \in N$; b) T and V are disjoint finite sets of terminals and variables; c) $S \in N$ is the start symbol with $dim(S) = 1$; d) P is a finite set of rules

$$A(\alpha_1, \ldots, \alpha_{dim(A)}) \to A_1(X_1^{(1)}, \ldots, X_{dim(A_1)}^{(1)}) \cdots A_m(X_1^{(m)}, \ldots, X_{dim(A_m)}^{(m)})$$

for $m \geq 0$ where $A, A_1, \ldots, A_m \in N$, $X_j^{(i)} \in V$ for $1 \leq i \leq m, 1 \leq j \leq dim(A_i)$ and $\alpha_i \in (T \cup V)^*$ for $1 \leq i \leq dim(A)$. For all $r \in P$, it holds that every variable X occurring in r occurs exactly once in the left-hand side (LHS) and exactly once in the right-hand side (RHS).

A rewriting rule describes how the yield of the LHS non-terminal can be computed from the yields of the RHS non-terminals. The rules $A(ab, cd) \to \varepsilon$ and $A(aXb, cYd) \to A(X, Y)$ for instance specify that 1. $\langle ab, cd \rangle$ is in the yield of A and 2. one can compute a new tuple in the yield of A from an already existing one by wrapping a and b around the first component and c and d around the second.

For every $A \in N$ in a LCFRS G, we define the yield of A, $yield(A)$ as follows:

a) For every $A(\vec{\alpha}) \to \varepsilon$, $\vec{\alpha} \in yield(A)$;

b) For every rule

$$A(\alpha_1, \ldots, \alpha_{dim(A)}) \to A_1(X_1^{(1)}, \ldots, X_{dim(A_1)}^{(1)}) \cdots A_m(X_1^{(m)}, \ldots, X_{dim(A_m)}^{(m)})$$

and all $\vec{\tau_i} \in yield(A_i)$ for $1 \leq i \leq m$, $\langle f(\alpha_1), \ldots, f(\alpha_{dim(A)}) \rangle \in yield(A)$ where f is defined as follows: (i) $f(t) = t$ for all $t \in T$, (ii) $f(X_j^{(i)}) = \vec{\tau_i}(j)$ for all $1 \leq i \leq m, 1 \leq j \leq dim(A_i)$ and (iii) $f(xy) = f(x)f(y)$ for all $x, y \in (T \cup V)^+$. f is the *composition function* of the rule.

c) Nothing else is in $yield(A)$.

The language is then $\{w \mid \langle w \rangle \in yield(S)\}$.

The *fan-out* of an LCFRS G is the maximal fan-out of all non-terminals in G. Furthermore, the RHS length of a rewriting rules $r \in P$ is called the *rank* of r and the maximal rank of all rules in P is called the *rank* of G. We call a LCFRS *ordered* if for every $r \in P$ and every RHS non-terminal A in r and each pair X_1, X_2 of arguments of A in the RHS of r, X_1 precedes X_2 in the RHS iff X_1 precedes X_2 in the LHS.

A *probabilistic LCFRS* (PLCFRS) (Kato et al., 2006) is a tuple $\langle N, T, V, P, S, p\rangle$ such that $\langle N, T, V, P, S\rangle$ is a LCFRS and $p : P \rightarrow [0..1]$ a function such that for all $A \in N$: $\Sigma_{A(\vec{x})\rightarrow\vec{\Phi}\in P} p(A(\vec{x}) \rightarrow \vec{\Phi}) = 1$.

3 The CYK Parser

We use a probabilistic version of the CYK parser from (Seki et al., 1991), applying techniques of weighted deductive parsing (Nederhof, 2003).

LCFRS can be binarized (Gómez-Rodríguez et al., 2009) and ε-components in the LHS of rules can be removed (Boullier, 1998). We can therefore assume that all rules are of rank 2 and do not contain ε components in their LHS. Furthermore, we assume POS tagging to be done before parsing. POS tags are non-terminals of fan-out 1. The rules are then either of the form $A(a) \rightarrow \varepsilon$ with A a POS tag and $a \in T$ or of the form $A(\vec{\alpha}) \rightarrow B(\vec{x})$ or $A(\vec{\alpha}) \rightarrow B(\vec{x})C(\vec{y})$ where $\vec{\alpha} \in (V^+)^{dim(A)}$, i.e., only the rules for POS tags contain terminals in their LHSs.

For every $w \in T^*$, where $w = w_1 \dots w_n$ with $w_i \in T$ for $1 \leq i \leq n$, we define: $Pos(w) := \{0, \dots, n\}$. A pair $\langle l, r\rangle \in Pos(w) \times Pos(w)$ with $l \leq r$ is a *range* in w. Its *yield* $\langle l, r\rangle(w)$ is the string $w_{l+1} \dots w_r$. The yield $\vec{\rho}(w)$ of a vector of ranges $\vec{\rho}$ is the vector of the yields of the single ranges. For two ranges $\rho_1 = \langle l_1, r_1\rangle, \rho_2 = \langle l_2, r_2\rangle$: if $r_1 = l_2$, then $\rho_1 \cdot \rho_2 = \langle l_1, r_2\rangle$; otherwise $\rho_1 \cdot \rho_2$ is undefined.

For a given rule $p : A(\alpha_1, \dots, \alpha_{dim(A)}) \rightarrow B(X_1, \dots, X_{dim(B)})C(Y_1, \dots, X_{dim(C)})$ we now extend the composition function f to ranges, given an input w: for all range vectors $\vec{\rho_B}$ and $\vec{\rho_C}$ of dimensions $dim(B)$ and $dim(C)$ respectively, $f_r(\vec{\rho_B}, \vec{\rho_C}) = \langle g(\alpha_1), \dots, g(\alpha_{dim(A)})\rangle$ is defined as follows: $g(X_i) = \vec{\rho_B}(i)$ for all $1 \leq i \leq dim(B)$, $g(Y_i) = \vec{\rho_C}(i)$ for all $1 \leq i \leq dim(C)$ and $g(xy) = g(x) \cdot g(y)$ for all $x, y \in V^+$. $p : A(f_r(\vec{\rho_B}, \vec{\rho_C})) \rightarrow B(\vec{\rho_B})C(\vec{\rho_C})$ is then called an *instantiated rule*.

For a given input w, our items have the form $[A, \vec{\rho}]$ where $A \in N$, $\vec{\rho} \in (Pos(w) \times Pos(w))^{dim(A)}$. The vector $\vec{\rho}$ characterizes the span of A. We specify the set of weighted parse items via the deduction rules in Fig. 2. Our parser performs a weighted deductive parsing (Nederhof, 2003), based on this deduction system. We use a chart $\mathcal{C}$ and an agenda $\mathcal{A}$, both initially empty, and we proceed as in Fig. 3.

Scan: $\dfrac{}{0 : [A, \langle\langle i, i+1\rangle\rangle]}$ A POS tag of w_{i+1}

Unary: $\dfrac{in : [B, \vec{\rho}]}{in + |log(p)| : [A, \vec{\rho}]}$ $p : A(\vec{\alpha}) \rightarrow B(\vec{\alpha}) \in P$

Binary: $\dfrac{in_B : [B, \vec{\rho_B}], in_C : [C, \vec{\rho_C}]}{in_B + in_C + log(p) : [A, \vec{\rho_A}]}$

where $p : A(\vec{\rho_A}) \rightarrow B(\vec{\rho_B})C(\vec{\rho_C})$ is an instantiated rule.

Goal: $[S, \langle\langle 0, n\rangle\rangle]$

Figure 2: Weighted CYK deduction system

```
add SCAN results to $\mathcal{A}$
while $\mathcal{A} \neq \emptyset$
  remove best item $x : I$ from $\mathcal{A}$
  add $x : I$ to $\mathcal{C}$
  if $I$ goal item
  then stop and output true
  else
    for all $y : I'$ deduced from $x : I$ and items in $\mathcal{C}$:
      if there is no $z$ with $z : I' \in \mathcal{C} \cup \mathcal{A}$
      then add $y : I'$ to $\mathcal{A}$
      else if $z : I' \in \mathcal{A}$ for some $z$
        then update weight of $I'$ in $\mathcal{A}$ to $max(y, z)$
```

Figure 3: Weighted deductive parsing

4 Outside Estimates

In order to speed up parsing, we add an estimate of the log of the outside probabilities of the items to their weights in the agenda. All our outside estimates are *admissible* (Klein and Manning, 2003a) which means that they never underestimate the actual outside probability of an item. However, most of them are not monotonic. In other words, it can happen that we deduce an item I_2 from an item I_1 where the weight of I_2 is greater than the weight of I_1. The parser can therefore end up in a local maximum that is not the global maximum we are searching for. In other words, our outside weights are only *figures of merit* (FOM). Only for the full SX estimate, the monotonicity is guaranteed and we can do true A* parsing as described in (Klein and Manning, 2003a) that always finds the best parse.

All outside estimates are computed for a certain maximal sentence length len_{max}.

POS tags: $\dfrac{}{0 : [A, \langle 1 \rangle]}$ A a POS tag

Unary: $\dfrac{in : [B, \vec{l}]}{in + log(p) : [A, \vec{l}]}$ $p : A(\vec{\alpha}) \rightarrow B(\vec{\alpha}) \in P$

Binary: $\dfrac{in_B : [B, \vec{l_B}], in_C : [C, \vec{l_C}]}{in_B + in_C + log(p) : [A, \vec{l_A}]}$

where $p : A(\vec{\alpha_A}) \rightarrow B(\vec{\alpha_B})C(\vec{\alpha_C}) \in P$ and the following holds: we define $\mathcal{B}(i)$ as $\{1 \leq j \leq dim(B) \,|\, \vec{\alpha_B}(j)$ occurs in $\vec{\alpha_A}(i)\}$ and $\mathcal{C}(i)$ as $\{1 \leq j \leq dim(C) \,|\, \vec{\alpha_C}(j)$ occurs in $\vec{\alpha_A}(i)\}$. Then for all i, $1 \leq i \leq dim(A)$: $\vec{l}_A(i) = \Sigma_{j \in \mathcal{B}(i)} \vec{l}_B(j) + \Sigma_{j \in \mathcal{C}(i)} \vec{l}_C(j)$.

Figure 4: Inside estimate

Axiom : $\dfrac{}{0 : [S, \langle 0, len, 0 \rangle]}$ $1 \leq len \leq len_{max}$

Unary: $\dfrac{w : [A, \vec{l}]}{w + log(p) : [B, \vec{l}]}$ $p : A(\vec{\alpha}) \rightarrow B(\vec{\alpha}) \in P$

Binary-right:

$$\frac{w : [X, \vec{l}_X]}{w + in(A, \vec{l'}_A) + log(p) : [B, \vec{l}_B]}$$

Binary-left:

$$\frac{w : [X, \vec{l}_X]}{w + in(B, \vec{l'}_B) + log(p) : [A, \vec{l}_A]}$$

where, for both rules, there is an instantiated rule $p : X(\vec{\rho}) \rightarrow A(\vec{\rho_A})B(\vec{\rho_B})$ such that $\vec{l}_X = l_{out}(\rho)$, $\vec{l}_A = l_{out}(\rho_A), \vec{l'}_A = l_{in}(\rho_A), \vec{l}_B = l_{out}(\rho_B, \vec{l}_B = l_{in}(\rho_B$.

Figure 5: Full SX estimate top-down

4.1 Full SX estimate

The full SX estimate, for a given sentence length n, is supposed to give the minimal costs (maximal probability) of completing a category X with a span ρ into an S with span $\langle\langle 0, n \rangle\rangle$.

For the computation, we need an estimate of the inside probability of a category C with a span ρ, regardless of the actual terminals in our input. This inside estimate is computed as shown in Fig. 4. Here, we do not need to consider the number of terminals outside the span of C (to the left or right or in the gaps), they are not relevant for the inside probability. Therefore the items have the form $[A, \langle l_1, \dots, l_{dim(A)} \rangle]$, where A is a non-terminal and l_i gives the length of its ith component. It holds that $\Sigma_{1 \leq i \leq dim(A)} l_i \leq len_{max} - dim(A) + 1$.

A straight-forward extension of the CFG algorithm from (Klein and Manning, 2003a) for computing the SX estimate is given in Fig. 5. For a given range vector $\rho = \langle\langle l_1, r_1 \rangle, \dots, \langle l_k, r_k \rangle\rangle$ and a sentence length n, we define its *inside length vector* $l_{in}(\rho)$ as $\langle r_1 - l_1, \dots, r_k - l_k \rangle$ and its *outside length vector* $l_{out}(\rho)$ as $\langle l_1, r_1 - l_1, l_2 - r_1, \dots, l_k - r_{k-1}, r_k - l_k, n - r_k \rangle$.

This algorithm has two major problems: Since it proceeds top-down, in the *Binary* rules, we must compute all splits of the antecedent X span into the spans of A and B which is very expensive. Furthermore, for a category A with a certain number of terminals in the components and the gaps, we compute the lower part of the outside estimate several times, namely for every combination of number of terminals to the left and to the right (first and last element in the outside length vector). In order to avoid these problems, we now abstract away from the lengths of the part to the left and the right, modifying our items such as to allow a bottom-up strategy.

The idea is to compute the weights of items representing the derivations from a certain lower C up to some A (C is a kind of "gap" in the yield of A) while summing up the inside costs of off-spine nodes and the log of the probabilities of the corresponding rules. We use items $[A, C, \rho_A, \rho_C, \mathit{shift}]$ where $A, C \in N$ and ρ_A, ρ_C are range vectors, both with a first component starting at position 0. The integer $\mathit{shift} \leq len_{max}$ tells us how many positions to the right the C span is shifted, compared to the starting position of the A. ρ_A and ρ_C represent the spans of C and A while disregarding the number of terminals to the left the right. I.e., only the lengths of the components and of the gaps are encoded. This means in particular that the length n of the sentence does not play a role here. The right boundary of the last range in the vectors is limited to len_{max}. For any $i, 0 \leq i \leq len_{max}$, and any range vector ρ, we define $\mathit{shift}(\rho, i)$ as the range vector one obtains from adding i to all range boundaries in ρ and $\mathit{shift}(\rho, -i)$ as the range vector one obtains from subtracting i from all boundaries in ρ.

The weight of $[A, C, \rho_A, \rho_C, i]$ estimates the costs for completing a C tree with yield ρ_C into an A tree with yield ρ_A such that, if the span of A starts at position j, the span of C starts at position $i + j$. Fig. 6 gives the computation. The value of $in(A, \vec{l})$ is the inside estimate of $[A, \vec{l}]$.

The SX-estimate for some predicate C with

POS tags: $\frac{}{0 : [C, C, \langle 0,1\rangle, \langle 0,1\rangle, 0]}$ C a POS tag

Unary: $\frac{0 : [B, B, \rho_B, \rho_B, 0]}{\mathit{log}(p) : [A, B, \rho_B, \rho_B, 0]}$ $p : A(\vec{\alpha}) \to B(\vec{\alpha}) \in P$

Binary-right:

$$\frac{0 : [A, A, \rho_A, \rho_A, 0], 0 : [B, B, \rho_B, \rho_B, 0]}{\mathit{in}(A, l(\rho_A)) + \mathit{log}(p) : [X, B, \rho_X, \rho_B, i]}$$

Binary-left:

$$\frac{0 : [A, A, \rho_A, \rho_A, 0], 0 : [B, B, \rho_B, \rho_B, 0]}{\mathit{in}(B, l(\rho_B)) + \mathit{log}(p) : [X, A, \rho_X, \rho_A, 0]}$$

where i is such that for $\mathit{shift}(\rho_B, i) = \rho'_B$ $p : X(\rho_X) \to A(\rho_A)B(\rho'_B)$ is an instantiated rule.

Starting sub-trees with larger gaps:

$$\frac{w : [B, C, \rho_B, \rho_C, i]}{0 : [B, B, \rho_B, \rho_B, 0]}$$

Transitive closure of sub-tree combination:

$$\frac{w_1 : [A, B, \rho_A, \rho_B, i], w_2 : [B, C, \rho_B, \rho_C, j]}{w_1 + w_2 : [A, C, \rho_A, \rho_C, i + j]}$$

Figure 6: Full SX estimate bottom-up

span ρ where i is the left boundary of the first component of ρ and with sentence length n is then given by the maximal weight of $[S, C, \langle 0, n\rangle, \mathit{shift}(-i, \rho), i]$. Among our estimates, the full SX estimate is the only one that is monotonic and that allows for true A* parsing.

4.2 SX with Left, Gaps, Right, Length

A problem of the previous estimate is that with a large number of non-terminals the computation of the estimate requires too much space. Our experiments have shown that for treebank parsing where we have, after binarization and markovization, appr. 12,000 non-terminals, its computation is not feasible. We therefore turn to simpler estimates with only a single non-terminal per item. We now estimate the outside probability of a non-terminal A with a span of a length length (the sum of the lengths of all the components of the span), with left terminals to the left of the first component, right terminals to the right of the last component and gaps terminals in between the components of the A span, i.e., filling the gaps. Our items have the form $[X, \mathit{len}, \mathit{left}, \mathit{right}, \mathit{gaps}]$ with $X \in N$, $\mathit{len} + \mathit{left} + \mathit{right} + \mathit{gaps} \leq \mathit{len}_{max}$, $\mathit{len} \geq \mathit{dim}(X)$, $\mathit{gaps} \geq \mathit{dim}(X) - 1$.

Let us assume that, in the rule $X(\vec{\alpha}) \to A(\vec{\alpha_A})B(\vec{\alpha_B})$, when looking at the vector $\vec{\alpha}$, we have $left_A$ variables for A-components preceding the first variable of a B component, $right_A$ variables for A-components following the last variable of a B component and $right_B$ variables for B-components following the last variable of a A component. (In our grammars, the first LHS argument always starts with the first variable from A.) Furthermore, $gaps_A = \mathit{dim}(A) - left_A - right_A$, $gaps_B = \mathit{dim}(B) - right_B$.

Axiom : $\frac{}{0 : [S, \mathit{len}, 0, 0, 0]}$ $1 \leq \mathit{len} \leq \mathit{len}_{max}$

Unary: $\frac{w : [X, \mathit{len}, l, r, g]}{w + \mathit{log}(p) : [A, \mathit{len}, l, r, g]}$

where $p : X(\vec{\alpha}) \to A(\vec{\alpha}) \in P$.

Binary-right:

$$\frac{w : [X, \mathit{len}, l, r, g]}{w + \mathit{in}(A, \mathit{len} - \mathit{len}_B) + \mathit{log}(p) : [B, \mathit{len}_B, l_B, r_B, g_B]}$$

Binary-left:

$$\frac{w : [X, \mathit{len}, l, r, g]}{w + \mathit{in}(B, \mathit{len} - \mathit{len}_A) + \mathit{log}(p) : [A, \mathit{len}_A, l_A, r_A, g_A]}$$

where, for both rules, $p : X(\vec{\alpha}) \to A(\vec{\alpha_A})B(\vec{\alpha_B}) \in P$.

Figure 7: SX with length, left, right, gaps

POS tags: $\frac{}{0 : [A, 1]}$ A a POS tag

Unary: $\frac{\mathit{in} : [B, l]}{\mathit{in} + \mathit{log}(p) : [A, l]}$ $p : A(\vec{\alpha}) \to B(\vec{\alpha}) \in P$

Binary: $\frac{\mathit{in}_B : [B, l_B], \mathit{in}_C : [C, l_C]}{\mathit{in}_B + \mathit{in}_C + \mathit{log}(p) : [A, l_B + l_C]}$

where either $p : A(\vec{\alpha_A}) \to B(\vec{\alpha_B})C(\vec{\alpha_C}) \in P$ or $p : A(\vec{\alpha_A}) \to C(\vec{\alpha_C})B(\vec{\alpha_B}) \in P$.

Figure 8: Inside estimate with total span length

Fig. 7 gives the computation of the estimate. The following side conditions must hold: For *Binary-right* to apply, the following constraints must be satisfied: a) $\mathit{len} + l + r + g = \mathit{len}_B + l_B + r_B + g_B$, b) $l_B \geq l + left_A$, c) if $right_A > 0$, then $r_B \geq r + right_A$, else ($right_A = 0$), $r_B = r$, d) $g_B \geq gaps_A$. Similarly, for *Binary-left* to apply, the following constraints must be satisfied: a) $\mathit{len} + l + r + g = \mathit{len}_A + l_A + r_A + g_A$, b) $l_A = l$, c) if $right_B > 0$, then $r_A \geq r + right_B$, else ($right_B = 0$), $r_A = r$ d) $g_A \geq gaps_B$.

The value $\mathit{in}(X, l)$ for a non-terminal X and a length l, $0 \leq l \leq \mathit{len}_{max}$ is an estimate of the probability of an X category with a span of length l. Its computation is specified in Fig. 8.

The SX-estimate for a sentence length n and for some predicate C with a range characterized by $\vec{\rho} = \langle\langle l_1, r_1\rangle, \ldots, \langle l_{\mathit{dim}(C)}, r_{\mathit{dim}(C)}\rangle\rangle$ where $\mathit{len} = \Sigma_{i=1}^{\mathit{dim}(C)}(r_i - l_i)$ and $r = n - r_{\mathit{dim}(C)}$ is then given by the maximal weight of the item $[C, \mathit{len}, l_1, r, n - \mathit{len} - l_1 - r]$.

Axiom : $\dfrac{}{0 : [S, len, 0, 0]}\ 1 \le len \le len_{max}$

Unary: $\dfrac{w : [X, len, lr, g]}{w + log(p) : [A, len, lr, g]}$

where $p : X(\vec{\alpha}) \to A(\vec{\alpha}) \in P$.

Binary-right:

$$\frac{w : [X, len, lr, g]}{w + in(A, len - len_B) + log(p) : [B, len_B, lr_B, g_B]}$$

Binary-left:

$$\frac{w : [X, len, lr, g]}{w + in(B, len - len_A) + log(p) : [A, len_A, lr_A, g_A]}$$

where, for both rules, $p : X(\vec{\alpha}) \to A(\vec{\alpha_A})B(\vec{\alpha_B}) \in P$.

Figure 9: SX estimate with length, LR, gaps

4.3 SX with LR, Gaps, Length

In order to further decrease the space complexity, we can simplify the previous estimate by subsuming the two lengths $left$ and $right$ in a single length lr. I.e., the items now have the form $[X, len, lr, gaps]$ with $X \in N$, $len + lr + gaps \le len_{max}$, $len \ge dim(X)$, $gaps \ge dim(X) - 1$.

The computation is given in Fig. 9. Again, we define $left_A$, $gaps_A$, $right_A$ and $gaps_B$, $right_B$ for a rule $X(\vec{\alpha}) \to A(\vec{\alpha_A})B(\vec{\alpha_B})$ as above. The side conditions are as follows: For *Binary-right* to apply, the following constraints must be satisfied: a) $len + lr + g = len_B + lr_B + g_B$, b) $lr < lr_B$, and c) $g_B \ge gaps_A$. For *Binary-left* to apply, the following must hold: a) $len + lr + g = len_A + lr_A + g_A$, b) if $right_B = 0$ then $lr = lr_A$, else $lr < lr_A$ and c) $g_A \ge gaps_B$.

The SX-estimate for a sentence length n and for some predicate C with a span $\vec{\rho} = \langle\langle l_1, r_1\rangle, \ldots, \langle l_{dim(C)}, r_{dim(C)}\rangle\rangle$ where $len = \Sigma_{i=1}^{dim(C)}(r_i - l_i)$ and $r = n - r_{dim(C)}$ is then the maximal weight of $[C, len, l_1 + r, n - len - l_1 - r]$.

5 Evaluation

The goal of our evaluation of our parser is to show that, firstly, reasonable parser speed can be achieved and, secondly, the parser output is of promising quality.

5.1 Data

Our data source is the German NeGra treebank (Skut et al., 1997). In a preprocessing step, following common practice (Kübler and Penn, 2008), we attach punctuation (not included in the NeGra annotation) as follows: In a first pass, using heuristics, we attach punctuation as high as possible while avoiding to introduce new crossing branches. In a second pass, parentheses and quotation marks preferably attach to the same node. Grammatical function labels on the edges are discarded.

We create data sets of different sizes in order to see how the size of the training set relates to the gain using context summary estimates and to the output quality of the parser. The first set uses the first 4000 sentences and the second one all sentences of NeGra. Due to memory limitations, in both sets, we limit ourselves to sentences of a maximal length of 25 words. We use the first 90% of both sets as training set and the remaining 10% as test set. Tab. 1 shows the resulting sizes.

	NeGra-small		NeGra	
	training	test	training	test
size	2839	316	14858	1651

Table 1: Test and training sets

5.2 Treebank Grammar Extraction

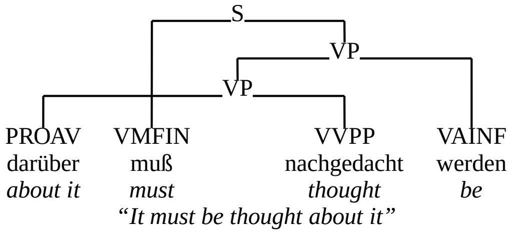


Figure 10: A sample tree from NeGra

As already mentioned, in NeGra, discontinuous phrases are annotated with crossing branches (see Fig. 10 for an example with two discontinuous VPs). Such discontinuities can be straightforwardly modelled with LCFRS. We use the algorithm from Maier and Søgaard (2008) to extract LCFRS rules from NeGra and TIGER. It first creates rules of the form $P(a) \to \varepsilon$ for each preterminal P dominating some terminal a. Then for all other nonterminals A_0 with the children $A_1 \cdots A_m$, a clause $A_0 \to A_1 \cdots A_m$ is created. The arguments of the $A_1 \cdots A_m$ are single variables where the number of arguments is the number of discontinuous parts in the yield of a predicate. The arguments of A_0 are concatenations of these variables that describe how the discontinuous parts of the yield of A_0 are obtained from the yields of its daughters. Different occurrences of the same non-terminal, only with different fan-outs, are distinguished by corresponding subscripts. Note that this extraction algorithm yields only *monotone* LCFRS (equivalent to ordered simple RCG). See Maier and Søgaard (2008) for further details. For Fig. 10, we obtain for instance the rules in Fig. 11.

PROAV(Darüber) $\rightarrow \varepsilon$ VMFIN(muß) $\rightarrow \varepsilon$
VVPP(nachgedacht) $\rightarrow \varepsilon$ VAINF(werden) $\rightarrow \varepsilon$
S$_1(X_1X_2X_3) \rightarrow$ VP$_2(X_1, X_3)$ VMFIN(X_2)
VP$_2(X_1, X_2X_3) \rightarrow$ VP$_2(X_1, X_2)$ VAINF(X_3)
VP$_2(X_1, X_2) \rightarrow$ PROAV(X_1) VVPP(X_2)

Figure 11: LCFRS rules for the tree in Fig. 10

5.3 Binarization and Markovization

Before parsing, we binarize the extracted LCFRS. For this we first apply Collins-style head rules, based on the rules the Stanford parser (Klein and Manning, 2003b) uses for NeGra, to mark the resp. head daughters of all non-terminal nodes. Then, we reorder the RHSs such that the sequence γ of elements to the right of the head daughter is reversed and moved to the beginning of the RHS. We then perform a binarization that proceeds from left to right. The binarization works like the transformation into Chomsky Normal Form for CFGs in the sense that for RHSs longer than 2, we introduce a new non-terminal that covers the RHS without the first element. The rightmost new rule, which covers the head daughter, is binarized to unary. We do not use a unique new non-terminal for every new rule. Instead, to the new symbols introduced during the binarization (VP$_{bin}$ in the example), a variable number of symbols from the vertical and horizontal context of the original rule is added in order to achieve *markovization*. Following the literature, we call the respective quantities v and h. For reasons of space we restrict ourselves here to the example in Fig. 12. Refer to Maier and Kallmeyer (2010) for a detailed presentation of the binarization and markovization.

The probabilities are then computed based on the rule frequencies in the transformed treebank, using a Maximum Likelihood estimator.

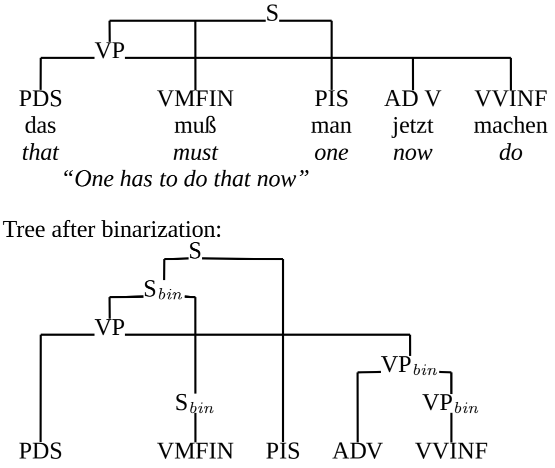


Figure 12: Sample binarization

5.4 Evaluation of Parsing Results

In order to assess the quality of the output of our parser, we choose an EVALB-style metric, i.e., we compare phrase boundaries. In the context of LCFRS, we compare sets of items $[A, \vec{\rho}]$ that characterize the span of a non-terminal A in a derivation tree. One set is obtained from the parser output, and one from the corresponding treebank trees. Using these item sets, we compute labeled and unlabeled recall (LR/UR), precision (LP/UP), and the F_1 measure (LF_1/UF_1). Note that if $k = 1$, our metric is identical to its PCFG equivalent.We are aware of the recent discussion about the shortcomings of EVALB. A discussion of this issue is presented in Maier (2010).

5.5 Experiments

In all experiments, we provide the parser with gold part-of-speech tags. For the experiments with *NeGra-small*, the parser is given the markovization settings $v = 1$ and $h = 1$. We compare the parser performance without estimates (OFF) with its performance with the estimates described in 4.2 (SIMPLE) and 4.3 (LR). Tab. 2 shows the results. Fig. 13 shows the number of items produced by the parser, indicating that the estimates have the desired effect of preventing unnecessary items from being produced. Note that it is even the case that the parser produces less items for the big set with LR than for the small set without estimate.

We can see that the estimates lead to a slightly

	OFF	SIMPLE	LR
UP/UR	72.29/72.40	70.49/71.81	72.10/72.60
UF_1	72.35	71.14	72.35
LP/LR	68.31/68.41	64.93/66.14	67.35/66.14
LF_1	68.36	65.53	65.53
Parsed	313 (99.05%)	313 (99.05%)	313 (99.05%)

Table 2: Experiments with *NeGra-small*

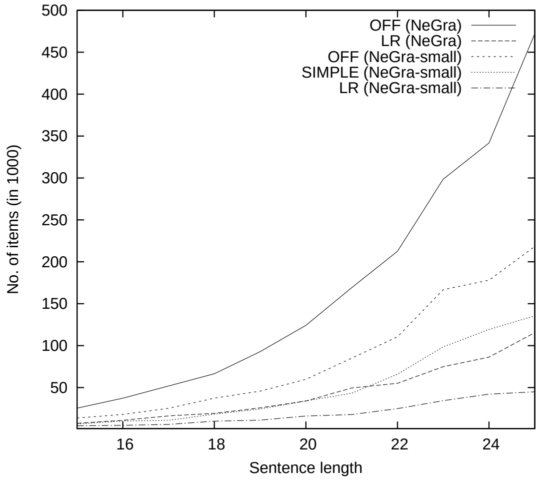


Figure 13: Items produced by the parser

lower F-score. However, while the losses in terms of F_1 are small, the gains in parsing time are substantial, as Fig. 13 shows.

Tab. 3 shows the results of experiments with *NeGra*, with the markovization settings $v = 2$ and $h = 1$ which have proven to be successful for PCFG parsing of NeGra (Rafferty and Manning, 2008). Unfortunately, due to memory restrictions, we were not able to compute SIMPLE for the large data set.[1] Resp. LR, the findings are comparable to the ones for *NeGra-short*. The speedup is paid with a lower F_1.

	OFF	LR
UP/UR	76.89/77.35	75.22/75.99
UF_1	77.12	75.60
LP/LR	73.03/73.46	70.98/71.70
LF_1	73.25	71.33
Parsed	1642 (99.45%)	1642 (99.45%)

Table 3: Experiments with *NeGra*

Our results are not directly comparable with PCFG parsing results, since LCFRS parsing is a harder task. However, since the EVALB metric coincides for constituents without crossing branches, in order to place our results in the context of previous work on parsing NeGra, we cite some of the results from the literature which were obtained using PCFG parsers[2]: Kübler (2005) (Tab. 1, plain PCFG) obtains 69.4, Dubey and Keller (2003) (Tab. 5, sister-head PCFG model) 71.12, Rafferty and Manning (2008) (Tab. 2, Stanford parser with markovization $v = 2$ and $h = 1$) 77.2, and Petrov and Klein (2007) (Tab. 1, Berkeley parser) 80.1. Plaehn (2004) obtains 73.16 Labeled F_1 using Probabilistic Discontinuous Phrase Structure Grammar (DPSG), albeit only on sentences with a length of up to 15 words. On those sentences, we obtain 81.27.

The comparison shows that our system delivers competitive results. Additionally, when comparing this to PCFG parsing results, one has to keep in mind that LCFRS parse trees contain non-context-free information about discontinuities. Therefore, a correct parse with our grammar is actually better than a correct CFG parse, evaluated with respect to a transformation of NeGra into a context-free treebank where precisely this information gets lost.

6 Conclusion

We have presented the first parser for unrestricted Probabilistic Linear Context-Free Rewriting Systems (PLCFRS), implemented as a CYK parser with weighted deductive parsing. To speed up parsing, we use context summary estimates for parse items. An evaluation on the NeGra treebank, both in terms of output quality and speed, shows that data-driven parsing using PLCFRS is feasible. Already in this first attempt with a straightforward binarization, we obtain results that are comparable to state-of-the-art PCFG results in terms of F_1, while yielding parse trees that are richer than context-free trees since they describe discontinuities. Therefore, our approach demonstrates convincingly that PLCFRS is a natural and tractable alternative for data-driven parsing which takes non-local dependencies into consideration.

[1] SIMPLE also proved to be infeasible to compute for the small set for the markovization settings $v = 2$ and $h = 1$ due to the greatly increased label set with this settings.

[2] Note that these results were obtained on sentences with a length of ≤ 40 words and that those parser possibly would deliver better results if tested on our test set.

References

Boullier, Pierre. 1998. A Proposal for a Natural Language Processing Syntactic Backbone. Technical Report 3342, INRIA.

Boyd, Adriane. 2007. Discontinuity revisited: An improved conversion to context-free representations. In *The Linguistic Annotation Workshop at ACL 2007*.

Charniak, Eugene and Sharon A. Caraballo. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24.

Collins, Michael. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

Dubey, Amit and Frank Keller. 2003. Probabilistic parsing for German using sisterhead dependencies. In *Proceedings of ACL*.

Gómez-Rodríguez, Carlos, Marco Kuhlmann, Giorgio Satta, and David Weir. 2009. Optimal reduction of rule length in linear context-free rewriting systems. In *Proceedings of NAACL-HLT*.

Hockenmaier, Julia. 2003. *Data and models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Johnson, Mark. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of ACL*.

Kato, Yuki, Hiroyuki Seki, and Tadao Kasami. 2006. Stochastic multiple context-free grammar for rna pseudoknot modeling. In *Proceedings of TAG+8*.

Klein, Dan and Christopher D. Manning. 2003a. A* Parsing: Fast Exact Viterbi Parse Selection. In *Proceedings of NAACL-HLT*.

Klein, Dan and Christopher D. Manning. 2003b. Fast exact inference with a factored model for natural language parsing. In *In Advances in Neural Information Processing Systems 15 (NIPS)*.

Kübler, Sandra and Gerald Penn, editors. 2008. *Proceedings of the Workshop on Parsing German at ACL 2008*.

Kübler, Sandra. 2005. How do treebank annotation schemes influence parsing results? Or how not to compare apples and oranges. In *Proceedings of RANLP 2005*.

Kuhlmann, Marco and Giorgio Satta. 2009. Treebank grammar techniques for non-projective dependency parsing. In *Proceedings of EACL*.

Levy, Roger and Christopher D. Manning. 2004. Deep dependencies from context-free statistical parsers: correcting the surface dependency approximation. In *Proceedings of ACL*.

Maier, Wolfgang and Laura Kallmeyer. 2010. Discontinuity and non-projectivity: Using mildly context-sensitive formalisms for data-driven parsing. In *Proceedings of TAG+10*.

Maier, Wolfgang and Timm Lichte. 2009. Characterizing Discontinuity in Constituent Treebanks. In *Proceedings of Formal Grammar 2009*.

Maier, Wolfgang and Anders Søgaard. 2008. Treebanks and mild context-sensitivity. In *Proceedings of Formal Grammar 2008*.

Maier, Wolfgang. 2010. Direct parsing of discontinuous constituents in german. In *Proceedings of the SPMRL workshop at NAACL HLT 2010*.

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of HLT*.

Nederhof, Mark-Jan. 2003. Weighted Deductive Parsing and Knuth's Algorithm. *Computational Linguistics*, 29(1).

Petrov, Slav and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of HLT-NAACL 2007*.

Plaehn, Oliver. 2004. Computing the most probable parse for a discontinuous phrase-structure grammar. In *New developments in parsing technology*. Kluwer.

Rafferty, Anna and Christopher D. Manning, 2008. *Parsing Three German Treebanks: Lexicalized and Unlexicalized Baselines*. In Kübler and Penn (2008).

Seki, Hiroyuki, Takahashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2).

Skut, Wojciech, Brigitte Krenn, Thorten Brants, and Hans Uszkoreit. 1997. An Annotation Scheme for Free Word Order Languages. In *Proceedings of ANLP*.

Vijay-Shanker, K., David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of ACL*.

Learning to Predict Readability using Diverse Linguistic Features

Rohit J. Kate[1] **Xiaoqiang Luo**[2] **Siddharth Patwardhan**[2] **Martin Franz**[2]
Radu Florian[2] **Raymond J. Mooney**[1] **Salim Roukos**[2] **Chris Welty**[2]
[1]Department of Computer Science
The University of Texas at Austin
{rjkate,mooney}@cs.utexas.edu
[2]IBM Watson Research Center
{xiaoluo,spatward,franzm,raduf,roukos,welty}@us.ibm.com

Abstract

In this paper we consider the problem of building a system to predict readability of natural-language documents. Our system is trained using diverse features based on syntax and language models which are generally indicative of readability. The experimental results on a dataset of documents from a mix of genres show that the predictions of the learned system are more accurate than the predictions of naive human judges when compared against the predictions of linguistically-trained expert human judges. The experiments also compare the performances of different learning algorithms and different types of feature sets when used for predicting readability.

1 Introduction

An important aspect of a document is whether it is easily processed and understood by a human reader as intended by its writer, this is termed as the document's *readability*. Readability involves many aspects including grammaticality, conciseness, clarity, and lack of ambiguity. Teachers, journalists, editors, and other professionals routinely make judgements on the readability of documents. We explore the task of learning to automatically judge the readability of natural-language documents.

In a variety of applications it would be useful to be able to automate readability judgements. For example, the results of a web-search can be ordered taking into account the readability of the retrieved documents thus improving user satisfaction. Readability judgements can also be used for automatically grading essays, selecting instructional reading materials, etc. If documents are generated by machines, such as summarization or machine translation systems, then they are prone to be less readable. In such cases, a readability measure can be used to automatically filter out documents which have poor readability. Even when the intended consumers of text are machines, for example, information extraction or knowledge extraction systems, a readability measure can be used to filter out documents of poor readability so that the machine readers will not extract incorrect information because of ambiguity or lack of clarity in the documents.

As part of the DARPA Machine Reading Program (MRP), an evaluation was designed and conducted for the task of rating documents for readability. In this evaluation, 540 documents were rated for readability by both experts and novice human subjects. Systems were evaluated based on whether they were able to match expert readability ratings better than novice raters. Our system learns to match expert readability ratings by employing regression over a set of diverse linguistic features that were deemed potentially relevant to readability. Our results demonstrate that a rich combination of features from syntactic parsers, language models, as well as lexical statistics all contribute to accurately predicting expert human readability judgements. We have also considered the effect of different genres in predicting readability and how the genre-specific language models can be exploited to improve the readability predictions.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 546–554,
Beijing, August 2010

2 Related Work

There is a significant amount of published work on a related problem: predicting the reading difficulty of documents, typically, as the school grade-level of the reader from grade 1 to 12. Some early methods measure simple characteristics of documents like average sentence length, average number of syllables per word, etc. and combine them using a linear formula to predict the grade level of a document, for example FOG (Gunning, 1952), SMOG (McLaughlin, 1969) and Flesh-Kincaid (Kincaid et al., 1975) metrics. These methods do not take into account the content of the documents. Some later methods use pre-determined lists of words to determine the grade level of a document, for example the Lexile measure (Stenner et al., 1988), the Fry Short Passage measure (Fry, 1990) and the Revised Dale-Chall formula (Chall and Dale, 1995). The word lists these methods use may be thought of as very simple language models. More recently, language models have been used for predicting the grade level of documents. Si and Callan (2001) and Collins-Thompson and Callan (2004) train unigram language models to predict grade levels of documents. In addition to language models, Heilman et al. (2007) and Schwarm and Ostendorf (2005) also use some syntactic features to estimate the grade level of texts.

Pitler and Nenkova (2008) consider a different task of predicting text quality for an educated adult audience. Their system predicts readability of texts from Wall Street Journal using lexical, syntactic and discourse features. Kanungo and Orr (2009) consider the task of predicting readability of web summary snippets produced by search engines. Using simple surface level features like the number of characters and syllables per word, capitalization, punctuation, ellipses etc. they train a regression model to predict readability values.

Our work differs from this previous research in several ways. Firstly, the task we have considered is different, we predict the readability of general documents, not their grade level. The documents in our data are also not from any single domain, genre or reader group, which makes our task more general. The data includes human written as well as machine generated documents. The task and the data has been set this way because it is aimed at filtering out documents of poor quality for later processing, like for extracting machine-processable knowledge from them. Extracting knowledge from openly found text, such as from the internet, is becoming popular but the quality of text found “in the wild”, like found through searching the internet, vary considerably in quality and genre. If the text is of poor readability then it is likely to lead to extraction errors and more problems downstream. If the readers are going to be humans instead of machines, then also it is best to filter out poorly written documents. Hence identifying readability of general text documents coming from various sources and genres is an important task. We are not aware of any other work which has considered such a task.

Secondly, we note that all of the above approaches that use language models train a language model for each difficulty level using the training data for that level. However, since the amount of training data annotated with levels is limited, they can not train higher-order language models, and most just use unigram models. In contrast, we employ more powerful language models trained on large quantities of generic text (which is not from the training data for readability) and use various features obtained from these language models to predict readability. Thirdly, we use a more sophisticated combination of linguistic features derived from various syntactic parsers and language models than any previous work. We also present ablation results for different sets of features. Fourthly, given that the documents in our data are not from a particular genre but from a mix of genres, we also train genre-specific language models and show that including these as features improves readability predictions. Finally, we also show comparison between various machine learning algorithms for predicting readability, none of the previous work compared learning algorithms.

3 Readability Data

The readability data was collected and released by LDC. The documents were collected from the following diverse sources or genres: newswire/newspaper text, weblogs, newsgroup posts, manual transcripts, machine translation output, closed-caption transcripts and Wikipedia articles. Documents for newswire, machine translation and closed captioned genres were collected automatically by first forming a candidate pool from a single collection stream and then randomly selecting documents. Documents for weblogs, newsgroups and manual transcripts were also collected in the same way but were then reviewed by humans to make sure they were not simply spam articles or something objectionable. The Wikipedia articles were collected manually, by searching through a data archive or the live web, using keyword and other search techniques. Note that the information about genres of the documents is not available during testing and hence was not used when training our readability model.

A total of 540 documents were collected in this way which were uniformly distributed across the seven genres. Each document was then judged for its readability by eight expert human judges. These expert judges are native English speakers who are language professionals and who have specialized training in linguistic analysis and annotation, including the machine translation post-editing task. Each document was also judged for its readability by six to ten naive human judges. These non-expert (naive) judges are native English speakers who are not language professionals (e.g. editors, writers, English teachers, linguistic annotators, etc.) and have no specialized language analysis or linguistic annotation training. Both expert and naive judges provided readability judgments using a customized web interface and gave a rating on a 5-point scale to indicate how readable the passage is (where 1 is lowest and 5 is highest readability) where readability is defined as a subjective judgment of how easily a reader can extract the information the writer or speaker intended to convey.

4 Readability Model

We want to answer the question whether a machine can accurately estimate readability as judged by a human. Therefore, we built a machine-learning system that predicts the readability of documents by training on expert human judgements of readability. The evaluation was then designed to compare how well machine and naive human judges predict expert human judgements. In order to make the machine's predicted score comparable to a human judge's score (details about our evaluation metrics are in Section 6.1), we also restricted the machine scores to integers. Hence, the task is to predict an integer score from 1 to 5 that measures the readability of the document.

This task could be modeled as a multi-class classification problem treating each integer score as a separate class, as done in some of the previous work (Si and Callan, 2001; Collins-Thompson and Callan, 2004). However, since the classes are numerical and not unrelated (for example, the score 2 is in between scores 1 and 3), we decided to model the task as a regression problem and then round the predicted score to obtain the closest integer value. Preliminary results verified that regression performed better than classification. Heilman et al. (2008) also found that it is better to treat the readability scores as ordinal than as nominal. We take the average of the expert judge scores for each document as its gold-standard score. Regression was also used by Kanungo and Orr (2009), although their evaluation did not constrain machine scores to be integers.

We tested several regression algorithms available in the Weka[1] machine learning package, and in Section 6.2 we report results for several which performed best. The next section describes the numerically-valued features that we used as input for regression.

5 Features for Predicting Readability

Good input features are critical to the success of any regression algorithm. We used three main categories of features to predict readability: syntactic features, language-model features, and lexical features, as described below.

5.1 Features Based on Syntax

Many times, a document is found to be unreadable due to unusual linguistic constructs or ungram-

[1] http://www.cs.waikato.ac.nz/ml/weka/

matical language that tend to manifest themselves in the syntactic properties of the text. Therefore, syntactic features have been previously used (Bernth, 1997) to gauge the "clarity" of written text, with the goal of helping writers improve their writing skills. Here too, we use several features based on syntactic analyses. Syntactic analyses are obtained from the Sundance shallow parser (Riloff and Phillips, 2004) and from the English Slot Grammar (ESG) (McCord, 1989).

Sundance features: The Sundance system is a rule-based system that performs a shallow syntactic analysis of text. We expect that this analysis over readable text would be "well-formed", adhering to grammatical rules of the English language. Deviations from these rules can be indications of unreadable text. We attempt to capture such deviations from grammatical rules through the following Sundance features computed for each text document: proportion of sentences with no verb phrases, average number of clauses per sentence, average sentence length in tokens, average number of noun phrases per sentence, average number of verb phrases per sentence, average number of prepositional phrases per sentence, average number of phrases (all types) per sentence and average number of phrases (all types) per clause.

ESG features: ESG uses slot grammar rules to perform a deeper linguistic analysis of sentences than the Sundance system. ESG may consider several different interpretations of a sentence, before deciding to choose one over the other interpretations. Sometimes ESG's grammar rules fail to produce a single complete interpretation of a sentence, in which case it generates partial parses. This typically happens in cases when sentences are ungrammatical, and possibly, less readable. Thus, we use the proportion of such incomplete parses within a document as a readability feature. In case of extremely short documents, this proportion of incomplete parses can be misleading. To account for such short documents, we introduce a variation of the above incomplete parse feature, by weighting it with a log factor as was done in (Riloff, 1996; Thelen and Riloff, 2002).

We also experimented with some other syntactic features such as average sentence parse scores from Stanford parser and an in-house maximum entropy statistical parer, average constituent scores etc., however, they slightly degraded the performance in combination with the rest of the features and hence we did not include them in the final set. One possible explanation could be that averaging diminishes the effect of low scores caused by ungrammaticality.

5.2 Features Based on Language Models

A probabilistic language model provides a prediction of how likely a given sentence was generated by the same underlying process that generated a corpus of training documents. In addition to a general n-gram language model trained on a large body of text, we also exploit language models trained to recognize specific "genres" of text. If a document is translated by a machine, or casually produced by humans for a weblog or newsgroup, it exhibits a character that is distinct from documents that go through a dedicated editing process (e.g., newswire and Wikipedia articles). Below we describe features based on generic as well as genre-specific language models.

Normalized document probability: One obvious proxy for readability is the score assigned to a document by a generic language model (LM). Since the language model is trained on well-written English text, it penalizes documents deviating from the statistics collected from the LM training documents. Due to variable document lengths, we normalize the document-level LM score by the number of words and compute the normalized document probability $NP(\mathcal{D})$ for a document $\mathcal{D}$ as follows:

$$NP(\mathcal{D}) = \big(P(\mathcal{D}|\mathcal{M})\big)^{\frac{1}{|\mathcal{D}|}}, \qquad (1)$$

where $\mathcal{M}$ is a general-purpose language model trained on clean English text, and $|\mathcal{D}|$ is the number of words in the document $\mathcal{D}$.

Perplexities from genre-specific language models: The usefulness of LM-based features in categorizing text (McCallum and Nigam, 1998; Yang and Liu, 1999) and evaluating readability (Collins-Thompson and Callan, 2004; Heilman et al., 2007) has been investigated in previous work. In our experiments, however, since documents were acquired through several different channels, such as machine translation or web logs, we also build models that try to predict the genre of a document. Since the genre information for many English documents is readily available, we trained a series of genre-specific 5-gram LMs using the modified Kneser-Ney smoothing (Kneser and Ney, 1995; Stanley and Goodman, 1996). Table 1 contains a list of a base LM and genre-specific LMs.

Given a document $\mathcal{D}$ consisting of tokenized word sequence $\{w_i : i = 1, 2, \cdots, |\mathcal{D}|\}$, its perplexity $L(\mathcal{D}|\mathcal{M}_j)$ with respect to a LM $\mathcal{M}_j$ is computed as:

$$L(\mathcal{D}|\mathcal{M}_j) = \ e^{\left(-\frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|}\log P(w_i|h_i;\mathcal{M}_j)\right)}, \quad (2)$$

where $|\mathcal{D}|$ is the number of words in $\mathcal{D}$ and h_i are the history words for w_i, and $P(w_i|h_i;\mathcal{M}_j)$ is the probability $\mathcal{M}_j$ assigns to w_i, when it follows the history words h_i.

Posterior perplexities from genre-specific language models: While perplexities computed from genre-specific LMs reflect the absolute probability that a document was generated by a specific model, a model's *relative* probability compared to other models may be a more useful feature. To this end, we also compute the posterior perplexity defined as follows. Let $\mathcal{D}$ be a document, $\{\mathcal{M}_i\}_{i=1}^{G}$ be G genre-specific LMs, and $L(\mathcal{D}|\mathcal{M}_i)$ be the perplexity of the document $\mathcal{D}$ with respect to $\mathcal{M}_i$, then the posterior perplexity, $R(\mathcal{M}_i|\mathcal{D})$, is defined as:

$$R(\mathcal{M}_i|\mathcal{D}) = \frac{L(\mathcal{D}|\mathcal{M}_i)}{\sum_{j=1}^{G} L(\mathcal{D}|\mathcal{M}_j)}. \quad (3)$$

We use the term "posterior" because if a uniform prior is adopted for $\{\mathcal{M}_i\}_{i=1}^{G}$, $R(\mathcal{M}_i|\mathcal{D})$ can be interpreted as the posterior probability of the genre LM $\mathcal{M}_i$ given the document $\mathcal{D}$.

5.3 Lexical Features

The final set of features involve various lexical statistics as described below.

Out-of-vocabulary (OOV) rates: We conjecture that documents containing typographical errors (e.g., for closed-caption and web log documents) may receive low readability ratings. Therefore, we compute the OOV rates of a document with respect to the various LMs shown in Table 1. Since modern LMs often have a very large vocabulary, to get meaningful OOV rates, we truncate the vocabularies to the top (i.e., most frequent) 3000 words. For the purpose of OOV computation, a document $\mathcal{D}$ is treated as a sequence of tokenized words $\{w_i : i = 1, 2, \cdots, |\mathcal{D}|\}$. Its OOV rate with respect to a (truncated) vocabulary $\mathcal{V}$ is then:

$$OOV(\mathcal{D}|\mathcal{V}) = \frac{\sum_{i=1}^{\mathcal{D}} I(w_i \notin \mathcal{V})}{|\mathcal{D}|}, \quad (4)$$

where $I(w_i \notin \mathcal{V})$ is an indicator function taking value 1 if w_i is not in $\mathcal{V}$, and 0 otherwise.

Ratio of function words: A characteristic of documents generated by foreign speakers and machine translation is a failure to produce certain function words, such as "the," or "of." So we predefine a small set of function words (mainly English articles and frequent prepositions) and compute the ratio of function words over the total number words in a document:

$$RF(\mathcal{D}) = \frac{\sum_{i=1}^{\mathcal{D}} I(w_i \in \mathcal{F})}{|\mathcal{D}|}, \quad (5)$$

where $I(w_i \in \mathcal{F})$ is 1 if w_i is in the set of function words $\mathcal{F}$, and 0 otherwise.

Ratio of pronouns: Many foreign languages that are source languages of machine-translated documents are pronoun-drop languages, such as Arabic, Chinese, and romance languages. We conjecture that the pronoun ratio may be a good indicator whether a document is translated by machine or produced by humans, and for each document, we first run a POS tagger, and then compute the ratio of pronouns over the number of words in the document:

$$RP(\mathcal{D}) = \frac{\sum_{i=1}^{\mathcal{D}} I(POS(w_i) \in \mathcal{P})}{|\mathcal{D}|}, \quad (6)$$

where $I(POS(w_i) \in \mathcal{F})$ is 1 if the POS tag of w_i is in the set of pronouns, $\mathcal{P}$, and 0 otherwise.

Fraction of known words: This feature measures the fraction of words in a document that occur either in an English dictionary or a gazetteer of names of people and locations.

6 Experiments

This section describes the evaluation methodology and metrics and presents and discusses our

Genre	Training Size(M tokens)	Data Sources
base	5136.8	mostly LDC's GigaWord set
NW	143.2	newswire subset of base
NG	218.6	newsgroup subset of base
WL	18.5	weblog subset of base
BC	1.6	broadcast conversation subset of base
BN	1.1	broadcast news subset of base
wikipedia	2264.6	Wikipedia text
CC	0.1	closed caption
ZhEn	79.6	output of Chinese to English Machine Translation
ArEn	126.8	output of Arabic to English Machine Translation

Table 1: Genre-specific LMs: the second column contains the number of tokens in LM training data (in million tokens).

experimental results. The results of the official evaluation task are also reported.

6.1 Evaluation Metric

The evaluation process for the DARPA MRP readability test was designed by the evaluation team led by SAIC. In order to compare a machine's predicted readability score to those assigned by the expert judges, the Pearson correlation coefficient was computed. The mean of the expert-judge scores was taken as the gold-standard score for a document.

To determine whether the machine predicts scores closer to the expert judges' scores than what an average naive judge would predict, a sampling distribution representing the underlying novice performance was computed. This was obtained by choosing a random naive judge for every document, calculating the Pearson correlation coefficient with the expert gold-standard scores and then repeating this procedure a sufficient number of times (5000). The upper critical value was set at 97.5% confidence, meaning that if the machine performs better than the upper critical value then we reject the null hypothesis that machine scores and naive scores come from the same distribution and conclude that the machine performs significantly better than naive judges in matching the expert judges.

6.2 Results and Discussion

We evaluated our readability system on the dataset of 390 documents which was released earlier during the training phase of the evaluation task. We used stratified 13-fold cross-validation in which the documents from various genres in each fold was distributed in roughly the same proportion as in the overall dataset. We first conducted experiments to test different regression algorithms using all the available features. Next, we ablated various feature sets to determine how much each feature set was contributing to making accurate readability judgements. These experiments are described in the following subsections.

Algorithm	Correlation
Bagged Decision Trees	0.8173
Decision Trees	0.7260
Linear Regression	0.7984
SVM Regression	0.7915
Gaussian Process Regression	0.7562
Naive Judges	
Upper Critical Value	0.7015
Distribution Mean	0.6517
Baselines	
Uniform Random	0.0157
Proportional Random	-0.0834

Table 2: Comparing different algorithms on the readability task using 13-fold cross-validation on the 390 documents using all the features. Exceeding the upper critical value of the naive judges' distribution indicates statistically significantly better predictions than the naive judges.

6.2.1 Regression Algorithms

We used several regression algorithms available in the Weka machine learning package and Table 2 shows the results obtained. The default values

Feature Set	Correlation
Lexical	0.5760
Syntactic	0.7010
Lexical + Syntactic	0.7274
Language Model based	0.7864
All	0.8173

Table 3: Comparison of different linguistic feature sets.

in Weka were used for all parameters, changing these values did not show any improvement. We used decision tree (reduced error pruning (Quinlan, 1987)) regression, decision tree regression with bagging (Breiman, 1996), support vector regression (Smola and Scholkopf, 1998) using polynomial kernel of degree two,[2] linear regression and Gaussian process regression (Rasmussen and Williams, 2006). The distribution mean and the upper critical values of the correlation coefficient distribution for the naive judges are also shown in the table.

Since they are above the upper critical value, all algorithms predicted expert readability scores significantly more accurately than the naive judges. Bagged decision trees performed slightly better than other methods. As shown in the following section, ablating features affects predictive accuracy much more than changing the regression algorithm. Therefore, on this task, the choice of regression algorithm was not very critical once good readability features are used. We also tested two simple baseline strategies: predicting a score uniformly at random, and predicting a score proportional to its frequency in the training data. As shown in the last two rows of Table 2, these baselines perform very poorly, verifying that predicting readability on this dataset as evaluated by our evaluation metric is not trivial.

6.2.2 Ablations with Feature Sets

We evaluated the contributions of different feature sets through ablation experiments. Bagged decision-tree was used as the regression algorithm in all of these experiments. First we compared syntactic, lexical and language-model based features as described in Section 5, and Table 3 shows the results. The language-model feature set performs the best, but performance improves when it is combined with the remaining features. The lexical feature set by itself performs the worst, even below the naive distribution mean (shown in Table 2); however, when combined with syntactic features it performs well.

In our second ablation experiment, we compared the performance of genre-independent and genre-based features. Since the genre-based features exploit knowledge of the genres of text used in the MRP readability corpus, their utility is somewhat tailored to this specific corpus. Therefore, it is useful to evaluate the performance of the system when genre information is not exploited. Of the lexical features described in subsection 5.3, the ratio of function words, ratio of pronoun words and all of the out-of-vocabulary rates except for the base language model are genre-based features. Out of the language model features described in the Subsection 5.2, all of the perplexities except for the base language model and all of the posterior perplexities[3] are genre-based features. All of the remaining features are genre-independent. Table 4 shows the results comparing these two feature sets. The genre-based features do well by themselves but the rest of the features help further improve the performance. While the genre-independent features by themselves do not exceed the upper critical value of the naive judges' distribution, they are very close to it and still outperform its mean value. These results show that for a dataset like ours, which is composed of a mix of genres that themselves are indicative of readability, features that help identify the genre of a text improve performance significantly.[4] For applications mentioned in the introduction and related work sections, such as filtering less readable documents from web-search, many of the input documents could come from some of the common genres considered in our dataset.

In our final ablation experiment, we evaluated

[2] Polynomial kernels with other degrees and RBF kernel performed worse.

[3] Base model for posterior perplexities is computed using other genre-based LMs (equation 3) hence it can not be considered genre-independent.

[4] We note that none of the genre-based features were trained on supervised readability data, but were trained on readily-available large unannotated corpora as shown in Table 1.

Feature Set	Correlation
Genre-independent	0.6978
Genre-based	0.7749
All	0.8173

Table 4: Comparison of genre-independent and genre-based feature sets.

Feature Set	By itself	Ablated from All
Sundance features	0.5417	0.7993
ESG features	0.5841	0.8118
Perplexities	0.7092	0.8081
Posterior perplexities	0.7832	0.7439
Out-of-vocabulary rates	0.3574	0.8125
All	0.8173	-

Table 5: Ablations with some individual feature sets.

the contribution of various individual feature sets. Table 5 shows that posterior perplexities perform the strongest on their own, but without them, the remaining features also do well. When used by themselves, some feature sets perform below the naive judges' distribution mean, however, removing them from the rest of the feature sets degrades the performance. This shows that no individual feature set is critical for good performance but each further improves the performance when added to the rest of the feature sets.

6.3 Official Evaluation Results

An official evaluation was conducted by the evaluation team SAIC on behalf of DARPA in which three teams participated including ours. The evaluation task required predicting the readability of 150 test documents using the 390 training documents. Besides the correlation metric, two additional metrics were used. One of them computed for a document the difference between the average absolute difference of the naive judge scores from the mean expert score and the absolute difference of the machine's score from the mean expert score. This was then averaged over all the documents. The other one was "target hits" which measured if the predicted score for a document fell within the width of the lowest and the highest expert scores for that document, and if so, computed a score inversely proportional to that width. The final target hits score was then computed by averaging it across all the documents. The upper critical values for these metrics were computed in a way analogous to that for the correlation metric which was described before. Higher score is better for all the three metrics. Table 6 shows the results of the evaluation. Our system performed favorably and always scored better than the upper critical value on each of the metrics. Its performance was in between the performance of the other two systems. The performances of the systems show that the correlation metric was the most difficult of the three metrics.

System	Correl.	Avg. Diff.	Target Hits
Our (A)	0.8127	0.4844	0.4619
System B	0.6904	0.3916	0.4530
System C	0.8501	0.5177	0.4641
Upper CV	0.7423	0.0960	0.3713

Table 6: Results of the systems that participated in the DARPA's readability evaluation task. The three metrics used were correlation, average absolute difference and target hits measured against the expert readability scores. The upper critical values are for the score distributions of naive judges.

7 Conclusions

Using regression over a diverse combination of syntactic, lexical and language-model based features, we built a system for predicting the readability of natural-language documents. The system accurately predicts readability as judged by linguistically-trained expert human judges and exceeds the accuracy of naive human judges. Language-model based features were found to be most useful for this task, but syntactic and lexical features were also helpful. We also found that for a corpus consisting of documents from a diverse mix of genres, using features that are indicative of the genre significantly improve the accuracy of readability predictions. Such a system could be used to filter out less readable documents for machine or human processing.

Acknowledgment

This research was funded by Air Force Contract FA8750-09-C-0172 under the DARPA Machine Reading Program.

References

Bernth, Arendse. 1997. Easyenglish: A tool for improving document quality. In *Proceedings of the fifth conference on Applied Natural Language Processing*, pages 159–165, Washington DC, April.

Breiman, Leo. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Chall, J.S. and E. Dale. 1995. *Readability Revisited: The New Dale-Chall Readability Formula*. Brookline Books, Cambridge, MA.

Collins-Thompson, Kevyn and James P. Callan. 2004. A language modeling approach to predicting reading difficulty. In *Proc. of HLT-NAACL 2004*, pages 193–200.

Fry, E. 1990. A readability formula for short passages. *Journal of Reading*, 33(8):594–597.

Gunning, R. 1952. *The Technique of Clear Writing*. McGraw-Hill, Cambridge, MA.

Heilman, Michael, Kevyn Collins-Thompson, Jamie Callan, and Maxine Eskenazi. 2007. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Proc. of NAACL-HLT 2007*, pages 460–467, Rochester, New York, April.

Heilman, Michael, Kevyn Collins-Thompson, and Maxine Eskenazi. 2008. An analysis of statistical models and features for reading difficulty prediction. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pages 71–79, Columbus, Ohio, June. Association for Computational Linguistics.

Kanungo, Tapas and David Orr. 2009. Predicting the readability of short web summaries. In *Proc. of WSDM 2009*, pages 202–211, Barcelona, Spain, February.

Kincaid, J. P., R. P. Fishburne, R. L. Rogers, and B.S. Chissom. 1975. Derivation of new readability formulas for navy enlisted personnel. Technical Report Research Branch Report 8-75, Millington, TN: Naval Air Station.

Kneser, Reinhard and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proc. of ICASSP-95*, pages 181–184.

McCallum, Andrew and Kamal Nigam. 1998. A comparison of event models for naive Bayes text classification. In *Papers from the AAAI-98 Workshop on Text Categorization*, pages 41–48, Madison, WI, July.

McCord, Michael C. 1989. Slot grammar: A system for simpler construction of practical natural language grammars. In *Proceedings of the International Symposium on Natural Language and Logic*, pages 118–145, May.

McLaughlin, G. H. 1969. Smog: Grading: A new readability formula. *Journal of Reading*, 12:639–646.

Pitler, Emily and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proc. of EMNLP 2008*, pages 186–195, Waikiki,Honolulu,Hawaii, October.

Quinlan, J. R. 1987. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234.

Rasmussen, Carl and Christopher Williams. 2006. *Gaussian Processes for Machine Leanring*. MIT Press, Cambridge, MA.

Riloff, E. and W. Phillips. 2004. An introduction to the Sundance and Autoslog systems. Technical Report UUCS-04-015, University of Utah School of Computing.

Riloff, Ellen. 1996. Automatically generating extraction patterns from untagged text. In *Proc. of 13th Natl. Conf. on Artificial Intelligence (AAAI-96)*, pages 1044–1049, Portland, OR.

Schwarm, Sarah E. and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proc. of ACL 2005*, pages 523–530, Ann Arbor, Michigan.

Si, Luo and James P. Callan. 2001. A statistical model for scientific readability. In *Proc. of CIKM 2001*, pages 574–576.

Smola, Alex J. and Bernhard Scholkopf. 1998. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, NeuroCOLT2.

Stanley, Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proc. of the 34th Annual Meeting of the Association for Computational Linguistics (ACL-96)*, pages 310–318.

Stenner, A. J., I. Horabin, D. R. Smith, and M. Smith. 1988. *The Lexile Framework*. Durham, NC: MetaMetrics.

Thelen, M. and E. Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proc. of EMNLP 2002*, Philadelphia, PA, July.

Yang, Yiming and Xin Liu. 1999. A re-examination of text cateogrization methods. In *Proc. of 22nd Intl. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 42–48, Berkeley, CA.

Value for Money: Balancing Annotation Effort, Lexicon Building and Accuracy for Multilingual WSD

Mitesh M. Khapra **Saurabh Sohoney** **Anup Kulkarni** **Pushpak Bhattacharyya**
Department of Computer Science and Engineering,
Indian Institute of Technology Bombay
{miteshk,saurabhsohoney,anup,pb}@cse.iitb.ac.in

Abstract

Sense annotation and lexicon building are costly affairs demanding prudent investment of resources. Recent work on multilingual WSD has shown that it is possible to leverage the annotation work done for WSD of one language (S_L) for another (T_L), by projecting Wordnet and sense marked corpus parameters of S_L to T_L. However, this work does not take into account the cost of manually cross-linking the words within aligned synsets. Further, it does not answer the question of *"Can better accuracy be achieved if a user is willing to pay additional money?"* We propose a measure for ***cost-benefit analysis*** which measures the *"value for money"* earned in terms of accuracy by investing in annotation effort and lexicon building. Two key ideas explored in this paper are (i) the use of ***probabilistic cross-linking model*** to reduce manual cross-linking effort and (ii) the use of ***selective sampling*** to inject a few training examples for hard-to-disambiguate words from the target language to boost the accuracy.

1 Introduction

Word Sense Disambiguation (WSD) is one of the most widely investigated problems of Natural Language Processing (NLP). Previous works have shown that supervised approaches to Word Sense Disambiguation which rely on sense annotated corpora (Ng and Lee, 1996; Lee et al., 2004) outperform unsupervised (Veronis, 2004) and knowledge based approaches (Mihalcea, 2005). However, creation of sense marked corpora has always remained a costly proposition, especially for some of the resource deprived languages.

To circumvent this problem, Khapra et al. (2009) proposed a WSD method that can be applied to a language even when no sense tagged corpus for that language is available. This is achieved by ***projecting Wordnet and corpus parameters*** from another language to the language in question. The approach is centered on a novel synset based multilingual dictionary (Mohanty et al., 2008) where the synsets of different languages are aligned and thereafter the words within the synsets are manually cross-linked. For example, the word W_{L_1} belonging to synset S of language L_1 will be manually cross-linked to the word W_{L_2} of the corresponding synset in language L_2 to indicate that W_{L_2} is the best substitute for W_{L_1} according to an experienced bilingual speaker's intuition.

We extend their work by addressing the following question on the economics of annotation, lexicon building and performance:

- *Is there an optimal point of balance between the annotation effort and the lexicon building (i.e. manual cross-linking) effort at which one can be assured of best value for money in terms of accuracy?*

To address the above question we first propose a probabilistic cross linking model to eliminate the effort of manually cross linking words within the source and target language synsets and calibrate the resultant trade-off in accuracy. Next, we show that by injecting examples for most frequent hard-to-disambiguate words from the target domain one can achieve higher accuracies at optimal

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 555–563,
Beijing, August 2010

cost of annotation. Finally, we propose a measure for ***cost-benefit analysis*** which identifies the optimal point of balance between these three related entities, viz., cross-linking, sense annotation and accuracy of disambiguation.

The remainder of this paper is organized as follows. In section 2 we present related work. In section 3 we describe the Synset based multilingual dictionary which enables parameter projection. In section 4 we discuss the work of Khapra et al. (2009) on parameter projection for multilingual WSD. Section 5 is on the economics of multilingual WSD. In section 6 we propose a probabilistic model for representing the cross-linkage of words within synsets. In section 7 we present a strategy for injecting hard-to-disambiguate cases from the target language using selective sampling. In section 8 we introduce a measure for **cost-benefit analysis** for calculating the value for money in terms of accuracy, annotation effort and lexicon building effort. In section 9 we describe the experimental setup. In section 10 we present the results followed by discussion in section 11. Section 12 concludes the paper.

2 Related Work

Knowledge based approaches to WSD such as Lesk's algorithm (Lesk, 1986), Walker's algorithm (Walker and Amsler, 1986), Conceptual Density (Agirre and Rigau, 1996) and PageRank (Mihalcea, 2005) are less demanding in terms of resources but fail to deliver good results. Supervised approaches like SVM (Lee et al., 2004) and k-NN (Ng and Lee, 1996), on the other hand, give better accuracies, but the requirement of large annotated corpora renders them unsuitable for resource scarce languages.

Recent work by Khapra et al. (2009) has shown that it is possible to project the parameters learnt from the annotation work of one language to another language provided aligned Wordnets for two languages are available. However, their work does not address the question of further improving the accuracy of WSD by using a small amount of training data from the target language. Some similar work has been done in the area of domain adaptation where Chan et al. (2007) showed that adding just 30% of the target data to the source data achieved the same performance as that obtained by taking the entire source and target data. Similarly, Agirre and de Lacalle (2009) reported a 22% error reduction when source and target data were combined for training a classifier, compared to the case when only the target data was used for training the classifier. However, such combining of training statistics has not been tried in cases where the source data is in one language and the target data is in another language.

To the best of our knowledge, no previous work has attempted to perform resource conscious **all-words multilingual Word Sense Disambiguation** by finding a trade-off between the cost (in terms of annotation effort and lexicon creation effort) and the quality in terms of F-score.

3 Synset based multilingual dictionary

A novel and effective method of storage and use of dictionary in a multilingual setting was proposed by Mohanty et al. (2008). For the purpose of current discussion, we will refer to this multilingual dictionary framework as *MultiDict*. One important departure in this framework from the traditional dictionary is that **synsets are linked, and after that the words inside the synsets are linked**. The basic mapping is thus between synsets and thereafter between the words.

Concepts	L1 (English)	L2 (Hindi)	L3 (Marathi)
04321: a youthful male person	{malechild, boy}	{लड़का *(ladkaa)*, बालक *(baalak)*, बच्चा *(bachchaa)*}	{मुलगा *(mulgaa)*, पोरगा *(porgaa)*, पोर *(por)*}

Table 1: Multilingual Dictionary Framework

Table 1 shows the structure of MultiDict, with one example row standing for the concept of *boy*. The first column is the pivot describing a concept with a unique ID. The subsequent columns show the words expressing the concept in respective languages (in the example table, *English, Hindi and Marathi*). After the synsets are linked, cross linkages are set up manually from the words of a synset to the words of a linked synset of the pivot language. For example, for the Marathi word मुलगा *(mulgaa)*, "a youthful male person", the correct lexical substitute from the corresponding Hindi synset is *लड़का(ladkaa)*. The average number of such links per synset per language pair is approximately 3.

4 Parameter Projection

Khapra et al. (2009) proposed that the various parameters essential for domain-specific Word Sense Disambiguation can be broadly classified into two categories:

Wordnet-dependent parameters:

- belongingness-to-dominant-concept
- conceptual distance
- semantic distance

Corpus-dependent parameters:

- sense distributions
- corpus co-occurrence

They proposed a scoring function (Equation (1)) which combines these parameters to identify the correct sense of a word in a context:

$$S^* = \arg\max_i(\theta_i V_i + \sum_{j \in J} W_{ij} * V_i * V_j) \qquad (1)$$

where,

$$\begin{aligned} i &\in \textit{Candidate Synsets} \\ J &= \textit{Set of disambiguated words} \\ \theta_i &= BelongingnessToDominantConcept(S_i) \\ V_i &= P(S_i|word) \\ W_{ij} &= CorpusCooccurrence(S_i, S_j) \\ &\quad * 1/WNConceptualDistance(S_i, S_j) \\ &\quad * 1/WNSemanticGraphDistance(S_i, S_j) \end{aligned}$$

The first component $\theta_i V_i$ of Equation (1) captures influence of the corpus specific sense of a word in a domain. The other component $W_{ij} * V_i * V_j$ captures the influence of interaction of the candidate sense with the senses of context words weighted by factors of co-occurrence, conceptual distance and semantic distance.

Wordnet-dependent parameters depend on the structure of the Wordnet whereas the ***Corpus-dependent parameters*** depend on various statistics learnt from a sense marked corpora. Both the tasks of (a) constructing a Wordnet from scratch and (b) collecting sense marked corpora for multiple languages are tedious and expensive. Khapra et al. (2009) observed that by ***projecting relations*** from the Wordnet of a language and by ***projecting corpus statistics*** from the sense marked corpora of the language to those of the target language, *the effort required in constructing semantic graphs for multiple Wordnets and collecting sense marked corpora for multiple languages can be avoided or reduced.* At the heart of their work lies the *MultiDict* described in previous section which facilitates parameter projection in the following manner:

1. By linking with the synsets of a pivot resource rich language (Hindi, in our case), the cost of building Wordnets of other languages is partly reduced (semantic relations are inherited). The Wordnet parameters of Hindi Wordnet now become projectable to other languages.

2. For calculating corpus specific sense distributions, $P(Sense\ S_i|Word\ W)$, we need the counts, $\#(S_i, W)$. By using cross linked words in the synsets, these counts become projectable to the target language (Marathi, in our case) as they can be approximated by the counts of the cross linked Hindi words calculated from the Hindi sense marked corpus as follows:

$$P(S_i|W) = \frac{\#(S_i, marathi\_word)}{\sum_j \#(S_j, marathi\_word)}$$

$$P(S_i|W) \approx \frac{\#(S_i, cross\_linked\_hindi\_word)}{\sum_j \#(S_j, cross\_linked\_hindi\_word)}$$

The rationale behind the above approximation is the observation that within a domain sense distributions remain the same across languages.

5 The Economics of Multilingual WSD

The problem of multilingual WSD using parameter projection can be viewed as an economic system consisting of three factors. The first factor is the cost of manually cross-linking the words in a synsets of the target language to the words in the corresponding synset in the pivot language. The second factor is the cost of sense annotated data from the target language. The third factor is the accuracy of WSD The first two factors in some sense relate to the cost of purchasing a commodity and the third factor relates to the commodity itself.

The work of Khapra et al. (2009) as described above does not attempt to reach an optimal cost-benefit point in this economic system. They place their bets on manual cross-linking only and settle for the accuracy achieved thereof. Specifically, they do not explore the inclusion of small amount of annotated data from the target language to boost the accuracy (as mentioned earlier, supervised systems which use annotated data from the target language are known to perform better). Further, it is conceivable that with respect to accuracy-cost trade-off, there obtains a case for *balancing* one cost against the other, *viz.*, the cost of cross-linking and the cost of annotation. In some cases bilingual lexicographers (needed for manual cross-linking) may be more expensive compared to monolingual annotators. There it makes sense to place fewer bets on manual cross-linking and more on collecting annotated corpora. On the other hand if manual cross-linking is cheap then a very small amount of annotated corpora can be used in conjunction with full manual cross-linking to boost the accuracy. Based on the above discussion, if k_a is the cost of sense annotating one word, k_c is the cost of manually cross-linking a word and A is the accuracy desired then the problem of multilingual WSD can be cast as an optimization problem:

$$\begin{aligned} &minimize \quad w_a * k_a + w_c * k_c \\ &\qquad s.t. \\ &\qquad\quad Accuracy \geq A \end{aligned}$$

where, w_c and w_a are the number of words to be manually cross linked and annotated respectively. Ours is thus a 3-factor economic model (cross-linking, annotation and accuracy) as opposed to the 2-factor model (cross-linking, accuracy) proposed by Khapra et al. (2009).

6 Optimal cross-linking

As mentioned earlier, in some cases where bilingual lexicographers are expensive we might be interested in reducing the effort of manual cross-linking. For such situations, we propose that only a small number of words, comprising of the most frequently appearing ones should be manually cross linked and the rest of the words should be cross-linked using a probabilistic model. The rationale here is simple: invest money in words which are bound to occur frequently in the test data and achieve maximum impact on the accuracy. In the following paragraphs, we explain our probabilistic cross linking model.

The model proposed by Khapra et al. (2009) is a deterministic model where the expected count for (Sense S, Marathi_Word W), *i.e.*, the number of times the word W appears in sense S is approximated by the count for the corresponding cross linked Hindi word. Such a model assumes that each Marathi word links to appropriate Hindi word(s) as identified manually by a lexicographer. Instead, **we propose a probabilistic model where a Marathi word can link to every word in the corresponding Hindi synset with some probability**. The expected count for (S, W) can then be estimated as:

$$E[\#(S,W)] = \sum_{h_i \in cross\_links} P(h_i|W,S) * \#(S,h_i) \quad (2)$$

where, $P(h_i|W,S)$ is the probability that the word h_i from the corresponding Hindi synset is the correct cross-linked word for the given Marathi word. For example, one of the senses of the Marathi word *maan* is {neck} *i.e.* *"the body part which connects the head to the rest of the body"*. The corresponding Hindi synset has 10 words {*gardan, gala, greeva, halak, kandhar and so on*}. Thus, using Equation (2), the expected count, $E[C(\{neck\}, maan)]$, is calculated as:

$$\begin{aligned} &E[\#(\{neck\}, maan)] = \\ &\quad P(gardan|maan, \{neck\}) * \#(\{neck\}, gardan) \\ &+ P(gala|maan, \{neck\}) * \#(\{neck\}, gala) \\ &+ P(greeva|maan, \{neck\}) * \#(\{neck\}, greeva) \\ &+ \ \ldots \ so\ on\ for\ all\ words\ in\ the\ Hindi\ synset \end{aligned}$$

Instead of using a uniform probability distribution over the Hindi words we go by the empirical observation that some words in a synset are more representative of that sense than other words, *i.e. some words are more preferred while expressing that sense*. For example, out of the 10 words in the Hindi synset only 2 words $\{$*gardan, gala*$\}$ appeared in the corpus. We thus estimate the value of $P(h_i|W, S)$ empirically from the Hindi sense marked corpus by making the following independence assumption:

$$P(h_i|W, S) = P(h_i|S)$$

The rationale behind the above independence assumption becomes clear if we represent words and synsets using the Bayesian network of Figure 1. Here, the Hindi word h_i and the Marathi word W are considered to be derived from the same parent concept S. In other words, they represent two different manifestations- one in Hindi and one in Marathi- of the same synset S. Given the above representation, it is easy to see that given the parent synset S, the Hindi word h_i is independent of the Marathi word W.

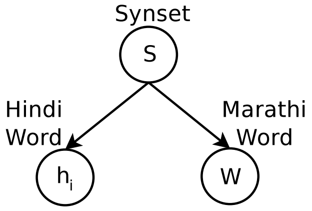


Figure 1: Bayesian network formed by a synset S and the constituent Hindi and Marathi words

7 Optimal annotation using Selective Sampling

In the previous section we dealt with the question of optimal cross-linking. Now we take up the other dimension of this economic system, *viz.*, optimal use of annotated corpora for better accuracy. In other words, if an application demands higher accuracy for WSD and is willing to pay for some annotation then there should be a way of ensuring best possible accuracy at lowest possible cost. This can be done by including small amount of sense annotated data from the target language. The simplest strategy is to randomly annotate text from the target language and use it as training data. However, this strategy of random sampling may not be the most optimum in terms of cost. Instead, we propose a selective sampling strategy where the aim is to identify ***hard-to-disambiguate*** words from the target language and use them for training.

The algorithm proceeds as follows:

1. First, using the probabilistic cross linking model and aligned Wordnets we learn the parameters described in Section 4.

2. We then apply this scoring function on untagged examples (development set) from the target language and identify ***hard-to-disambiguate*** words *i.e.*, the words which were disambiguated with a very low confidence.

3. Training instances of these words are then injected into the training data and the parameters learnt from them are used instead of the projected parameters learnt from the source language corpus.

Thus, the selective sampling strategy ensures that we get maximum value for money by spending it on annotating only those words which would otherwise not have been disambiguated correctly. A random selection strategy, in contrast, might bring in words which were disambiguated correctly using only the projected parameters.

8 A measure for cost-benefit analysis

We need a measure for cost-benefit analysis based on the three dimensions of our economic system, *viz.*, annotation effort, lexicon creation effort and performance in terms of F-score. The first two dimensions can be fused into a single dimension by expressing the annotation effort and lexicon creation effort in terms of cost incurred. For example, we assume that the cost of annotating one word is k_a and the cost of cross-linking one word is k_c rupees. Further, we define a baseline and an upper bound for the F-score. In this case, the baseline would be the accuracy that can be obtained without spending any money on cross-linking and annotation in the target language. An upper bound could be the best F-score obtained using a large amount of annotated corpus in the target domain. Based on the above description, an ideal measure for cost-benefit analysis would assign a

1. reward depending on the improvement over the baseline performance.

2. penalty depending on the difference from the upper bound on performance.

3. reward inversely proportional to the cost incurred in terms of annotation effort and/or manual cross-linking.

Based on the above wish-list we propose a measure for cost-benefit analysis. Let,

$$\begin{aligned} MGB &= Marginal\ Gain\ over\ Baseline\ (MGB) \\ &= \frac{Performance(P) - Baseline(B)}{Cost(C)} \\ MDU &= Marginal\ Drop\ from\ Upperbound\ (MDU) \\ &= \frac{UpperBound(U) - Performance(P)}{Cost(C)} \end{aligned}$$

then

$$CostBenefit(CB) = MGB - MDU$$

9 Experimental Setup

We used Hindi as the source language (S_L) and trained a WSD engine using Hindi sense tagged corpus. The parameters thus learnt were then projected using the *MultiDict* (refer section 3 and 4) to build a resource conscious Marathi (T_L) WSD engine. We used the same dataset as described in Khapra et al. (2009) for all our experiments. The data was collected from two domains, *viz.*, Tourism and Health. The data for Tourism domain was collected by manually translating English documents downloaded from Indian Tourism websites into Hindi and Marathi. Similarly, English documents for Health domain were obtained from two doctors and were manually translated into Hindi and Marathi. The Hindi and Marathi documents thus created were manually sense annotated by two lexicographers adept in Hindi and Marathi using the respective Wordnets as sense repositories. Table 2 summarizes some statistics about the corpora.

As for cross-linking, Hindi is used as the pivot language and words in Marathi synset are linked to the words in the corresponding Hindi synset. The total number of cross-links that were manually setup were 3600 for Tourism and 1800 for Health. The cost of cross-linking as well as sense annotating one word was taken to be 10 rupees. These costs were estimated based on quotations from lexicographers. However, these costs need to be taken as representative values only and may vary greatly depending on the availability of skilled bilingual lexicographers and skilled monolingual annotators.

Language	#of polysemous words		average degree of polysemy	
	Tourism	Health	Tourism	Health
Hindi	56845	30594	3.69	3.59
Marathi	34156	10337	3.41	3.60

Table 2: Number of polysemous words and average degree of polysemy.

10 Results

Tables 3 and 4 report the average 4-fold performance on Marathi Tourism and Health data using different proportions of available resources, *i.e.*, annotated corpora and manual cross-links. In each of these tables, along the rows, we increase the amount of Marathi sense annotated corpora from 0K to 6K. Similarly, along the columns we show the increase in the number of manual cross links (MCL) used. For example, the second column of Tables 3 and 4 reports the F-scores when probabilistic cross-linking (PCL) was used for all words (*i.e.*, no manual cross-links) and varying amounts of sense annotated corpora from Marathi were used. Similarly, the first row represents the case in which no sense annotated corpus from Marathi was used and varying amounts of manual cross-links were used.

We report three values in the tables, *viz.*, F-score (F), cost in terms of money (C) and the cost-benefit (CB) obtained by using x amount of annotated corpus and y amount of manual cross-links. The cost was estimated using the values given in section 9 (*i.e.*, 10 rupees for cross-linking or sense annotating one word). For calculating, the cost-benefit baseline was taken as the F-score obtained by using no cross-links and no annotated corpora *i.e.* 68.21% for Tourism and 67.28% for Health (see first F-score cell of Tables 3 and 4). Similarly the upper bound (F-scores obtained by training on entire Marathi sense marked corpus) for Tourism and Health were 83.16% and 80.67% respectively (see last row of Table 5).

Due to unavailability of large amount of tagged Health corpus, the injection size was varied from 0-to-4K only. In the other dimension, we varied the cross-links from 0 to 1/3rd to 2/3rd to full only

Selective Sampling	Only PCL			1/3 MCL			2/3 MCL			Full MCL		
	F	C	CB	F	C	CB	F	C	CB	F	C	CB
0K	68.21	0	-	72.08	12	-0.601	73.31	24	-0.198	73.34	36	-0.130
1K	71.18	10	-0.901	74.96	22	-0.066	77.58	34	0.111	77.73	46	0.089
2K	74.35	20	-0.134	76.96	32	0.080	**78.57**	**44**	**0.131**	79.23	56	0.127
3K	75.21	30	-0.032	77.78	42	0.100	78.68	54	0.111	79.8	66	0.125
4K	76.40	40	0.036	78.66	52	0.114	79.18	64	0.110	80.36	76	0.123
5K	77.04	50	0.054	78.51	62	0.091	79.60	74	0.106	80.46	86	0.111
6K	78.58	60	0.097	79.75	72	0.113	80.8	84	0.122	80.44	96	0.099

Table 3: F-Score (F) in %, Cost (C) in thousand rupees and Cost Benefit (CB) values using different amounts of sense annotated corpora and manual cross links in Tourism domain.

Selective Sampling	Only PCL			1/3 MCL			2/3 MCL			Full MCL		
	F	C	CB	F	C	CB	F	C	CB	F	C	CB
0K	67.28	0	-	71.39	6	-0.862	73.06	12	-0.153	73.34	18	-0.071
1K	72.51	10	-0.293	75.57	16	0.199	**77.41**	**22**	**0.312**	78.16	28	0.299
2K	75.64	20	0.167	77.29	26	0.255	78.13	32	0.260	78.63	38	0.245
3K	76.78	30	0.187	79.35	36	0.299	79.79	42	0.277	79.88	48	0.246
4K	77.42	40	0.172	79.59	46	0.244	80.54	52	0.253	80.15	58	0.213

Table 4: F-Score (F) in %, Cost (C) in thousand rupees and Cost Benefit (CB) values using different amounts of sense annotated corpora and manual cross links in Health domain.

Strategy	Tourism	Health
WFS	57.86	52.77
Only PCL	68.21	67.28
1/6 MCL	69.95	69.57
2/6 MCL	72.08	71.39
3/6 MCL	72.97	72.61
4/6 MCL	73.39	73.06
5/6 MCL	73.55	73.27
Full MCL	73.62	73.34
Upper Bound	83.16	80.67

Table 5: F-score (in %) obtained by using different amounts of manually cross linked words

Strategy	Size of target side annotated corpus						
	0K	1K	2K	3K	4K	5K	6K
Random + PCL	68.21	70.62	71.79	73.03	73.61	76.42	77.52
Random + MCL	73.34	75.32	75.89	76.79	76.83	78.91	80.87
Selective Sampling + PCL	68.21	71.18	74.35	75.21	76.40	77.04	78.58
Selective Sampling + MCL	73.34	77.73	79.23	79.8	79.8	80.46	80.44

Table 6: Comparing F-scores obtained using random sampling and selective sampling (Tourism)

Strategy	Size of target side annotated corpus						
	0K	1K	2K	3K	4K	5K	6K
Annotation + PCL	68.21	71.20	74.35	75.21	76.40	77.04	78.58
Only Annotation	57.86	62.32	64.84	66.86	68.89	69.64	71.82

Table 7: Comparing F-scores obtained using Only Annotation and Annotation + PCL(Tourism)

(refer to Tables 3 and 4). However, to give an idea about the soundness of probabilistic cross-linking we performed a separate set of experiments by varying the number of cross-links and using no sense annotated corpora. Table 5 summarizes these results and compares them with the baseline (Wordnet first sense) and skyline.

In Table 6 we compare our selective sampling strategy with random sampling when fully probabilistic cross-linking (PCL) is used and when fully manual cross-linking (MCL) is used. Here again, due to lack of space we report results only on Tourism domain. However, we would like to mention that similar experiments on Health domain showed that the results were indeed consistent.

Finally, in Table 7 we compare the accuracies obtained when certain amount of annotated corpus from Marathi is used alone, with the case when the same amount of annotated corpus is used in conjunction with probabilistic cross-linking. While calculating the results for the second row in Table 7, we found that the recall was very low due to the small size of injections. Hence, to ensure a fair comparison with our strategy (first row) we used the Wordnet first sense (WFS) for these recall errors (a typical practice in WSD literature).

11 Discussions

We make the following observations:

1. PCL v/s MCL: Table 5 shows that the probabilistic cross-linking model performs much better than the WFS (a typically reported baseline) and it comes very close to the performance of manual cross-linking. This establishes the soundness of the probabilistic model and suggests that with a little compromise in the accuracy, the model can be used as an approximation to save the cost of manual cross-linking. Further, in Table 7 we see that when PCL is used in conjunction with certain amount of annotated corpus we get up to 9% improvement in F-score as compared to the case when the same amount of annotated corpus is used alone. Thus, in the absence of skilled bilingual lexicographers, PCL can still be used to boost the accuracy obtained using annotated corpora.

2. Selective Sampling v/s Random Annotation: Table 6 shows the benefit of selective sampling over random annotation. This benefit is felt more when the amount of training data injected from Marathi is small. For example, when an annotated corpus of size 2K is used, selective sampling gives an advantage of 3% to 4% over random selection. Thus the marginal gain (*i.e.*, value for money) obtained by using selective sampling is more than that obtained by using random annotation.

3. Optimal cost-benefit: Finally, we address the main message of our work, *i.e.*, finding the best cost benefit. By referring to Tables 3 and 4, we see that the best value for money in Tourism domain is obtained by manually cross-linking 2/3rd of all corpus words and sense annotating 2K target words and in the Health domain it is obtained by manually cross-linking 2/3rd of all corpus words but sense annotating only 1K words. This suggests that striking a balance between cross-linking and annotation gives the best value for money. Further, we would like to highlight that our 3-factor economic model is able to capture these relations better than the 2-factor model of Khapra et al. (2010). As per their model the best F-score achieved using manual cross-linking for ALL words was 73.34% for both Tourism and Health domain at a cost of 36K and 18K respectively. On the other hand, using our model we obtain higher accuracies of 76.96% in the Tourism domain (using 1/3rd manual cross-links and 2K injection) at a lower total cost (32K rupees) and 75.57% in the Health domain (using only 1/3rd cross-linking and 1K injection) at a lower cost (16K rupees).

12 Conclusion

We reported experiments on multilingual WSD using different amounts of annotated corpora and manual cross-links. We showed that there exists some trade-off between the accuracy and *balancing* the cost of annotation and lexicon creation. In the absence of skilled bilingual lexicographers one can use a probabilistic cross-linking model and still obtain good accuracies. Also, while sense annotating a corpus, careful selection of words using selective sampling can give better marginal gain as compared to random sampling.

References

Agirre, Eneko and Oier Lopez de Lacalle. 2009. Supervised domain adaption for wsd. In *EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 42–50, Morristown, NJ, USA. Association for Computational Linguistics.

Agirre, Eneko and German Rigau. 1996. Word sense disambiguation using conceptual density. In *In Proceedings of the 16th International Conference on Computational Linguistics (COLING)*.

Chan, Y.S., H. T. Ng, and D. Chiang. 2007. Word sense disambiguation improves statistical machine translation. In *In Proc. of ACL*.

Khapra, Mitesh M., Sapan Shah, Piyush Kedia, and Pushpak Bhattacharyya. 2009. Projecting parameters for multilingual word sense disambiguation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 459–467, Singapore, August. Association for Computational Linguistics.

Khapra, Mitesh, Sapan Shah, Piyush Kedia, and Pushpak Bhattacharyya. 2010. Domain-specific word sense disambiguation combining corpus based and wordnet based parameters. In *5th International Conference on Global Wordnet (GWC2010)*.

Lee, Yoong Keok, Hwee Tou Ng, and Tee Kiah Chia. 2004. Supervised word sense disambiguation with support vector machines and multiple knowledge sources. In *Proceedings of Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 137–140.

Lesk, Michael. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *In Proceedings of the 5th annual international conference on Systems documentation*.

Mihalcea, Rada. 2005. Large vocabulary unsupervised word sense disambiguation with graph-based algorithms for sequence data labeling. In *In Proceedings of the Joint Human Language Technology and Empirical Methods in Natural Language Processing Conference (HLT/EMNLP)*, pages 411–418.

Mohanty, Rajat, Pushpak Bhattacharyya, Prabhakar Pande, Shraddha Kalele, Mitesh Khapra, and Aditya Sharma. 2008. Synset based multilingual dictionary: Insights, applications and challenges. In *Global Wordnet Conference*.

Ng, Hwee Tou and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word senses: An exemplar-based approach. In *In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 40–47.

Veronis, Jean. 2004. Hyperlex: Lexical cartography for information retrieval. In *Computer Speech and Language*, pages 18(3):223–252.

Walker, D. and R. Amsler. 1986. The use of machine readable dictionaries in sublanguage analysis. In *In Analyzing Language in Restricted Domains, Grishman and Kittredge (eds), LEA Press*, pages 69–83.

A Cross-lingual Annotation Projection Approach for Relation Detection

Seokhwan Kim[†], **Minwoo Jeong**[‡], **Jonghoon Lee**[†], **Gary Geunbae Lee**[†]
[†]Department of Computer Science and Engineering,
Pohang University of Science and Technology
{megaup|jh21983|gblee}@postech.ac.kr
[‡]Saarland University
m.jeong@mmci.uni-saarland.de

Abstract

While extensive studies on relation extraction have been conducted in the last decade, statistical systems based on supervised learning are still limited because they require large amounts of training data to achieve high performance. In this paper, we develop a cross-lingual annotation projection method that leverages parallel corpora to bootstrap a relation detector without significant annotation efforts for a resource-poor language. In order to make our method more reliable, we introduce three simple projection noise reduction methods. The merit of our method is demonstrated through a novel Korean relation detection task.

1 Introduction

Relation extraction aims to identify semantic relations of entities in a document. Many relation extraction studies have followed the Relation Detection and Characterization (RDC) task organized by the Automatic Content Extraction project (Doddington et al., 2004) to make multilingual corpora of English, Chinese and Arabic. Although these datasets encourage the development and evaluation of statistical relation extractors for such languages, there would be a scarcity of labeled training samples when learning a new system for another language such as Korean. Since manual annotation of entities and their relations for such *resource-poor languages* is very expensive, we would like to consider instead a weakly-supervised learning technique in order to learn the relation extractor without significant annotation efforts. To do this, we propose to leverage parallel corpora to project the relation annotation on the source language (e.g. English) to the target (e.g. Korean).

While many supervised machine learning approaches have been successfully applied to the RDC task (Kambhatla, 2004; Zhou et al., 2005; Zelenko et al., 2003; Culotta and Sorensen, 2004; Bunescu and Mooney, 2005; Zhang et al., 2006), few have focused on weakly-supervised relation extraction. For example, (Zhang, 2004) and (Chen et al., 2006) utilized weakly-supervised learning techniques for relation extraction, but they did not consider weak supervision in the context of cross-lingual relation extraction. Our key hypothesis on the use of parallel corpora for learning the relation extraction system is referred to as *cross-lingual annotation projection*. Early studies of cross-lingual annotation projection were accomplished for lexically-based tasks; for example part-of-speech tagging (Yarowsky and Ngai, 2001), named-entity tagging (Yarowsky et al., 2001), and verb classification (Merlo et al., 2002). Recently, there has been increasing interest in applications of annotation projection such as dependency parsing (Hwa et al., 2005), mention detection (Zitouni and Florian, 2008), and semantic role labeling (Pado and Lapata, 2009). However, to the best of our knowledge, no work has reported on the RDC task.

In this paper, we apply a cross-lingual annotation projection approach to binary *relation detection*, a task of identifying the relation between two entities. A simple projection method propagates the relations in source language sentences to

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 564–571,
Beijing, August 2010

word-aligned target sentences, and a target relation detector can bootstrap from projected annotation. However, this automatic annotation is unreliable because of mis-classification of source text and word alignment errors, so it causes a critical falling-off in annotation projection quality. To alleviate this problem, we present three noise reduction strategies: a heuristic filtering; an alignment correction with dictionary; and an instance selection based on assessment, and combine these to yield a better result.

We provide a quantitive evaluation of our method on a new Korean RDC dataset. In our experiment, we leverage an English-Korean parallel corpus collected from the Web, and demonstrate that the annotation projection approach and noise reduction method are beneficial to build an initial Korean relation detection system. For example, the combined model of three noise reduction methods achieves F1-scores of 36.9% (59.8% precision and 26.7% recall), favorably comparing with the 30.5% shown by the supervised baseline.[1]

The remainder of this paper is structured as follows. In Section 2, we describe our cross-lingual annotation projection approach to relation detection task. Then, we present the noise reduction methods in Section 3. Our experiment on the proposed Korean RDC evaluation set is shown in Section 4 and Section 5, and we conclude this paper in Section 6.

2 Cross-lingual Annotation Projection for Relation Detection

The annotation projection from a resource-rich language L_1 to a resource-poor language L_2 is performed by a series of three subtasks: annotation, projection and assessment.

The annotation projection for relation detection can be performed as follows:

1) For a given pair of bi-sentences in parallel corpora between a resource-rich language L_1 and a target language L_2, the relation detection task is carried out for the sentence in L_1.

2) The annotations obtained by analyzing the sentence in L_1 are projected onto the sentence in L_2 based on the word alignment information.

3) The projected annotations on the sentence in L_2 are utilized as resources to perform the relation detection task for the language L_2.

2.1 Annotation

The first step to projecting annotations from L_1 onto L_2 is obtaining annotations for the sentences in L_1. Since each instance for relation detection is composed of a pair of entity mentions, the information about entity mentions on the given sentences should be identified first. We detect the entities in the L_1 sentences of the parallel corpora. Entity identification generates a number of instances for relation detection by coupling two entities within each sentence. For each instance, the existence of semantic relation between entity mentions is explored, which is called relation detection. We assume that there exist available models or systems for all annotation processes, including not only an entity tagger and a relation detector themselves, but also required preprocessors such as a part-of-speech tagger, base-phrase chunker, and syntax parser for analyzing text in L_1.

Figure 1 shows an example of annotation projection for relation detection of a bitext in English and Korean. The annotation of the sentence in English shows that “Jan Mullins” and “Computer Recycler Incorporated” are entity mentions of a person and an organization, respectively. Furthermore, the result indicates that the pair of entities has a semantic relationship categorized as “ROLE.Owner” type.

2.2 Projection

In order to project the annotations from the sentences in L_1 onto the sentences in L_2, we utilize the information of word alignment which plays an important role in statistical machine translation techniques. The word alignment task aims to identify translational relationships among the words in a bitext and produces a bipartite graph with a set of edges between words with translational relationships as shown in Figure 1. In the same manner as the annotation in L_1, entities are

[1] The dataset and the parallel corpus are available on the author’s website,
http://isoft.postech.ac.kr/~megaup/research/resources/.

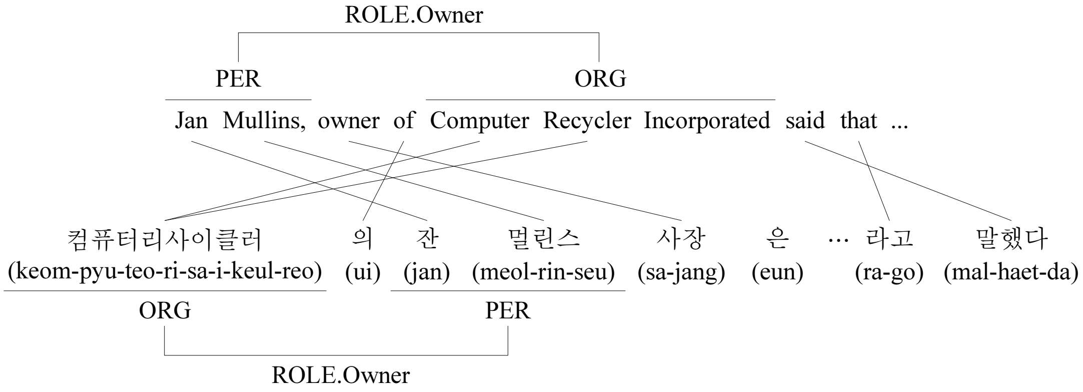


Figure 1: An example of annotation projection for relation detection of a bitext in English and Korean

considered as the first units to be projected. We assume that the words of the sentences in L_2 aligned with a given entity mention in L_1 inherit the information about the original entity in L_1.

After projecting the annotations of entity mentions, the projections for relational instances follow. A projection is performed on a projected instance in L_2 which is a pair of projected entities by duplicating annotations of the original instance in L_1.

Figure 1 presents an example of projection of a positive relational instance between “Jan Mullins” and “Computer Recycler Incorporated” in the English sentence onto its translational counterpart sentence in Korean. “Jan meol-rin-seu” and “keom-pyu-teo-ri-sa-i-keul-reo” are labeled as entity mentions with types of a person’s name and an organization’s name respectively. In addition, the instance composed of the two projected entities is annotated as a positive instance, because its original instance on the English sentence also has a semantic relationship.

As the description suggests, the annotation projection approach is highly dependant on the quality of word alignment. However, the results of automatic word alignment may include several noisy or incomplete alignments because of technical difficulties. We present details to tackle the problem by relieving the influence of alignment errors in Section 3.

2.3 Assessment

The most important challenge for annotation projection approaches is how to improve the robustness against the erroneous projections. The noise produced by not only word alignment but also mono-lingual annotations in L_1 accumulates and brings about a drastic decline in the quality of projected annotations.

The simplest policy of utilizing the projected annotations for relation detection in L_2 is to consider that all projected instances are equivalently reliable and to employ entire projections as training instances for the task without any filtering. In contrast with this policy, which is likely to be substandard, we propose an alternative policy where the projected instances are assessed and only the instances judged as reliable by the assessment are utilized for the task. Details about the assessment are provided in Section 3.

3 Noise Reduction Strategies

The efforts to reduce noisy projections are considered indispensable parts of the projection-based relation detection method in a resource-poor language. Our noise reduction approach includes the following three strategies: heuristic-based alignment filtering, dictionary-based alignment correction, and assessment-based instance selection.

3.1 Heuristic-based Alignment Filtering

In order to improve the performance of annotation projection approaches, we should break the bottleneck caused by the low quality of automatic word alignment results. As relation detection is carried out for each instance consisting of two entity mentions, the annotation projection for relation detection concerns projecting only entity mentions and their relational instances. Since this is different from other shallower tasks such as part-of-speech tagging, base phrase chunking, and dependency parsing which should consider projections for all word units, we define and apply some heuristics specialized to projections of entity mentions and relation instances to improve robustness of the method against erroneous alignments, as follows:

- A projection for an entity mention should be based on alignments between contiguous word sequences. If there are one or more gaps in the word sequence in L2 aligned with an entity mention in the sentence in L1, we assume that the corresponding alignments are likely to be erroneous. Thus, the alignments of non-contiguous words are excluded in projection.

- Both an entity mention in L_1 and its projection in L_2 should include at least one base noun phrase. If no base noun phrase occurs in the original entity mention in L_1, it may suggest some errors in annotation for the sentence in L_1. The same case for the projected instance raises doubts about alignment errors. The alignments between word sequences without any base noun phrase are filtered out.

- The projected instance in L2 should satisfy the clausal agreement with the original instance in L1. If entities of an instance are located in the same clause (or different clauses), its projected instance should be in the same manner. The instances without clausal agreement are ruled out.

3.2 Dictionary-based Alignment Correction

The errors in word alignment are composed of not only imprecise alignments but also incomplete alignments. If an alignment of an entity among two entities of a relation instance is not provided in the result of the word alignment task, the projection for the corresponding instance is unavailable. Unfortunately, the above-stated alignment filtering heuristics for improving the quality of projections make the annotation loss problems worse by filtering out several alignments likely to be noisy.

In order to solve this problem, a dictionary-based alignment correction strategy is incorporated in our method. The strategy requires a bilingual dictionary for entity mentions. Each entry of the dictionary is a pair of entity mention in L_1 and its translation or transliteration in L_2. For each entity to be projected from the sentence in L_1, its counterpart in L_2 is retrieved from the bilingual dictionary. Then, we seek the retrieved entity mention from the sentence in L_2 by finding the longest common subsequence. If a subsequence matched to the retrieved mention is found in the sentence in L_2, we make a new alignment between it and its original entity on the L_1 sentence.

3.3 Assessment-based Instance Selection

The reliabilities of instances projected via a series of independent modules are different from each other. Thus, we propose an assessment strategy for each projected instance. To evaluate the reliability of a projected instance in L_2, we use the confidence score of monolingual relation detection for the original counterpart instance in L_1. The acceptance of a projected instance is determined by whether the score of the instance is larger than a given threshold value θ. Only accepted instances are considered as the results of annotation projection and applied to solve the relation detection task in target language L_2.

4 Experimental Setup

To demonstrate the effectiveness of our cross-lingual annotation projection approach for relation detection, we performed an experiment on relation detection in Korean text with propagated annotations from English resources.

4.1 Annotation

The first step to evaluate our method was annotating the English sentences in a given parallel corpus. We use an English-Korean parallel corpus crawled from an English-Korean dictionary on the web. The parallel corpus consists of 454,315 bi-sentence pairs in English and Korean [2]. The English sentences in the parallel corpus were prepro-

[2] The parallel corpus collected and other resources are all available in our website
http://isoft.postech.ac.kr/∼megaup/research/resources/

cessed by the Stanford Parser [3] (Klein and Manning, 2003) which provides a set of analyzed results including part-of-speech tag sequences, a dependency tree, and a constituent parse tree for a sentence.

The annotation for English sentences is divided into two subtasks: entity mention recognition and relation detection. We utilized an off-the-shelf system, Stanford Named Entity Recognizer [4] (Finkel et al., 2005) for detecting entity mentions on the English sentences. The total number of English entities detected was 285,566. Each pair of recognized entities within a sentence was considered as an instance for relation detection.

A classification model learned with the training set of the ACE 2003 corpus which consists of 674 documents and 9,683 relation instances was built for relation detection in English. In our implementation, we built a tree kernel-based SVM model using SVM-Light [5] (Joachims, 1998) and Tree Kernel Tools [6] (Moschitti, 2006). The subtree kernel method (Moschitti, 2006) for shortest path enclosed subtrees (Zhang et al., 2006) was adopted in our model. Our relation detection model achieved 81.2/69.8/75.1 in Precision/Recall/F-measure on the test set of the ACE 2003 corpus, which consists of 97 documents and 1,386 relation instances.

The annotation of relations was performed by determining the existence of semantic relations for all 115,452 instances with the trained model for relation detection. The annotation detected 22,162 instances as positive which have semantic relations.

4.2 Projection

The labels about entities and relations in the English sentences of the parallel corpora were propagated into the corresponding sentences in Korean. The Korean sentences were preprocessed by our part-of-speech tagger [7] (Lee et al., 2002) and a dependency parser implemented by MSTParser with a model trained on the Sejong corpus (Kim, 2006).

Filter	Without assessing	With assessing
none	97,239	39,203
+ heuristics	31,652	12,775
+ dictionary	39,891	17,381

Table 1: Numbers of projected instances

The annotation projections were performed on the bi-sentences of the parallel corpus followed by descriptions mentioned in Section 2.2. The bi-sentences were processed by the GIZA++ software (Och and Ney, 2003) in the standard configuration in both English-Korean and Korean-English directions. The bi-direcional alignments were joined by the grow-diag-final algorithm, which is widely used in bilingual phrase extraction (Koehn et al., 2003) for statistical machine translation. This system achieved 65.1/41.6/50.8 in Precision/Recall/F-measure in our evaluation of 201 randomly sampled English-Korean bi-sentences with manually annotated alignments.

The number of projected instances varied with the applied strategies for reducing noise as shown in Table 1. Many projected instances were filtered out by heuristics, and only 32.6% of the instances were left. However, several instances were rescued by dictionary-based alignment correction and the number of projected instances increased from 31,652 to 39,891. For all cases of noise reduction strategies, we performed the assessment-based instance selection with a threshold value θ of 0.7, which was determined empirically through the grid search method. About 40% of the projected instances were accepted by instance selection.

4.3 Evaluation

In order to evaluate our proposed method, we prepared a dataset for the Korean RDC task. The dataset was built by annotating the information about entities and relations in 100 news documents in Korean. The annotations were performed by two annotators following the guidelines for the ACE corpus processed by LDC. Our Korean RDC corpus consists of 835 sentences, 3,331 entity mentions, and 8,354 relation instances. The sen-

[3] http://nlp.stanford.edu/software/lex-parser.shtml

[4] http://nlp.stanford.edu/software/CRF-NER.shtml

[5] http://svmlight.joachims.org/

[6] http://disi.unitn.it/∼moschitt/Tree-Kernel.htm

[7] http://isoft.postech.ac.kr/∼megaup/research/postag/

Model	w/o assessing			with assessing		
	P	R	F	P	R	F
Baseline	60.5	20.4	30.5	-	-	-
Non-filtered	22.5	6.5	10.0	29.1	13.2	18.2
Heuristic	51.4	15.5	23.8	56.1	22.9	32.5
Heuristic + Dictionary	55.3	19.4	28.7	59.8	26.7	36.9

Table 2: Experimental Results

tences of the corpus were preprocessed by equivalent systems used for analyzing Korean sentences for projection. We randomly divided the dataset into two subsets with the same number of instances for use as a training set to build the baseline system and for evaluation.

For evaluating our approach, training instance sets to learn models were prepared for relation detection in Korean. The instances of the training set (half of the manually built Korean RDC corpufs) were used to train the baseline model. All other sets of instances include these baseline instances and additional instances propagated by the annotation projection approach. The training sets with projected instances are categorized into three groups by the level of applied strategies for noise reduction. While the first set included all projections without any noise reduction strategies, the second included only the instances accepted by the heuristics. The last set consisted of the results of a series of heuristic-based filtering and dictionary-based correction. For each training set with projected instances, an additional set was derived by performing assessment-based instance selection.

We built the relation detection models for all seven training sets (a baseline set, three projected sets without assessing, and three projected sets with assessing). Our implementations are based on the SVM-Light and Tree Kernel Tools described in the former subsection. The shortest path dependency kernel (Bunescu and Mooney, 2005) implemented by the subtree kernel method (Moschitti, 2006) was adopted to learn all models.

The performance for each model was evaluated with the predictions of the model on the test set, which was the other half of Korean RDC corpus. We measured the performances of the models on true entity mentions with true chaining of coreference. Precision, Recall and F-measure were adopted for our evaluation.

5 Experimental Results

Table 2 compares the performances of the different models which are distinguished by the applied strategies for noise reduction. It shows that:

- The model with non-filtered projections achieves extremely poor performance due to a large number of erroneous instances. This indicates that the efforts for reducing noise are urgently needed.

- The heuristic-based alignment filtering helps to improve the performance. However, it is much worse than the baseline performance because of a falling-off in recall.

- The dictionary-based correction to our projections increased both precision and recall compared with the former models with projected instances. Nevertheless, it still fails to achieve performance improvement over the baseline model.

- For all models with projection, the assessment-based instance selection boosts the performances significantly. This means that this selection strategy is crucial in improving the performance of the models by excluding unreliable instances with low confidence.

- The model with heuristics and assessments finally achieves better performance than the baseline model. This suggests that the projected instances have a beneficial influence on the relation detection task when at least these two strategies are adopted for reducing noises.

- The final model incorporating all proposed noise reduction strategies outperforms the baseline model by 6 in F-measure. This is due to largely increased recall by absorbing more useful features from the well-refined set of projected instances.

The experimental results show that our proposed techniques effectively improve the performance of relation detection in the resource-poor Korean language with a set of annotations projected from the resource-rich English language.

6 Conclusion

This paper presented a novel cross-lingual annotation projection method for relation extraction in a resource-poor language. We proposed methods of propagating annotations from a resource-rich language to a target language via parallel corpora. In order to relieve the bad influence of noisy projections, we focused on the strategies for reducing the noise generated during the projection. We applied our methods to the relation detection task in Korean. Experimental results show that the projected instances from an English-Korean parallel corpus help to improve the performance of the task when our noise reduction strategies are adopted.

We would like to introduce our method to the other subtask of relation extraction, which is relation categorization. While relation detection is a binary classification problem, relation categorization can be solved by a classifier for multiple classes. Since the fundamental approaches of the two tasks are similar, we expect that our projection-based relation detection methods can be easily adapted to the relation categorization task.

For this further work, we are concerned about the problem of low performance for Korean, which was below 40 for relation detection. The relation categorization performance is mostly lower than detection because of the larger number of classes to be classified, so the performance of projection-based approaches has to be improved in order to apply them. An experimental result of this work shows that the most important factor in improving the performance is how to select the reliable instances from a large number of projections. We plan to develop more elaborate strategies for instance selection to improve the projection performance for relation extraction.

Acknowledgement

This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2010-C1090-1031-0009).

References

Bunescu, Razvan C. and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, page 724731.

Chen, Jinxiu, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. 2006. Relation extraction using label propagation based semi-supervised learning. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 129–136, Sydney, Australia. Association for Computational Linguistics.

Culotta, Aron and Jaffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of ACL*, volume 4.

Doddington, George, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) programtasks, data, and evaluation. In *Proceedings of LREC*, volume 4, page 837840.

Finkel, Jenny R., Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, volume 43, page 363.

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

Joachims, Thorsten. 1998. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142.

Kambhatla, Nanda. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22, Barcelona, Spain. Association for Computational Linguistics.

Kim, Hansaem. 2006. Korean national corpus in the 21st century sejong project. In *Proceedings of the 13th NIJL International Symposium*, page 4954.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 423–430, Sapporo, Japan. Association for Computational Linguistics.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, volume 1, pages 48–54.

Lee, Gary Geunbae, Jeongwon Cha, and Jong-Hyeok Lee. 2002. Syllable pattern-based unknown morpheme segmentation and estimation for hybrid part-of-speech tagging of korean. *Computational Linguistics*, 28(1):53–70.

Merlo, Paola, Suzanne Stevenson, Vivian Tsang, and Gianluca Allaria. 2002. A multilingual paradigm for automatic verb classification. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 207–214, Philadelphia, Pennsylvania. Association for Computational Linguistics.

Moschitti, Alessandro. 2006. Making tree kernels practical for natural language learning. In *Proceedings of EACL06*.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, March.

Pado, Sebastian and Mirella Lapata. 2009. Cross-lingual annotation projection of semantic roles. *Journal of Artificial Intelligence Research*, 36(1):307340.

Yarowsky, David and Grace Ngai. 2001. Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora. In *Second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies 2001*, pages 1–8, Pittsburgh, Pennsylvania. Association for Computational Linguistics.

Yarowsky, David, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the first international conference on Human language technology research*, pages 1–8, San Diego. Association for Computational Linguistics.

Zelenko, Dmitry, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106.

Zhang, Min, Jie Zhang, Jian Su, and Guodong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 825–832, Sydney, Australia. Association for Computational Linguistics.

Zhang, Zhu. 2004. Weakly-supervised relation classification for information extraction. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 581–588, Washington, D.C., USA. ACM.

Zhou, Guodong, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, page 434.

Zitouni, Imed and Radu Florian. 2008. Mention detection crossing the language barrier. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 600–609, Honolulu, Hawaii. Association for Computational Linguistics.

Evaluating N-gram based Evaluation Metrics for Automatic Keyphrase Extraction

Su Nam Kim, Timothy Baldwin
CSSE
University of Melbourne
sunamkim@gmail.com, tb@ldwin.net

Min-Yen Kan
School of Computing
National University of Singapore
kanmy@comp.nus.edu.sg

Abstract

This paper describes a feasibility study of n-gram-based evaluation metrics for automatic keyphrase extraction. To account for near-misses currently ignored by standard evaluation metrics, we adapt various evaluation metrics developed for machine translation and summarization, and also the R-precision evaluation metric from keyphrase evaluation. In evaluation, the R-precision metric is found to achieve the highest correlation with human annotations. We also provide evidence that the degree of semantic similarity varies with the location of the partially-matching component words.

1 Introduction

Keyphrases are noun phrases (NPs) that are representative of the main content of documents. Since they represent the key topics in documents, extracting good keyphrases benefits various natural language processing (NLP) applications such as summarization, information retrieval (IR) and question-answering (QA). Keyphrases can also be used in text summarization as semantic metadata (Barzilay and Elhadad, 1997; Lawrie et al., 2001; D'Avanzo and Magnini, 2005). In search engines, keyphrases supplement full-text indexing and assist users in creating good queries.

In the past, a large body of work on keyphrases has been carried out as an extraction task, utilizing three types of cohesion: (1) document cohesion, i.e. cohesion between documents and keyphrases (Frank et al., 1999; Witten et al., 1999; Matsuo and Ishizuka, 2004; Medelyan and Witten, 2006; Nguyen and Kan, 2007; Wan and Xiao, 2008); (2) keyphrase cohesion, i.e. cohesion among keyphrases (Turney, 2003); and (3) term cohesion, i.e. cohesion among terms in a keyphrase (Park et al., 2004).

Despite recent successes in keyphrase extraction (Frank et al., 1999; Turney, 2003; Park et al., 2004; Medelyan and Witten, 2006; Nguyen and Kan, 2007), current work is hampered by the inflexibility of standard metrics in evaluating different approaches. As seen in other fields, e.g. machine translation (MT) and multi-document summarization, the advent of standardized automatic evaluation metrics, combined with standardized datasets, has enabled easy comparison of systems and catalyzed the respective research areas. Traditionally, the evaluation of automatic keyphrase extraction has relied on the number of exact matches in author-assigned keyphrases and reader-assigned keyphrases. The main problem with this approach is that even small variants in the keyphrases are not given any credit. For example, given the gold-standard keyphrase *effective grid computing algorithm*, *grid computing algorithm* is a plausible keyphrase candidate and should be scored appropriately, rather than being naively evaluated as wrong. Additionally, author-assigned keyphrases and even reader-assigned keyphrases often have their own problems in this type of evaluation (Medelyan and Witten, 2006). For example, some keyphrases are often partly or wholly subsumed by other candidates or may not even occur in the document. Therefore, counting the exactly-matching candidates has been shown to be suboptimal (Jarmasz

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 572–580,
Beijing, August 2010

and Barriere, 2004).

Our goal in this paper is to evaluate the reliability of automatic evaluation metrics that better account for near-misses. Prior research based on semantic similarity (Jarmasz and Barriere, 2004; Mihalcea and Tarau, 2004; Medelyan and Witten, 2006) has taken the approach of using external resources such as large corpora, Wikipedia or manually-curated index words. While we acknowledge that these methods can help address the near-miss problem, they are impractical due to the effort required to compile the requisite resources for each individual evaluation exercise, and furthermore, the resources tend to be domain-specific. In order to design a cheap, practical and stable keyphrase evaluation metric, our aim is to properly account for these near-misses without reliance on costly external resources.

According to our analysis, the degree of semantic similarity of keyphrase candidates varies relative to the location of overlap. For example, the candidate *grid computing algorithm* has higher semantic similarity than *computing algorithm* with the gold-standard keyphrase *effective grid computing algorithm*. Also, *computing algorithm* is closer than *effective grid* to the same gold-standard keyphrase. From these observations, we infer that n-gram-based evaluation metrics can be applied to evaluating keyphrase extraction, but also that candidates with the same relative n-gram overlap are not necessarily equally good.

Our primary goal is to test the utility of n-gram based evaluation metrics to the task of keyphrase extraction evaluation. We test the following evaluation metrics: (1) evaluation metrics from MT and multi-document summarization (BLEU, NIST, METEOR and ROUGE); and (2) R-precision (Zesch and Gurevych, 2009), an n-gram-based evaluation metric developed specifically for keyphrase extraction evaluation which has yet to be evaluated against humans at the extraction task. Secondarily, we attempt to shed light on the bigger question of whether it is feasible to expect that n-gram-based metrics without access to external resources should be able to capture subtle semantic differences in keyphrase candidates. To this end, we experimentally verify the impact of lexical overlap of different types on keyphrase similarity, and use this as the basis for proposing a variant of R-precision.

In the next section, we present a brief primer on keyphrases. We then describe the MT and summarization evaluation metrics trialled in this research, along with R-precision, modified R-precision and a semantic similarity-based evaluation metric for keyphrase evaluation (Section 3). In Section 4, we discuss our gold-standard and candidate extraction method. We compare the evaluation metrics with human assigned scores for suitability in Section 5, before concluding the paper.

2 A Primer on Keyphrases

Keyphrases can be either simplex words (e.g. *query*, *discovery*, or *context-awareness*)[1] or larger N-bars/noun phrases (e.g. *intrusion detection*, *mobile ad-hoc network*, or *quality of service*). The majority of keyphrases are 1–4 words long (Paukkeri et al., 2008).

Keyphrases are normally composed of nouns and adjectives, but may occasionally contain adverbs (e.g. *dynamically allocated task*, or *partially observable Markov decision process*) or other parts of speech. They may also contain hyphens (e.g. *sensor-grouping* or *multi-agent system*) and apostrophes for possessives (e.g. *Bayes' theorem* or *agent's goal*).

Keyphrases can optionally incorporate PPs (e.g. *service quality* vs. *quality of service*). A variety of prepositions can be used (e.g. *incentive <u>for</u> cooperation, inequality <u>in</u> welfare, agent security <u>via</u> approximate policy*), although the genetive *of* is the most common.

Keyphrases can also be coordinated, either as simple nouns at the top level (e.g. *performance and scalability* or *group and partition*) or within more complex NPs or between N-bars (e.g. *history of past encounter and transitivity* or *task and resource allocation in agent system*).

When candidate phrases get too long, abbreviations also help to form valid keyphrases (e.g. *computer support collaborative work* vs. *CSCW*, or *partially observable Markov decision process* vs. *POMDP*).

[1] All examples in this section are taken from the data set outlined in Section 4.

3 Evaluation Metrics

There have been various evaluation metrics developed and validated for reliability in fields such as MT and summarization (Callison-Burch et al., 2009). While n-gram-based metrics don't capture systematic alternations in keyphrases, they do support partial match between keyphrase candidates and the reference keyphrases.

In this section, we first introduce a range of popular n-gram-based evaluation metrics from the MT and automatic summarization literature, which we naively apply to the task of keyphrase evaluation. We then present R-precision, an n-gram-based evaluation metric developed specifically for keyphrase evaluation, and propose a modified version of R-precision which weights n-grams according to their relative position in the keyphrase. Finally, we present a semantic similarity method.

3.1 Machine Translation and Summarization Evaluation Metrics

In this research, we experiment with four popular n-gram-based metrics from the MT and automatic summarization fields — BLEU, METEOR, NIST and ROUGE. The basic task performed by the respective evaluation metrics is empirical determination of *how good an approximation is string*$_1$ *of string*$_2$*?*, which is not far removed from the requirements of keyphrase evaluation. We briefly outline each of the methods below.

One subtle property of keyphrase evaluation is that there is no a priori preference for shorter keyphrases over longer keyphrases, unlike MT where shorter strings tend to be preferred. Hence, we use the longer NP as reference and the shorter NP as a translation, to avoid the length penalty in most MT metrics.[2]

BLEU (Papineni et al., 2002) is an evaluation metric for measuring the relative similarity between a candidate translation and a set of reference translations, based on n-gram composition. It calculates the number of overlapping n-grams between the candidate translation and the set of reference translations. In order to avoid having very short translations receive artificially high scores, BLEU adds a brevity penalty to the scoring equation.

METEOR (Agarwal and Lavie, 2008) is similar to BLEU, in that it measures string-level similarity between the reference and candidate translations. The difference is that it allows for more match flexibility, including stem variation and WordNet synonymy. The basic metric is based on the number of mapped unigrams found between the two strings, the total number of unigrams in the translation, and the total number of unigrams in the reference.

NIST (Martin and Przybocki, 1999) is once again similar to BLEU, but integrates a proportional difference in the co-occurrences for all n-grams while weighting more heavily n-grams that occur less frequently, according to their information value.

ROUGE (Lin and Hovy, 2003) — and its variants including ROUGE-N and ROUGE-L — is similarly based on n-gram overlap between the candidate and reference summaries. For example, ROUGE-N is based on co-occurrence statistics, using higher-order n-grams ($n > 1$) to estimate the fluency of summaries. ROUGE-L uses longest common subsequence (LCS)-based statistics, based on the assumption that the longer the substring overlap between the two strings, the greater the similar Saggion et al. (2002). ROUGE-W is a weighted LCS-based statistic that prioritizes consecutive LCSes. In this research, we experiment exclusively with the basic ROUGE metric, and unigrams (i.e. ROUGE-1).

3.2 R-precision

In order to analyze near-misses in keyphrase extraction evaluation, Zesch and Gurevych (2009) proposed R-precision, an n-gram-based evaluation metric for keyphrase evaluation.[3] R-precision contrasts with the majority of previous work on keyphrase extraction evaluation, which has used semantic similarity based on external resources

[2] While we don't present the numbers in this paper, the results were lower for the MT evaluation metrics without this reordering of the reference and candidate keyphrases.

[3] Zesch and Gurevych's R-precision has nothing to do with the information retrieval evaluation metric of the same name, where P@N is calculated for N equal to the number of relevant documents.

(Jarmasz and Barriere, 2004; Mihalcea and Tarau, 2004; Medelyan and Witten, 2006). As our interest is in fully automated evaluation metrics which don't require external resources and are domain independent (for maximal reproducibility of results), we experiment only with R-precision in this paper.

R-precision is based on the number of overlapping words between a keyphrase and a candidate, as well as the length of each. The metric differentiates three types of near-misses: *INCLUDE*, *PARTOF* and *MORPH*. The first two types are based on an n-gram approach, while the third relies on lexical variation. As we use stemming, in line with the majority of previous work on keyphrase extraction evaluation, we focus exclusively on the first two cases, namely *INCLUDE*, and *PARTOF*. The final score returned by R-precision is:

$$\frac{\text{number of overlapping word(s)}}{\text{length of keyphrase/candidate}}$$

where the denominator is the longer of the keyphrase and candidate.

Zesch and Gurevych (2009) evaluated R-precision over three corpora (Inspec, DUC and SP) based on 566 non-exact matching candidates. In order to evaluate the human agreement, they hired 4 human annotators to rate the near-miss candidates, and reported agreements of 80% and 44% for the INCLUDE and PARTOF types, respectively. They did not, however, perform holistic evaluation with human scores to verify its reliability in full system evaluation. This is one of our contributions in this paper.

3.3 Modified R-precision

In this section, we describe a modification to R-precision which assigns different weights for component words based on their position in the keyphrase (unlike R-precision which assigns the same score for each matching component word). The head noun generally encodes the core semantics of the keyphrase, and as a very rough heuristic, the further a word is from the head noun, the less semantic import on the keyphrase it has. As such, modified R-precision assigns a score to each component word relative to its position as $CW = \frac{1}{N-i+1}$ where N is the number of component words in the keyphrase and i is the position of the component word in the keyphrase (1 = leftmost word).

For example, *AB* and *BC* from *ABC* would be scored as $\frac{\frac{1}{3}+\frac{1}{2}}{\frac{1}{3}+\frac{1}{2}+\frac{1}{1}} = \frac{5}{11}$ and $\frac{\frac{1}{2}+\frac{1}{1}}{\frac{1}{3}+\frac{1}{2}+\frac{1}{1}} = \frac{9}{11}$, respectively. Thus, with the keyphrase *effective grid computing algorithm* and candidates *effective grid*, *grid computing* and *computing algorithm*, modified R-precision assigns different scores for each candidate (*computing algorithm* $>$ *grid computing* $>$ *effective grid*). In contrast, the original R-precision assigns the same score to all candidates.

3.4 Semantic Similarity

In Jarmasz and Barriere (2004) and Mihalcea and Tarau (2004), the authors used a large data set to compute the semantic similarity of two NPs to assign partial credits for semantically similar candidate keyphrases. To simulate these methods, we adopted the distributional semantic similarity using web documents. That is, we computed the similarity between a keyphrase and its substring by cosine measure over collected the snippets from Yahoo! BOSS.[4] We use the computed similarity as our score for near-misses.

4 Data

4.1 Data Collection

We constructed a keyphrase extraction dataset using papers across 4 different categories[5] of the ACM Digital Library.[6] In addition to author-assigned keyphrases provided as part of the ACM Digital Library, we generated reader-assigned keyphrases by assigning 250 students 5 papers each, a list of candidate keyphrases (see below for details), and standardized instructions on how to assign keyphrases. It took them an average of 15 minutes to annotate each paper. This is the same

[4] http://developer.yahoo.com/search/boss/

[5] C2.4 (Distributed Systems), H3.3 (Information Search and Retrieval), I2.11 (Distributed Artificial Intelligence – Multiagent Systems) and J4 (Social and Behavioral Sciences – Economics).

[6] http://portal.acm.org/dl.cfm

	Author	Reader	Total
Total	1298/1305	3110/3221	3816/3962
NPs	937	2537	3027
Average	3.85/4.01	12.44/12.88	15.26/15.85
Found	769	2509	2864

Table 1: Details of the keyphrase dataset

(**Rule1**) NBAR = (NN*|JJ*)*(NN*)
e.g. *complexity, effective algorithm, distributed web-service discovery architecture*
(**Rule2**) NBAR `IN` NBAR
e.g. *quality of service, sensitivity of VOIP traffic, simplified instantiation of zebroid*

Table 2: Regular expressions for candidate selection

document collection and set of keyphrase annotations as was used in the SemEval 2010 keyphrase extraction task (Kim et al., 2010).

Table 1 shows the details of the final dataset. The numbers after the slashes indicate the number of keyphrases after including alternate keyphrases based on *of*-PPs. Despite the reliability of author-assigned keyphrases discussed in Medelyan and Witten (2006), many author-assigned keyphrases and some reader-assigned keyphrases are not found verbatim in the source documents because: (1) many of them are substrings of the candidates or vice versa (about 75% of the total keyphrases are found in the documents); and (2) our candidate selection method does not extract keyphrases in forms such as coordinated NPs or adverbial phrases.

4.2 Candidate Selection

During preprocessing, we first converted the PDF versions of the papers into text using `pdftotext`. We then lemmatized and POS tagged all words using `morpha` and the `Lingua` POS tagger. Next, we applied the regular expressions in Table 2 to extract candidates, based on Nguyen and Kan (2007). Finally, we selected candidates in terms of their frequency: simplex words with frequency ≥ 2 and NPs with frequency ≥ 1. We observed that for reader-assigned keyphrases, NPs were often selected regardless of their frequency in the source document. In addition, we allowed variation in the possessive form, noun number and abbreviations.

Rule1 detects simplex nouns or N-bars as candidates. *Rule2* extracts N-bars with post-modifying PPs. In Nguyen and Kan (2007), *Rule2* was not used to additionally extract N-bars inside modifying PPs. For example, our rules extract not only *performance of grid computing* as a candidate, but also *grid computing*. However, we did not extend the candidate selection rules to cover NPs including adverbs (e.g. *partially-observable Markov decision process*) or conjunctions (e.g. *behavioral evolution and extrapolation*), as they are rare.

4.3 Human Assigned Score

We hired four graduate students working in NLP to assign human scores to substrings in the gold-standard data. The scores are between 0 and 4 (0 means no semantic overlap between a NP and its substring, while 4 means semantically indistinguishable).

We broke down the candidate–keyphrases pairs into subtypes, based on where the overlap occurs relative to the keyphrase (e.g. *ABCD*): (1) *Head*: the candidate contains the head noun of the keyphrase (e.g. *CD*); (2) *First*: the candidate contains the first word of the keyphrase (e.g. *AB*); and (3) *Middle*: the candidate overlaps with the keyphrase, but contains neither its first word nor its head word (e.g. *BC*). The average human scores are 1.94 and 2.11 for *First* and *Head*, respectively, when the candidate is shorter, while they are 2.00, 1.89 and 2.15 for *First*, *Middle*, and *Head*, respectively when the candidate is longer. Note that we did not have *Middle* instances with candidates as the shorter string. The scores are slightly higher for the keyphrases as substrings than for the candidates as substrings.

5 Correlation

To check the feasibility of metrics for keyphrase evaluation, we checked the Spearman rank correlation between the machine-generated score and the human-assigned score for each keyphrase–candidate pairing.

As the percentage of annotators who agree on the exact score is low (i.e. 2 subjects agree ex-

		Human	R-precision		BLEU	METEOR	NIST	ROUGE	Semantic Similarity
			Orig	Mod					
Average	All	.4506	.4763	.2840	.3250	.3246	.3366	.3246	.2116
	$L \leq 4$	.4510	.5264	.2806	.3242	.3238	.3369	.3240	.2050
	$L \leq 3$	.4551	.4834	.2893	.3439	.3437	.3584	.3437	.1980
Majority	All	.4603	.4763	.3438	.3407	.3403	.3514	.3404	.2224
	$L \leq 4$	.4604	.5264	.3434	.3423	.3421	.3547	.3422	.2168
	$L \leq 3$	.4638	.4838	.3547	.3679	.3675	.3820	.3676	.2123

Table 3: Rank correlation between humans and the different evaluation metrics, based on the human average (top half) and majority (bottom half)

		Human	R-precision		BLEU	METEOR	NIST	ROUGE
			Orig	Mod				
Location	First	**.5508**	**.5032**	**.5033**	.3844	.3844	.4057	.3844
	Middle	**.5329**	**.5741**	**.5988**	**.4669**	**.4669**	.4055	**.4669**
	Head	**.3783**	**.4838**	**.4838**	.3865	.3860	.3780	.3864
Complexity	Simple	.4452	**.4715**	.2790	.3653	.3445	.3527	.3445
	PP	**.4771**	**.4814**	.1484	.3367	.3122	.3443	.3123
	CC	.3645	.3810	.3140	.3748	.3446	.3384	.3748
POS	AdjN	**.4616**	**.4844**	.3507	.3147	.3132	.3115	.3133
	NN	.4467	**.4586**	.2581	.3321	.3321	.3488	.3322

Table 4: Rank correlation between human average judgments and n-gram-based metrics

actly on 55%-70% of instances, 3 subjects agree exactly on 25%-35% of instances), we require a method for combining the annotations. We experiment with two combination methods: majority and average. The majority is simply the label with the majority of annotations associated with it; in the case of a tie, we break the tie by selecting that annotation which is closest to the median. The average is simply the average score across all annotators.

5.1 Overall Correlation with Human Scores

Table 3 presents the correlations between the human scores (acting as an upper bound for the task), as well as those between human scores with machine-generated scores. We first present the overall results, then results over the subset of keyphrases of length 4 words or less, and also 3 words or less. We present the results for the annotator average and majority in top and bottom half, respectively, of the table.

To compute the correlation between the human annotators, we used leave-one-out cross-validation, holding out one annotator, and comparing them to the combination of the remaining annotators (using either the majority or average method to combine the remaining annotations). This was repeated across all annotators, and the Spearman's ρ was averaged across the annotators.

Overall, we found that R-precision achieved the highest correlation with humans, above the inter-annotator correlation in all instances. That is, based on the evaluation methodology employed, it is performing slightly above the average level of a single annotator. The relatively low inter-annotator correlation is, no doubt, due to the difficulty of the task, as all of our near-misses have 2 or more terms, and the annotators have to make very fine-grained, and ultimately subjective, decisions about the true quality of the candidate.

Comparing the n-gram-based methods with the semantic similarity-based method, the n-gram-based metrics achieved higher correlations across the board, with BLEU, METEOR, NIST and ROUGE all performing remarkably consistently, but well

		Human	R-precision		BLEU	METEOR	NIST	ROUGE
			Orig	Mod				
LOCATION	First	**.5642**	**.5162**	**.5163**	.4032	.4032	.4297	.4032
	Middle	**.5510**	**.4991**	**.5320**	.4175	.4175	.3653	.4175
	Head	.4147	**.5073**	**.5074**	.4156	.4153	.4042	.4156
COMPLEXITY	Simple	.4580	**.4869**	.3394	.3653	.3651	.3715	.3651
	PP	**.4715**	**.5068**	.3724	.3367	.3367	.3652	.3367
	CC	**.5777**	**.5513**	.3841	**.5745**	**.5571**	**.5600**	**.5745**
POS	AdjN	.4501	**.4861**	.3968	.3266	.3251	.3246	.3252
	NN	**.4631**	**.4733**	.3244	.3499	.3499	.3648	.3500

Table 5: Rank correlation between human majority and n-gram-based metrics

below the level of R-precision. Due to the markedly lower performance of the semantic similarity-based method, we do not consider it for the remainder of our experiments. A general finding was that as the length of the keyphrase (L) got longer, the correlation tended to be higher across all n-gram-based metrics.

One disappointment at this stage is that the results for modified R-precision are well below those of the original, especially over the average of the human annotators.

5.2 Correlation with Different NP Subtypes

To get a clearer sense of how the different evaluation metrics are performing, we broke down the keyphrases according to three syntactic subclassifications: (1) the location of overlap (see Section 4.3); (2) the complexity of the NP (does the keyphrase contain a preposition [PP], a conjunction [CC] or neither a preposition nor a conjunction [Simple]?); and (3) the word class sequence of the keyphrase (is the keyphrase an NN [NN] or an AdjN sequence [AdjN]?). We present the results in Tables 4 and Table 4 for the human average and majority, respectively, presenting results in **boldface** when the correlation for a given method is higher than for that same method in our holistic evaluation in Table 3 (i.e. .4506 and .4603, for the average and majority human scores, respectively).

All methods, including inter-annotator correlation, improve in raw numbers over the subsets of the data based on overlap location, indicating that the data was partitioned into more internally-consistent subsets. Encouragingly, modified R-precision equalled or bettered the performance of the original R-precision over each subset of the data based on overlap location. Where modified R-precision appears to fall down most noticeably is over keyphrases including prepositions, as our assumption about the semantic import based on linear ordering clearly breaks down in the face of post-modifying PPs. It is also telling that it does worse over noun–noun sequences than adjective–noun sequences. In being agnostic to the effects of syntax, the original R-precision appears to benefit overall. Another interesting effect is that the performance of BLEU, METEOR and ROUGE is notably better over candidates which match with non-initial and non-final words in the keyphrase.

We conclude from this analysis that keyphrase scoring should be sensitive to overlap location. Furthermore, our study also shows that n-gram-based MT and summarization metrics are surprisingly adept at capturing partial matches in keyphrases, despite them being much shorter than the strings they are standardly applied to. More compellingly, we found that R-precision is the best overall performer, and that it matches the performance of our human annotators across the board. This is the first research to establish this fact. Our findings for modified R-precision were more sobering, but its location sensitivity was shown to improve over R-precision for instances of overlap in the middle or with the head of the keyphrase.

6 Conclusion

In this work, we have shown that preexisting n-gram-based evaluation metrics from MT, summarization and keyphrase extraction evaluation are able to handle the effects of near-misses, and that R-precision performs at or above the average level of a human annotator. We have also shown that a semantic similarity-based method which uses web data to model distributional similarity performed below the level of all of the n-gram-based methods, despite them requiring no external resources (web or otherwise). We proposed a modification to R-precision based on the location of match, but found that while it could achieve better performance over certain classes of keyphrases, its net effect was to drag the performance of R-precision down. Other methods were found to be remarkably consistent across different subtypes of keyphrase.

Acknowledgements

Many thanks to the anonymous reviewers for their insightful comments. We wish to acknowledge the generous funding from National Research Foundation grant R 252-000-279-325 in supporting Min-Yen Kan's work.

References

Abhaya Agrwal and Alon Lavie. METEOR, M-BLEU and M-TER: Evaluation Metrics for High-Correlation with Human Rankings of Machine Translation Output. In *Proceedings of ACL Workshop on Statistical Machine Translation*. 2008.

Ken Barker and Nadia Corrnacchia. Using noun phrase heads to extract document keyphrases. In *Proceedings of BCCSCSI : Advances in Artificial Intelligence*. 2000, pp.96–103.

Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. In *Proceedings of ACL/EACL Workshop on Intelligent Scalable Text Summarization*. 1997, pp. 10–17.

Chris Callison-Burch, Philipp Koehn, Christof Monz and Josh Schroeder. Proceedings of 4th Workshop on Statistical Machine Translation. 2009.

Ernesto D'Avanzo and Bernado Magnini. A Keyphrase-Based Approach to Summarization: the LAKE System at DUC-2005. In *Proceedings of DUC*. 2005.

Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin and Craig G. Nevill-Manning. Domain Specific Keyphrase Extraction. In *Proceedings of IJCAI*. 1999, pp.668–673.

Mario Jarmasz and Caroline Barriere. Using semantic similarity over Tera-byte corpus, compute the performance of keyphrase extraction. In *Proceedings of CLINE*. 2004.

Su Nam Kim, Olena Medelyan, Min-Yen Kan and Timothy Baldwin. SemEval-2010 Task 5: Automatic Keyphrase Extraction from Scientific Articles. In *Proceedings of SemEval-2: Evaluation Exercises on Semantic Evaluation*. to appear.

Dawn Lawrie, W. Bruce Croft and Arnold Rosenberg. Finding Topic Words for Hierarchical Summarization. In *Proceedings of SIGIR*. 2001, pp. 349–357.

Chin-Yew Lin and Edward H. Hovy. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *In Proceedings of HLT-NAACL*. 2003.

Alvin Martin and Mark Przybocki. The 1999 NIST Speaker Recognition Evaluation, Using Summed Two-Channel Telephone Data for Speaker Detection and Speaker Tracking. In *Proceedings of EuroSpeech*. 1999.

Yutaka Matsuo and Mitsuru Ishizuka. Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information. *International Journal on Artificial Intelligence Tools*. 2004, 13(1), pp. 157–169.

Olena Medelyan and Ian Witten. Thesaurus based automatic keyphrase indexing. In *Proceedings of ACM/IEED-CS JCDL*. 2006, pp. 296–297.

Rada Mihalcea and Paul Tarau. TextRank: Bringing Order into Texts. In *Proceedings of EMNLP 2004*. 2004, pp. 404–411.

Guido Minnen, John Carroll and Darren Pearce. Applied morphological processing of English. *NLE*. 2001, 7(3), pp. 207–223.

Thuy Dung Nguyen and Min-Yen Kan. Key phrase Extraction in Scientific Publications. In *Proceeding of ICADL*. 2007, pp. 317–326.

Sebastian Padó, Michel Galley, Dan Jurafsky and Christopher D. Manning. Textual Entailment Features for Machine Translation Evaluation. In *Proceedings of ACL Workshop on Statistical Machine Translation*. 2009, pp. 37–41.

Kishore Papineni, Salim Roukos, Todd Ward and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*. 2001, pp. 311–318.

Youngja Park, Roy J. Byrd and Branimir Boguraev. Automatic Glossary Extraction Beyond Terminology Identification. In *Proceedings of COLING*. 2004, pp. 48–55.

Mari-Sanna Paukkeri, Ilari T. Nieminen, Matti Polla and Timo Honkela. A Language-Independent Approach to Keyphrase Extraction and Evaluation. In *Proceedings of COLING*. 2008, pp. 83–86.

Horacio Saggion, Dragomir Radev, Simon Teufel, Wai Lam and Stephanie Strassel. Meta-evaluation of Summaries in a Cross-lingual Environment using Content-based Metrics. In *Proceedings of COLING*. 2002, pp. 1–7.

Peter Turney. Coherent keyphrase extraction via Web mining. In *Proceedings of IJCAI*. 2003, pp. 434–439.

Xiaojun Wan and Jianguo Xiao. CollabRank: towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of COLING*. 2008, pp. 969–976.

Ian Witten, Gordon Paynter, Eibe Frank, Car Gutwin and Craig Nevill-Manning. KEA:Practical Automatic Key phrase Extraction. In *Proceedings of ACM conference on Digital libraries*. 1999, pp. 254–256.

Torsten Zesch and Iryna Gurevych. Approximate Matching for Evaluating Keyphrase Extraction. *International Conference on Recent Advances in Natural Language Processing*. 2009.

Improving the Quality of Text Understanding by Delaying Ambiguity Resolution

Doo Soon Kim
Dept. of Computer Science
University of Texas
onue5@cs.utexas.edu

Ken Barker
Dept. of Computer Science
University of Texas
kbarker@cs.utexas.edu

Bruce Porter
Dept. of Computer Science
University of Texas
porter@cs.utexas.edu

Abstract

Text Understanding systems often commit to a *single best* interpretation of a sentence before analyzing subsequent text. This interpretation is chosen by resolving ambiguous alternatives to the one with the highest confidence, given the context available at the time of commitment. Subsequent text, however, may contain information that changes the confidence of alternatives. This may especially be the case with multiple redundant texts on the same topic. Ideally, systems would delay choosing among ambiguous alternatives until more text has been read.

One solution is to maintain multiple candidate interpretations of each sentence until the system acquires disambiguating evidence. Unfortunately, the number of alternatives explodes quickly. In this paper, we propose a *packed graphical (PG) representation* that can efficiently represent a large number of alternative interpretations along with dependencies among them. We also present an algorithm for combining multiple PG representations to help resolve ambiguity and prune alternatives when the time comes to commit to a single interpretation.

Our controlled experiments show that by delaying ambiguity resolution until multiple texts have been read, our prototype's accuracy is higher than when committing to interpretations sentence-by-sentence.

1 Introduction

A typical text understanding system confronts ambiguity while parsing, mapping words to concepts and formal relations, resolving co-references, and integrating knowledge derived from separate sentences or texts. The system discards many candidate interpretations to avoid combinatorial explosion. Commonly, after reading each sentence, a system will commit to its top ranked interpretation of the sentence before reading the next.

If a text understanding system could postpone committing to an interpretation without being swamped by a combinatorial explosion of alternatives, its accuracy would almost surely improve. This intuition follows from the observation that text is redundant in at least two ways. First, within a single coherent text (about the same entities and events), each sentence informs the interpretation of its neighbors. Second, within a corpus of texts on the same topic, the same information is expressed in different surface forms, ambiguous in different ways. Related fields, such as Information Extraction, exploit textual redundancy to good effect, and perhaps text understanding can as well.

One approach is for the text understanding system to maintain multiple complete candidate interpretations. After reading each sentence, for example, the system would retain a beam of the n-best interpretations of the sentence. While this approach avoids a combinatorial explosion (for reasonable values of n), several problems remain. First, because the beam width is limited, the system may still discard correct interpretations before benefiting from the extra context from related text. Second, enumeration of the candidate interpreta-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 581–589,
Beijing, August 2010

tions does not represent the dependencies among them. For example, there may be multiple candidate word senses and semantic roles for a given sentence, but sense alternatives might be dependent on role selection (and vice-versa). The set of reasonable interpretations may be a subset of all combinations. Finally, maintaining distinct interpretations does not contribute to addressing the problem of combining evidence to narrow down alternatives and ultimately select a single best interpretation of a text.

This paper addresses these three problems. We propose an approach that postpones committing to an interpretation of a text by representing ambiguities and the dependencies among them. There may still be combinatorial growth in the set of alternative interpretations, but they are represented only intensionally, using a packed representation, which maintains alternatives while avoiding enumerating them. We also propose an algorithm for updating and pruning the packed representation as more sentences and texts are read.

We evaluate our approach by comparing two reading systems: a baseline system that commits to its best interpretation after each sentence, and our prototype system that uses a packed representation to maintain all interpretations until further reading enables it to prune. For this initial proof of concept, we use a small corpus of redundant texts. The results indicate that our approach improves the quality of text interpretation by preventing aggressive pruning while avoiding combinatorial explosion.

In the following sections, we first describe our target semantic representation of the interpretation of sentences. We then present the details of our *packed graphical representation (PG representation)* and our algorithm to resolve ambiguities in the PG representations as disambiguating evidence from subsequent text accrues. We describe the architecture of a prototype that produces PG representations for text and implements the disambiguating algorithm. Finally, we present the results from controlled experiments designed to compare the accuracy of the prototype to a baseline system that prunes more aggressively.

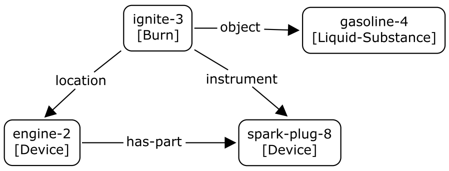


Figure 1: The target semantic graph representation for S1

2 Target semantic representation

Our target representation is a semantic graph in which nodes are words and the ontological types to which they map. Edges are semantic relations corresponding either to function words or syntactic relations in the sentence's parse.

Fig. 1 shows the target semantic representation for the following simple sentence:

S1: *An engine ignites gasoline with its spark plug.*

3 PG representation

Alternative semantic interpretations for a sentence can be captured with a single PG representation with ambiguities represented as local alternatives. Because candidate representations are often structurally similar, a PG representation can significantly compress the representation of alternatives.

Fig. 2 shows the PG representation of alternate interpretations of S1 (PG1). The different types of ambiguity captured by the PG representation are as follows.

3.1 Word-Type ambiguity

In PG1, the node engine-2a corresponds to the word "engine" in S1. Its annotation [LIVING-ENTITY .3 | DEVICE .7] captures the mapping to either LIVING-ENTITY (probability 0.3) or DEVICE (probability 0.7). The PG representation does not presume a particular uncer-

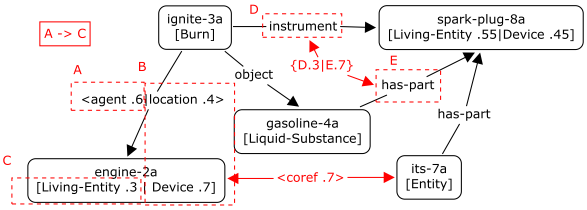


Figure 2: The PG representation for S1 (PG1)

tainty formalism. Any formalism, (Dempster-Shafer theory (Pearl, 1988), Markov Logic Networks (Richardson and Domingos, 2006), etc.) could be used.

3.2 Semantic Relation ambiguity

In PG1, the edge label <agent .6 | location .4> from ignite-3a to engine-2a says that the engine is either *agent* or *location* of the ignition.

3.3 Structural ambiguity

In PG1, edges D and E are alternatives corresponding to the different prepositional phrase attachments for "with its spark plug" (to ignite-3a or gasoline-4a). The annotation {D .3 | E .7} says that the choices are mutually exclusive with probabilities of 0.3 and 0.7.

3.4 Co-reference ambiguity

Co-reference of nodes in a PG representation is captured using a "co-reference" edge. In PG1, the edge labeled <coref .7> represents the probability that engine-2a and its-7a are co-referent.

In addition to storing ambiguities explicitly, the PG representation also captures dependencies among alternatives.

3.5 Simple dependency

The existence of one element in the graph depends on the existence of another element. If subsequent evidence suggests that an element is incorrect, its dependents should be pruned. For example, the dependency A → C, means that if LIVING-ENTITY is ultimately rejected as the type for engine-2a, the agent relation should be pruned.

3.6 Mutual dependency

Elements of a mutual dependency set are mutually confirming. Evidence confirming or rejecting an element also confirms or rejects other elements in the set. In the example, the box labeled B says that (engine-2a type DEVICE) and (ignite-3a location engine-2a) should both be confirmed or pruned when either of them is confirmed or pruned.

Formally, the PG representation is a structure consisting of (a) *semantic triples* – e.g., (ignite-3a type BURN), (b) *macros* – e.g., the symbol A refers to (ignite-3a agent engine-2a), and (c) *constraints* – e.g., A depends on C.

4 Combining PG representations

Maintaining ambiguity within a PG representation allows us to delay commitment to an interpretation until disambiguating evidence appears. For any text fragment that results in a PG representation (PGa) containing ambiguity, there may exist other text fragments that are partly redundant, but result in a less ambiguous (or differently ambiguous) representation (PGb). PGb can be used to adjust confidences in PGa. Enough such evidence allows us to prune unlikely interpretations, ultimately disambiguating the original representation.

For example, sentence S3 does not have sufficient context to disambiguate between the MOTOR sense of "engine" and the VEHICLE sense (as in *locomotive*).

S3: *General Electric announced plans this week for their much anticipated new engine.*

The PG3 representation for S3 (PG3) would maintain the ambiguous representation (with confidences for each sense based on prior probabilities, for example). On subsequently encountering sentence S4, a Lesk-based word sense disambiguation module (as in our prototype) would produce a PG4 with a strong preference for the locomotive sense of "engine", given the more specific context of S4.

S4: *The announcement comes to the relief of many in the railway industry looking to replace the engines in their aging locomotive fleets.*

To use PG4 to help disambiguate PG3, we need to align PG3 and PG4 semantically and merge their conflict sets. (In the simple example, the conflict sets for the word "engine" might be [MOTOR .5 | VEHICLE .5] in PG3 and [MOTOR .2 | VEHICLE .8] in PG4).

Algorithm 1 describes how two PG representations can be combined to help resolve their ambiguities. The algorithm identifies their isomorphic subgraphs (redundant portions of the interpretations) and uses the information to disambiguate their ambiguities. For illustration, we will step through Algorithm 1, merging PG1 (Fig. 2) with

Algorithm 1 Disambiguating PG representations

Input : PG1, PG2

Output: new PG representation

1. *Identify semantically aligned parts between PG1 and PG2.* Use graph matching to identify alignments (redundant portions) between PG1 and PG2: align nodes with the same base word or with taxonomically related types; from the node alignments, align identical types as type alignments; align relations if the relations are the same and their head and tail nodes have been aligned.

2. *Use alignments to disambiguate PG1 and PG2.* With the available information (the confidence scores and the constraints in PG1 and PG2 and the alignments between them), use joint inference to calculate the confidence score of each candidate interpretation. If the confidence score of one interpretation becomes much higher than competing ones, the interpretation is chosen while the others are discarded.

3. *Combine the disambiguated PG1 and PG2 into one PG representation using the alignments identified in the first step.*

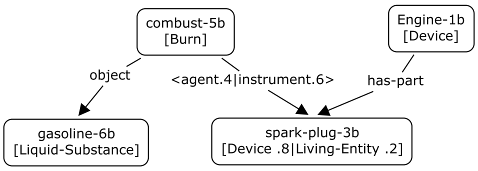


Figure 3: PG representation for S2, *"The engine's spark plug combusts gasoline."*

PG2 (Fig. 3).

1. The graph matcher identifies alignments between PG1 and PG2. Type alignments include (engine-2a[Device], Engine-1b[Device]), (spark-plug-8a[Living-Entity], spark-plug-3b[Living-Entity]). Relation alignments include ((combust-5b instrument spark-plug-3b), (ignite-3 instrument spark-plug-8)), ((ignite-3a instrument spark-plug-8a) (combust-5b instrument spark-plug-3b)).

2. In this example, when two interpretations are aligned, we simply add their confidence scores. (We are currently incorporating *Alchemy* (Richardson and Domingos, 2006) in the prototype system to do the joint inference). For example, aligning engine-2a with Engine-1b results in a score of 1.7 for Device (1 + .7). The confidence score of Living-Entity in engine-2a is unchanged at .3. Since the resulting score for Device is much higher than [1] the score for Living-Entity, Living-Entity is discarded. Deleting Living-Entity causes deletion of the *agent* edge between ignite-3a and engine-2a due to the dependency constraint A → C.

3. The disambiguated PG1 and PG2 are merged into a single PG representation (PG1+2) based on the alignments. Any remaining ambiguity persists in PG1+2, possibly to be resolved with another sentence.

5 Prototype system

5.1 Parser

Our prototype system uses the Stanford Parser (Klein and Manning, 2003). To capture structural ambiguity for our experiments, we manually edited the parser output by adding corrections as alternatives wherever the parse tree was incorrect. This gave a syntactic PG representation with both incorrect and correct alternatives. We gave the original, incorrect alternatives high confidence scores and the added, correct alternatives low scores, simulating a parser pruning correct interpretations in favor of incorrect ones with higher confidence scores. The syntactic PG for S1 is shown in Fig. 4. We have recently designed a modification to the Stanford Parser to make it produce syntactic PG representations natively, based on the complete chart built during parsing.

5.2 Semantic Interpreter

The semantic interpreter assigns types to nodes in the syntactic PG representation and semantic relations to the edges.

Type ambiguity. Types and confidence scores are assigned to words using SenseRelate (Patwardhan et al., 2005), WSD software based on the

[1] In our prototype, we set the pruning threshold at $\frac{1}{3}\times$the score of the top-scored interpretation.

Lesk Algorithm (Lesk, 1986). Assigned senses are then mapped to our *Component Library* ontology (Barker et al., 2001) using its built-in WordNet mappings.

Relational ambiguity. Semantic relations are assigned to the dependency relations in the syntactic PG representation according to semantic interpretation rules. Most rules consider the head and tail types as well as the dependency relation, but do not produce confidence scores. Our prototype scores candidates equally. We plan to incorporate a more sophisticated scoring method such as (Punyakanok et al., 2005).

Structural ambiguity. Parse ambiguities (such as PA vs. PB in Fig. 4) are converted directly to structural ambiguity representations (D vs. E in Fig. 2) in the semantic PG representation.

Simple Dependency. A dependency is installed between a type t for word w and a semantic relation r when (1) r is produced by a rule based on t and (2) r is dependent on no other candidate type for w. In Fig. 2, a dependency relation is installed from A to C, because (1) LIVING-ENTITY in engine-2a was used in the rule assigning *agent* between ignite-3a and engine-2a and (2) the assignment of *agent* is not dependent on DEVICE, the other candidate type of engine-2a.

Mutual dependency. If multiple interpretations depend on one another, a mutual dependency set is created to include them.

5.3 PG Merger

The PG Merger implements Algorithm 1 to combine PG representations. The PG representation for each sentence is merged with the cumulative PG from previous sentences. The global PG representation integrates sentence-level PG representations to the extent that they align semantically. In the worst case (completely unrelated sentences), the global PG representation would simply be the union of individual PG representations. The extent to which the global PG is more coherent reflects redundancy and semantic overlap in the sentences.

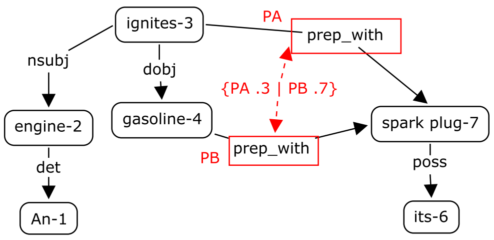


Figure 4: Syntactic PG representation for S1, capturing the PP-attachment ambiguity of “with its spark plug”.

Original Text Hearts pump blood through the body. Blood carries oxygen to organs throughout the body. Blood leaves the heart, then goes to the lungs where it is oxygenated. The oxygen given to the blood by the lungs is then burned by organs throughout the body. Eventually the blood returns to the heart, depleted of oxygen.
Paraphrase The heart begins to pump blood into the body. The blood first travels to the lungs, where it picks up oxygen. The blood will then be deposited into the organs, which burn the oxygen. The blood will then return to the heart, where it will be lacking oxygen, and start over again.

Figure 5: The original text and a paraphrase

6 Experiment 1

We first wanted to evaluate our hypothesis that Algorithm 1 can improve interpretation accuracy over multiple redundant texts. We manually generated ten redundant texts by having volunteers rewrite a short, tutorial text, using Amazon Turk (http://mturk.com) [2] The volunteers had no knowledge of the purpose of the task, and were asked to rewrite the text using “different” language. Fig. 5 shows the original text and one volunteer’s rewrite. The total number of sentences over the ten texts was 37. Average sentence length was 14.5 words.

6.1 Evaluation Procedure

We ran two systems over the ten texts. The baseline system commits to the highest scoring consistent interpretation after each sentence. The prototype system produces an ambiguity-preserving

[2] We ultimately envision a system whose task is to develop a model of a particular topic by interpreting multiple texts. Such a system might be given a cluster of documents or use its own information retrieval to find similar documents given a tutorial text.

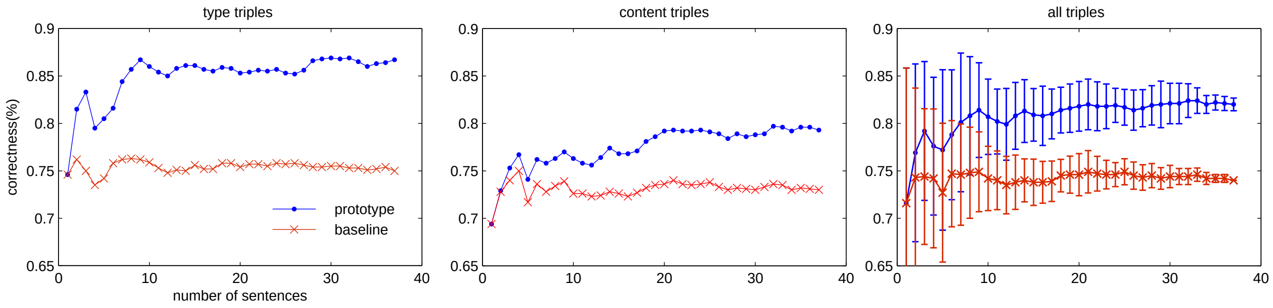


Figure 6: Correctness scores for the prototype vs. baseline system on (a) type triples (word sense assignment), (b) content triples (semantic relations) and (c) all triples (with standard deviation).

PG representation. For each sentence, the prototype's PG Merger merges the PG of the sentence with the merged PG of the previous sentences. After N sentences (varying N from 1..37), the system is forced to commit to the highest scoring consistent interpretation in the merged PG. For N=1 (commit after the first sentence), both the baseline and prototype produce the same result. For N=2, the baseline produces the union of the highest scoring interpretations for each of the first two sentences. The prototype produces a merged PG for the first two sentences and then prunes to the highest scoring alternatives.

At each value of N, we measured the correctness of the interpretations (the percentage of correct semantic triples) for each system by comparing the committed triples against human-generated gold standard triples.

We repeated the experiment ten times with different random orderings of the 37 sentences, averaging the results.

6.2 Evaluation result

Fig. 6 shows that both type assignment and semantic relation assignment by the prototype improve as the system reads more sentences. This result confirms our hypothesis that delaying commitment to an interpretation resolves ambiguities better by avoiding overly aggressive pruning.

To determine an upper bound of correctness for the prototype, we inspected the PG representations to see how many alternative sets contained the correct interpretation even if not the highest scoring alternative. This number is different from the correctness score in Fig. 6, which is the percentage of gold standard triples that are the highest scoring alternatives in the merged PG.

	baseline	prototype
nodes w/ the correct type	76	**91**
edges w/ the correct relation	74	**88**

Table 1: Percentage of nodes and edges containing the correct types and semantic relations in the baseline and the prototype for all 37 sentences.

Table. 1 shows that 91% of the nodes in the PG contain the correct type (though not necessarily the highest scoring). 88% of the edges contain the correct semantic relations among the alternatives. In contrast, the baseline has pruned away 24% of the correct types and 26% of the correct semantic relations.

7 Experiment 2

Our second experiment aims to evaluate the claim that the prototype can efficiently manage a large number of alternative interpretations. The top line in Fig. 7 shows the number of triples in the PG representations input to the prototype. This is the total number of triples (including ambiguous alternatives) in the PG for each sentence prior to invoking Algorithm 1. The middle line is the number of triples remaining after merging and pruning by Algorithm 1. The bottom line is the number of triples after pruning all but the highest scoring alternatives (the baseline system). The results show that Algorithm 1 achieves significant compression over unmerged PG representations. The resulting size of the merged PG representations more closely tracks the size of the aggressively pruned

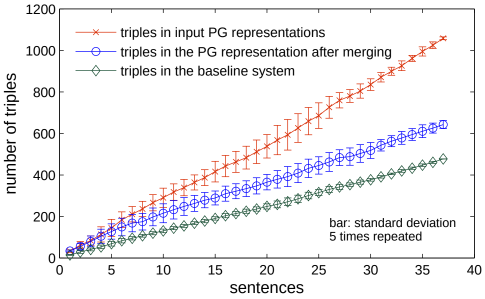


Figure 7: Total number of triples in individual sentence PG representations (top); total number of triples in the PG representation after merging in the prototype system (middle); total number of triples after pruning to the highest scoring alternative (bottom).

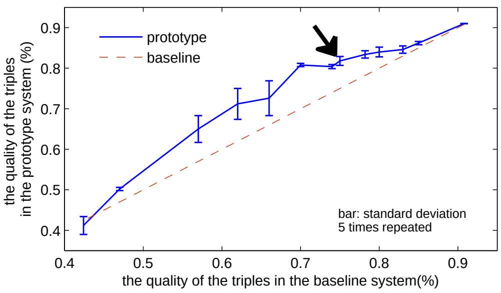


Figure 8: Sensitivity of the prototype and baseline systems to the quality of the NL system output. The quality of input triples is perturbed affecting performance accuracy of the two systems. For example, when the quality of input triples is such that the baseline system performs at 70% accuracy, the prototype system performs at 80%. The arrow indicates unperturbed language interpreter performance.

representations.

8 Experiment 3

Finally, we wanted to measure the sensitivity of our approach to the quality of the natural language interpretation. In this experiment, we artificially varied the confidence scores for the correct interpretations in the PG representations input to the prototype and baseline systems by a fixed percentage. For example, consider a node heart-1 with multiple candidate types, including the correct sense for its context: INTERNAL-ORGAN with confidence 0.8. We reran Experiment 1 varying the confidence in INTERNAL-ORGAN in increments of +/-10%, while scaling the confidences in the incorrect types equally. As the confidence in correct interpretations is increased, all correct interpretations become the highest scoring, so aggressive pruning is justified and the baseline performance approaches the prototype performance. As the confidences in correct interpretations are decreased, they are more likely to be pruned by both systems.

Fig. 8 shows that Algorithm 1 is able to recover at least some correct interpretations even when their original scores (relative to incorrect alternatives) is quite low.

9 Discussion and Future Work

Our controlled experiments suggest that it is both desirable and feasible to delay ambiguity resolution beyond sentence and text boundaries. Improvements in the correctness of semantic interpretation of sentences is possible without an explosion in size when maintaining multiple interpretations.

Nevertheless, these experiments are proofs of concept. The results confirm that it is worthwhile to subject our prototype to a more real-world, practical application. To do so, we need to address several issues.

First, we manually simulated structural (parse) ambiguities. We will complete modifications to the Stanford Parser to produce PG representations natively. This change will result in a significant increase in the number of alternatives stored in the PG representation over the current prototype. Our initial investigations suggest that there is still enough structural overlap among the candidate parse trees to allow the PG representation to control explosion, but this is an empirical question that will need to be confirmed.

We are modifying our semantic interpreter to admit induced semantic interpretation rules which will allow us to train the system in new domains.

The current prototype uses a naive heuristic for identifying co-reference candidates. We are investigating the use of off-the-shelf co-reference systems.

Finally, we are incorporating the Alchemy (Richardson and Domingos, 2006) probabilistic inference engine to calculate the probability that a candidate interpretation is correct given the PG constraints and alignments, in order to inform confirmation or pruning of interpretations.

Once these updates are complete, we will perform more wide-scale evaluations. We will investigate the automatic construction of a test corpus using text clustering to find redundant texts, and we will conduct experiments in multiple domains.

10 Related Work

Succinctly representing multiple interpretations has been explored by several researchers. The packed representation (Maxwell III and Kaplan, 1981; Crouch, 2005) uses logical formulae to denote alternative interpretations and treats the disambiguation task as the propositional satisfiability problem. Core Language Engine (Alshawi, 1992) introduces two types of packing mechanism. First, a quasi logical form allows the underspecification of several types of information, such as anaphoric references, ellipsis and semantic relations (Alshawi and Crouch, 1992). Second, a packed quasi logical form (Alshawi, 1992) compactly represents the derivations of alternative quasi logical forms. In contrast, the PG representation is (1) based on a graphical representation, (2) explicitly represents constraints and (3) includes confidence scores.

These representations and the PG representation have one feature in common: they represent a set of complete alternative interpretations of a text. Another class of compact representations, called "underspecification", has been studied as a formal representation of ambiguous sentences. These representations include Hole Semantics (Bos, 2004), Underspecified Discourse Representation Semantics (Reyle, 1995), Minimal Recursion Semantics (Copestake et al., 2005) and Dominance Constraints (Egg et al., 2001). These representations, rather than packing fully-represented candidate interpretations, specify fragments of interpretations which are unambiguously interpreted, along with constraints on their combination (corresponding to different interpretations). They generally focus on specific ambiguities such as scope ambiguity (Bos, 2004) (Egg et al., 2001) (Copestake et al., 2005) or discourse relations (Schilder, 1998) (Regneri et al., 2008).

Disambiguating compact representations has received relatively less attention. (Riezler et al., 2002; Geman and Johnson, 2002) use a packed representation to train parsers on a corpus and uses the learned statistics to disambiguate packed representations. (Clark and Harrison, 2010) uses paraphrase databases and a hand-built knowledge base to resolve underspecified representations.

Different architectures have been proposed to improve the pipeline architecture. (Sutton and McCallum, 2005; Wellner et al., 2004) maintain a beam of n best interpretations in the pipeline architecture. Their pipeline, however, consists of only two components. (Finkel et al., 2006) uses sampling over the distribution of alternative interpretations at each stage of the pipeline and then passes the sampled data to the next component. The packed representation (Crouch, 2005) and CLE (Alshawi, 1992) use packed representation in the pipeline, though both, at some stages, unpack them and re-pack the processed result. (Crouch and King, 2006) later proposes a new method that does not require unpacking and then repacking.

11 Conclusion

We have begun to address the challenge of efficiently managing multiple alternative interpretations of text. We have presented (1) a *packed graphical representation* that succinctly represents multiple alternative interpretations as well as the constraints among them, and (2) an algorithm for combining multiple PG representations to reinforce correct interpretations and discount implausible interpretations. Controlled experiments show that it is possible to improve the correctness of semantic interpretations of text by delaying disambiguation, without incurring the cost of an exponentially expanding representation.

12 Acknowledgement

Support for this research was provided in part by Air Force Contract FA8750-09-C-0172 under the DARPA Machine Reading Program

References

Alshawi, Hiyan and Richard S. Crouch. 1992. Monotonic semantic interpretation. In *ACL*, pages 32–39.

Alshawi, Hiyan, editor. 1992. *The Core Language Engine*. MIT Press, Cambridge, Massachusetts.

Barker, Ken, Bruce Porter, and Peter Clark. 2001. A library of generic concepts for composing knowledge bases. In *Proceedings of the international conference on Knowledge capture*, pages 14–21.

Bos, Johan. 2004. Computational semantics in discourse: Underspecification, resolution, and inference. *Journal of Logic, Language, and Information*, 13(2):139–157.

Clark, Peter and Phil Harrison. 2010. Exploiting paraphrases and deferred sense commitment to interpret questions more reliably. In *To appear in Proceedings of CoLing 2010*.

Copestake, Ann, Dan Flickinger, Carl Pollard, and Ivan Sag. 2005. Minimal recursion semantics: an introduction. *Research on Language and Computation*, 3:281–332.

Crouch, Richard S. and Tracy Holloway King. 2006. Semantics via f-struecture rewriting. In *Proceedings of LFG06 Conference*.

Crouch, Dick. 2005. Packed rewriting for mapping semantics to kr. In *In Proceedings Sixth International Workshop on Computational Semantics*.

Egg, Markus, Alexander Koller, and Joachim Niehren. 2001. The constraint language for lambda structures. *Journal of Logic, Language, and Information Vol 10 (4), 2001, pp.457-485*, 10:457–485.

Finkel, Jenny Rose, Christopher D. Manning, and Andrew Y. Ng. 2006. Solving the problem of cascading errors: approximate bayesian inference for linguistic annotation pipelines. In *EMNLP*, pages 618–626, Morristown, NJ, USA.

Geman, Stuart and Mark Johnson. 2002. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *ACL*, pages 279–286.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*, pages 423–430, Morristown, NJ, USA.

Lesk, Michael. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *SIGDOC '86: Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26, New York, NY, USA.

Maxwell III, John T. and Ronald M. Kaplan. 1981. A method for disjunctive constraint satisfaction. In Tomita, Masaru, editor, *Current Issues in Parsing Technology*, pages 173–190. Kluwer Academic Publishers, Dordrecht.

Patwardhan, Siddharth, Satanjeev Banerjee, and Ted Pedersen. 2005. Senserelate:: Targetword-A generalized framework for word sense disambiguation. In *ACL*.

Pearl, Judea. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufmann.

Punyakanok, Vasin, Dan Roth, and Wen tau Yih. 2005. The necessity of syntactic parsing for semantic role labeling. In Kaelbling, Leslie Pack and Alessandro Saffiotti, editors, *IJCAI*, pages 1117–1123. Professional Book Center.

Regneri, Michaela, Markus Egg, and Alexander Koller. 2008. Efficient processing of underspecified discourse representations. In *HLT*, pages 245–248, Morristown, NJ, USA.

Reyle, Uwe. 1995. Underspecified discourse representation structures and their logic. *Logic Journal of the IGPL*, 3(2-3):473–488.

Richardson, Matthew and Pedro Domingos. 2006. *Markov logic networks*. Kluwer Academic Publishers.

Riezler, Stefan, Tracy H. King, Ronald M. Kaplan, Richard S. Crouch, John T. Maxwell III, and Mark Johnson. 2002. Parsing the wall street journal using a lexical-functional grammar and discriminative estimation techniques. In *ACL*, pages 271–278.

Schilder, Frank. 1998. An underspecified segmented discourse representation theory (USDRT). In *COLING-ACL*, pages 1188–1192.

Sutton, Charles and Andrew McCallum. 2005. Joint parsing and semantic role labeling. In *CONLL*, pages 225–228, Morristown, NJ, USA.

Wellner, Ben, Andrew McCallum, Fuchun Peng, and Michael Hay. 2004. An integrated, conditional model of information extraction and coreference with application to citation matching. In *UAI*, pages 593–601, Arlington, Virginia, United States.

Automatic generation of inter-passage links based on semantic similarity

Petr Knoth, Jakub Novotny, Zdenek Zdrahal
Knowledge Media Institute
The Open University
p.knoth@open.ac.uk

Abstract

This paper investigates the use and the prediction potential of semantic similarity measures for automatic generation of links across different documents and passages. First, the correlation between the way people link content and the results produced by standard semantic similarity measures is investigated. The relation between semantic similarity and the length of the documents is then also analysed. Based on these findings a new method for link generation is formulated and tested.

1 Introduction

Text retrieval methods are typically designed to find documents relevant to a query based on some criterion, such as BM25 or cosine similarity (Manning et al., 2008). Similar criteria have also been used to identify documents relevant to the given reference document, thus in principle linking the reference document to the related documents (Wilkinson and Smeaton, 1999). This paper studies the correspondence between the results of this approach and the way linking is performed by people. The study confirms that the length of documents is an important factor usually causing the quality of current link generation approaches to deteriorate. As a result, methods working at a finer granularity than documents should be investigated. This will also improve the speed of access to information. For example, when users read through a long document, they should be able to quickly access a passage in another possibly long document related to the discussed topic. The automatic detection of document pairs containing highly related passages is the task addressed in this paper.

A number of approaches for automatic link generation have used measures of semantic similarity. While these measures were widely used for the discovery of related documents in practise, their correspondence to the way people link content has not been sufficiently investigated (see Section 2). As our contribution to this topic, we present in this paper an approach which tries to first investigate this correspondence on a large text corpus. The resulting method is then motivated by the outcomes of this analysis.

It has been recognised in information retrieval that when a collection contains long documents, better performance is often achieved by breaking each document into subparts or passages and comparing these rather than the whole documents to a query (Manning et al., 2008). A suitable granularity of the breakdown is dependent on a number of circumstances, such as the type of the document collection or the information need. In this work, we have decided to work at the level of documents and paragraphs. Our task can be formalized as a two-step process:

1. Given a collection of documents, our goal is to identify candidate pairs of documents between which a link may be induced.
2. Given each candidate pair of documents, our task is to identify pairs of passages, such that the topics in the passages are related in both documents.

The method presented in this paper has many

This work has been partially supported by Eurogene - Contract no. ECP-2006-EDU-410018)

potential applications. First, it may be used for the interlinking of resources that were not originally created as hypertext documents and for the maintenance or the discovery of new links as the collection grows. Second, the method can be applied to improve navigation in collections with long texts, such as books or newspaper articles. A link may be identified by the system automatically and the user can be pointed immediately to the part of the text which is relevant to the block of text currently being read. Similar application has been developed by (Kolak and Schilit, 2008) who provided a method for mining repeated word sequences (quotations) from very large text collections and integrated it with the Google Books archive. Other application areas may involve text summarization and information retrieval.

The paper makes the following contributions:

- It provides a new interpretation and insight in the use of semantic similarity measures for the automatic generation of links.
- It develops a novel two-step approach for the discovery of passage-passage links across potentially long documents and it identifies and discusses the selection of the parameters.

The rest of the paper is organized as follows. Section 2 presents the related work in the field. Section 3 discusses the data selected for our experiment and Section 4 describes how the data were processed in order to perform our investigation. In Section 5, the analysis in which we compared the results produced by semantic similarity measures with respect to the way people link content is presented. Section 6 then draws on this analysis and introduces the method for automatic generation of links which is finally evaluated in Section 7.

2 Related Work

In the 1990s, the main application area for link generation methods were hypertext construction systems. A survey of these methods is provided by (Wilkinson and Smeaton, 1999). In the last decade, methods for finding related documents became the de-facto standard in large digital repositories, such as PubMed or the ACM Digital Library. Search engines including Google also generate links to related pages or research articles.

Generating links pointing to units of a smaller granularity than a document, which can be considered as a task of *passage* or *focused* retrieval, has also been addressed recently. In this task, the system locates the relevant information inside the document instead of only providing a link to the document. The Initiative for the Evaluation of XML retrieval (INEX) started to play an essential role in link generation by providing tracks for the evaluation of link generation systems (Huang et al., 2008; Huang et al., 2009) using the Wikipedia collection at both the document and the passage level.

Current approaches can be divided into three groups: (1) *link-based* approaches discover new links by exploiting an existing link graph (Itakura and Clarke, 2008; Jenkinson et al., 2008; Lu et al., 2008). (2) *semi-structured* approaches try to discover new links using semi-structured information, such as the anchor texts or document titles (Geva, 2007; Dopichaj et al., 2008; Granitzer et al., 2008). (3) *purely content-based* approaches use as an input plain text only. They typically discover related resources by calculating semantic similarity based on document vectors (Allan, 1997; Green, 1998; Zeng and Bloniarz, 2004; Zhang and Kamps, 2008; He, 2008). Some of the mentioned approaches, such as (Lu et al., 2008), combine multiple approaches.

Although link generation methods are widely used in practise, more work is needed to understand which features contribute to the quality of the generated links. Work in this area includes the study of (Green, 1999) who investigated how lexical chaining based on ontologies can contribute to the quality of the generated links, or the experiments of (Zeng and Bloniarz, 2004) who compared the impact of the manually and automatically extracted keywords. There has also been effort in developing methods that can in addition to link generation assign a certain semantic type to the extracted links and thus describe the relationship between documents (Allan, 1997).

The method presented in this paper is purely content-based and therefore is applicable in any text collection. Its use in combination with link-based or semi-structured approaches is also possible. The rationale for the method comes from the analysis of the prediction potential of semantic similarity for automatic link generation presented in Section 5. Related analysis is presented in (He, 2008) which claims that linked articles are more likely to be semantically similar[1], however, the study does not provide sufficient evidence to confirm and describe this relationship. In link generation, we are more interested in asking the opposite question, i.e. whether articles with higher semantic similarity are more likely to be linked. Our study provides a new insight into this relationship and indicates that the relationship is in fact more complex than originally foreseen by He.

3 Data selection

This section introduces the document collection used for the analysis and the experiments. The following properties were required for the document collection to be selected for the experiments. First, in order to be able to measure the correlation between the way people link content and the results produced by semantic similarity measures, it was necessary to select a document collection which can be considered as relatively well interlinked. Second, it was important for us to work with a collection containing a diverse set of topics. Third, we required the collection to contain articles of varied length. We were mostly interested in long documents, which create conditions for the testing of passage retrieval methods. We decided to use the Wikipedia collection, because it satisfies all our requirements and has also been used in the INEX Link-The-Wiki-Track.

Wikipedia consists of more than four million pages spread across five hundred thousands categories. As it would be for our calculation unnecessarily expensive to work with the whole encyclopedia, a smaller, but still a sufficiently large subset of Wikipedia, which satisfies our requirements of topic diversity and document length, was selected. Our document collection was generated from articles in categories containing the words United Kingdom. This includes categories, such as United Kingdom, Geography of United Kingdom or History of the United Kingdom. There are about 3,000 such categories and 57,000 distinct articles associated to them. As longer articles provide better test conditions for passage retrieval methods, we selected the 5,000 longest articles out of these 57,000. This corresponds to a set where each article has the length of at least 1,280 words.

4 Data preprocessing

Before discussing the analysis performed on the document collection, let us briefly describe how the documents were processed and the semantic similarity calculated.

First, the N articles/documents $D = \{d_1, d_2, \ldots, d_N\}$ in our collection were preprocessed to extract plain text by removing the Wiki markup. The documents were then tokenized and a dictionary of terms $T = \{t_1, t_2, \ldots, t_M\}$ was created. Assuming that the order of words can be neglected (the bag-of-words assumption) the document collection can be represented using a $N \times M$ term-document matrix. In this way, each document is modelled as a vector corresponding to a particular row of the matrix. As it is inefficient to represent such a sparse vector in memory (most of the values are zeros), only the non-zero values were stored. *Term frequency - inverse document frequency (tfidf)* weighting was used to calculate the values of the matrix. Term frequency tf_{t_i,d_j} is a normalized frequency of term t_i in document d_j:

$$tf_{t_i,d_j} = \frac{f(t_i, d_j)}{\sum_k f(t_k, d_j)}$$

Inverse document frequency idf_{t_i} measures the general importance of term t_i in the collection of documents D by counting the number of documents which contain term t_i:

$$idf_{t_i} = \log \frac{|D|}{|d_j : t_i \in d_j|}$$

$$tfidf_{t_i,d_j} = tf_{t_i,d_j} . idf_{t_i}$$

Similarity is then defined as the function $sim(\overrightarrow{x}, \overrightarrow{y})$ of the document vectors $\overrightarrow{x}$ and $\overrightarrow{y}$. There exists a number of similarity measures used for the calculation of similarity between two vectors (Manning and Schuetze, 1999), such as *cosine*, *overlap*, *dice* or *Jaccard* measures. Some studies employ algorithms for the reduction of dimensions of the vectors prior to the calculation

[1] With respect to the cosine similarity measure.

of similarity to improve the results. These approaches may involve techniques, such as lexical chaining (Green, 1999), Latent Semantic Indexing (Deerwester et al., 1990), random indexing (Widdows and Ferraro, 2008) and Latent Dirichlet Allocation (Blei et al., 2003). In this work we intentionally adopted perhaps the most standard similarity measure - cosine similarity calculated on the *tfidf* vectors and no dimensionality reduction technique was used. The formula is provided for completeness:

$$sim_{cosine}(\overrightarrow{x}, \overrightarrow{y}) = \frac{\overrightarrow{x}.\overrightarrow{y}}{|x|.|y|}$$

Cosine similarity with *tfidf* vectors has been previously used in automatic link generation systems producing state-of-the-art results when compared to other similarity measures (Chen et al., 2004). This allows us to report on the effectiveness of the most widely used measure with respect to the way the task is completed by people. While more advanced techniques might be in some cases better predictors for link generation, we did not experiment with them as we preferred to focus on the investigation of the correlation between the most widely used measure and manually created links. Such study has to our knowledge never been done before, but it is necessary for the justification of automatic link generation methods.

5 Semantic similarity as a predictor for link generation

The document collection described in Section 3 has been analysed as follows. First, pair-wise similarities using the formulas described in Section 4 were calculated. Cosine similarity is a symmetric function and, therefore, the calculation of all inter-document similarities in the dataset of $5,000$ documents requires the evaluation of $\frac{5,000^2}{2} - 5,000 = 12,495,000$ combinations. Figure 1 shows the distribution of the document pairs (on a log_{10} scale) with respect to their similarity value. The frequency follows a power law distribution. In our case, 99% of the pairs have similarity lower than 0.1.

To compare the semantic similarity measures with the links created by Wikipedia authors, all inter-document intra-collection links, i.e. links created by users of Wikipedia commencing from and pointing to a document within our collection, were extracted. These links represent the connections as seen by the users regardless of their direction. Each of these links can be associated with a similarity value calculated in the previous step. Documents with similarity lower than 0.1 were ignored. Out of the $120,602$ document pairs with inter-document similarity higher than 0.1, $17,657$ pairs were also connected by a user-created link.

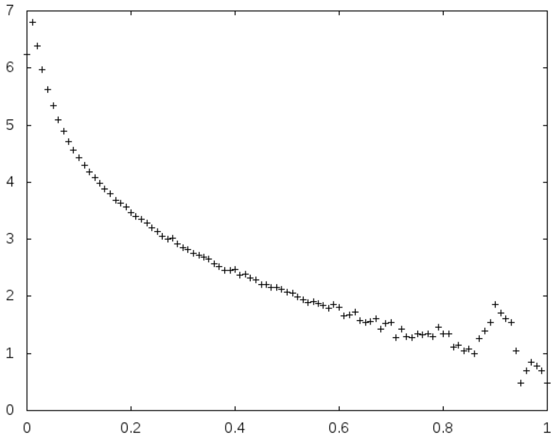


Figure 1: The histogram shows the number of document pairs on a $\log_{10}$ scale (y-axis) with respect to their cosine similarity (x-axis).

For the evaluation, interval with cosine similarity $[0.1, 1]$ was divided evenly into 100 buckets and all 120,602 document pairs were assigned to the buckets according their similarity values. From the distribution shown in Figure 1, buckets corresponding to higher similarity values contain fewer document pairs than buckets corresponding to smaller similarity values. Therefore, for each bucket, the number of user created links within the bucket was normalized by the number of document pairs in the bucket. This number is the likelihood of the document pair being linked and will be called *linked-pair likelihood.* The relation between semantic similarity and linked-pair likelihood is shown in Figure 2.

As reported in Section 2, semantic similarity has been previously used as a predictor for the automatic generation of links. The typical scenario was that the similarity between pairs of documents was calculated and the links between the

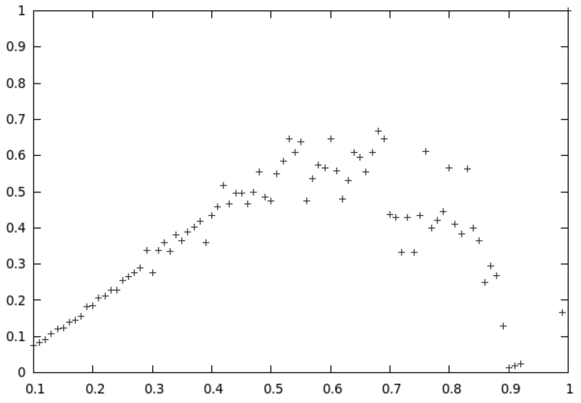


Figure 2: The linked-pair likelihood (y-axis) with respect to the cosine similarity (x-axis).

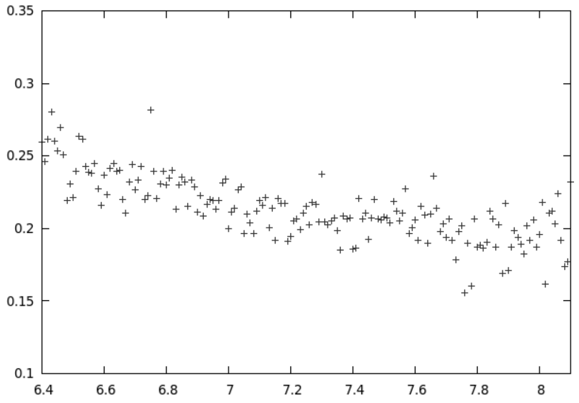


Figure 3: The average cosine similarity (y-axis) of document pairs of various length (x-axis) between which there exists a link. The x-axis is calculated as a $\log_{10}(l_1.l_2)$

most similar documents were generated (Wilkinson and Smeaton, 1999). If this approach was correct, we would expect the curve shown in Figure 2 to be monotonically increasing. However, the relation shown in Figure 2 is in accordance with our expectations only up to the point 0.55. For higher values of inter-document similarity the linked-pair likelihood does not rise or it even decreases.

Spearman's rank correlation and Pearson correlation were applied to estimate the correlation coefficients and to test the statistical significance of our observation. This was performed in two intervals: $[0, 0.55]$ and $[0.55, 1]$. A very strong positive correlation 0.986 and 0.987 have been received in the first interval for the Spearman's and Pearson coefficients respectively. A negative correlation -0.640 and -0.509 have been acquired for the second interval again for the Spearman's and Pearson coefficients respectively. All the measured correlations are significant for p-value well beyond $p < 0.001$. Very similar results have been achieved using different collections of documents.

The results indicate that high similarity value is not necessarily a good predictor for automatic link generation. A possible explanation for this phenomenon is that people create links between related documents that provide new information and therefore do not link nearly identical content. However, as content can be in general linked for various purposes, more research is needed to investigate if document pairs at different similarity levels also exhibit different qualitative properties. More specifically, can the value of semantic similarity be used as a predictor for relationship typing?

An important property of semantic similarity as a measure for automatic generation of links is the robustness with respect to the length of documents. As mentioned in Section 4, cosine similarity is by definition normalized by the product of the documents length. Ideally the cosine similarity should be independent of the documents length. To verify this in our dataset, we have taken pairs of documents between which Wikipedia users assigned links and divided them into buckets with respect to the function $\log_{10}(l_1.l_2)$, where l_1 and l_2 are the lengths of the two documents in the document pair and the logarithm is used for scaling. The value of each bucket was calculated as an average similarity of the bucket members. The results are shown in Figure 3. The graph shows that the average similarity value is slightly decreasing with respect to the length of the articles. Values -0.484 and -0.231 were obtained for Spearman's and Pearson correlation coefficients respectively. Both correlations are statistically significant for $p < 0.001$. A much stronger correlation was measured for Spearman's than for Pearson which can be explained by the fact that Spearman's correlation is calculated based on ranks rather than real values and is thus less sensitive to outliers.

Our experience from repeating the same experiment on another Wikipedia subset generated from categories containing the word Geography tells us that the decrease is even more noticeable when short and long articles are combined. The decrease in average similarity suggests that if cosine similarity is used for the automatic generation of links then document pairs with higher value of $l_1.l_2$ have a higher linked-pair likelihood than pairs with a smaller value of this quantity. In other words, links created between documents with small $l_1.l_2$ typically exhibit a larger value of semantic similarity than links created between documents with high value of $l_1.l_2$. Although the decrease may seem relatively small, we believe that this knowledge may be used for improving automatic link generation methods by adaptively modifying the thresholds with respect to the $l_1.l_2$ length.

6 Link generation method

In this section we introduce the method for the automatic generation of links. The method can be divided into two parts (1) Identification of candidate link pairs (i.e. the generation of document-to-document links) (2) Recognition of passages sharing a topic between the two documents (i.e. the generation of passage-to-passage links).

6.1 Document-to-document links

The algorithm for link generation at the granularity of a document is motivated by the findings reported in Section 5.

```
Algorithm 1: Generate document links
Input: A set of document vectors $D$,
min. sim. $\alpha$, max. sim. $\beta \in [0,1], C = \emptyset$
Output: A set $C$ of candidate links
of form $\langle d_i, d_j, sim \rangle \in C$ where $d_i$ and $d_j$ are
documents and $sim \in [0,1]$ is their similarity
for each $\{\langle d_i, d_j \rangle | i, j \in \aleph_0 \wedge i < j < |D|\}$ do
    $sim_{d_i,d_j} := similarity(d_i, d_j)$
    if $sim_{d_i,d_j} > \alpha \wedge sim_{d_i,d_j} < \beta$ then
        $C := C \cup \langle d_i, d_j, sim_{d_i,d_j} \rangle$
```

The algorithm takes as the input a set of document vectors and two constants - the minimum and maximum similarity thresholds - and iterates over all pairs of document vectors. It outputs all document vector pairs, such that their similarity is higher than α and smaller than β. For well chosen β, the algorithm does not generate links between nearly duplicate pairs. If we liked to rank the discovered links according to the confidence of the system, we would suggest to assign each pair a value using the following function.

$$rank_{d_i,d_j} = |sim_{d_i,d_j} - (\alpha + \frac{\beta - \alpha}{2})|$$

The ranking function makes use of the fact that the system is most confident in the middle of the similarity region defined by constants α and β, under the assumption that suitable values for these constants are used. The higher the rank of a document pair, the better the system's confidence.

6.2 Passage-to-passage links

Due to a high number of combinations, it is typically infeasible even for relatively small collections to generate passage-to-passage links across documents directly. However, the complexity of this task is substantially reduced when passage-to-passage links are discovered in a two-step process.

```
Algorithm 2: Generate passage links
Input: Sets $P_i, P_j$ of paragraph document
vectors for each pair in $C$
min. sim. $\gamma$, max. sim. $\delta \in [0,1]$ such that
$\alpha < \gamma \wedge \beta < \delta, , L = \emptyset$
Output: A set $L$ of passage links
of form $\langle p_{k_i}, p_{l_j}, sim \rangle \in L$ where $p_{k_i}$ and
$p_{l_j}$ are paragraphs in documents $d_i, d_j$
and $sim \in [0,1]$ is their similarity
for each $\{\langle p_{k_i}, p_{l_j} \rangle | p_{k_i} \in P_i, p_{l_j} \in P_j\}$ do
    $sim_{p_{k_i},p_{l_j}} := similarity(p_{k_i}, p_{l_j})$
    if $sim_{p_{k_i},p_{l_j}} > \gamma \wedge sim_{p_{k_i},p_{l_j}} < \delta$ then
        $L := L \cup \langle p_{k_i}, p_{l_j}, sim_{p_{k_i},p_{l_j}} \rangle$
```

As Section 5 suggests, the results of Algorithm 1 may be improved by adaptive changing of the thresholds α and β based on the length of the document vectors. More precisely, in the case of cosine similarity, this is the quantity $lr = l_1.l_2$. The value α should be higher (β lower) for pairs with low lr than for pairs with high lr and vice versa. Although the relative quantification of this ratio is left for future work, we believe that we can exploit these findings for the generation of passage-to-passage links.

More specifically, we know that the length of passages (paragraphs in our case) is lower than the length of the whole documents. Hence, the similarity of a linked passage-to-passage pair should be on average higher than the similarity of a linked document-to-document pair, as revealed by the results of our analysis. This knowledge is used within Algorithm 2 to set the parameters γ and δ. The algorithm shows, how passage-to-passage links are calculated for a single document pair previously identified by Algorithm 1. Applying the two-step process allows the discovery of document pairs, which are likely to contain strongly linked passages, at lower similarity levels and to recognize the related passages at higher similarity levels while still avoiding duplicate content.

7 Results

The experimental evaluation of the methods presented in Section 6 is divided into two parts: (1) the evaluation of document-to-document links (Algorithm 1) and (2) the evaluation of passage-to-passage links (Algorithm 2).

7.1 Evaluation of document-to-document links

As identified in Section 5 (and shown in Figure 2), the highest linked-pair likelihood does not occur at high similarity values, but rather somewhere between similarity 0.5 and 0.7. According to Figure 2, the linked-pair likelihood in this similarity region ranges from 60% to 70%. This value is in our view relatively high and we think that it can be explained by the fact that Wikipedia articles are under constant scrutiny by users who eventually discover most of the useful connections. However, how many document pairs that could be linked in this similarity region have been missed by the users? That is, how much can our system help in the discovery of possible connections?

Suppose that our task would be to find document pairs about linking of which the system is most certain. In that case we would set the thresholds α and β somewhere around these values depending on how many links we would like to obtain. In our evaluation, we have extracted pairs of documents from the region between $\alpha = 0.65$ and $\beta = 0.70$ regardless of whether there originally was a link assigned by Wikipedia users. An evaluation tool which allowed a subject to display the pair of Wiki documents next to each other and to decide whether there should or should not be a link between the documents was then developed. We did not inform the subject about the existence or non-existence of links between the pages. More specifically, the subject was asked to decide yes (link generated correctly) if and only if they found it beneficial for a reader of the first or the second article to link them together regardless of the link direction. The subject was asked to decide no (link generated incorrectly) if and only if they felt that navigating the user from or to the other document does not provide additional value. For example, in cases where the relatedness of the documents is based on their lexical rather than their semantic similarity.

The study revealed that 91% of the generated links were judged by the subject as correct and 9% as incorrect. Table 1 shows the results of the experiment with respect to the links originally assigned by the users of Wikipedia. It is interesting to notice that in 3% of the cases the subject decided not to link the articles even though they were in fact linked on Wikipedia. Overall, the algorithm discovered in 30% of the cases a useful connection which was missing in Wikipedia. This is in line with the findings of (Huang et al., 2008) who claims that the validity of existing links in Wikipedia is sometimes questionable and useful links may be missing.

An interesting situation in the evaluation occurred when the subject discovered a pair of articles with titles *Battle of Jutland* and *Night Action at the Battle of Jutland*. The Wikipedia page indicated that it is an orphan and asked users of Wikipedia to link it to other Wikipedia articles. Our method would suggest the first article as a good choice.

		Wikipedia link	
		yes	no
Subject's decision	yes	0.61	0.30
	no	0.03	0.06

Table 1: Document-to-document links from the $[0.65, 0.7]$ similarity region. The subject's decision in comparison to the Wikipedia links.

		Wikipedia link	
		yes	no
Subject's decision at page level	yes	0.16	0.10
	no	0.18	0.56

Table 2: Document-to-document candidate links generation from the $[0.2, 0.21]$ similarity region and document pairs with high lr ($lr \in [7.8 - 8]$).

7.2 Evaluation of passage-to-passage linking

The previous section provided evidence that the document-to-document linking algorithm is capable of achieving high performance when parameters α, β are well selected. However, Section 5 indicated that it is more difficult to discover links across long document pairs. Thereby, we have evaluated the passage-to-passage linking on document pairs with quite low value of similarity $[0.2, 0.21]$. According to Figure 2, this region has only 15% linked-pair likelihood.

Clearly, our goal was not to evaluate the approach in the best possible environment, but rather to check whether the method is able to discover valuable passage-to-passage links from very long articles with low similarity. Articles with this value of similarity would be typically ranked very poorly by link generation methods working at the document level.

Table 2 shows the results after the first step of the approach, described in Section 6, with respect to the links assigned by Wikipedia users. As in the previous experiment, the subject was given pairs of documents and decided whether they should or should not be linked. Parameters α and β were set to 0.2, 0.21 respectively. Table 2 indicates that that the accuracy $(16\% + 10\% = 26\%)$ is at this similarity region much lower than the one reported in Table 1, which is exactly in line with our expectations. It should be noticed that 34% of the document pairs were linked by Wikipedia users, even though only 15% would be predicted by linked-pair likelihood shown in Figure 2. This confirms that long document pairs exhibit a higher probability of being linked in the same similarity region than shorter document pairs.

		System's decision	
		yes	no
Subject's decision	yes (correct)	0.14	0.46
	no (incorrect)	0.24	0.16

Table 3: Passage-to-passage links generation for very long documents. Passages extracted from the $[0.4, 0.8]$ similarity region.

If our approach for passage-to-passage link generation (Algorithm 2) is correct, we should be able to process the documents paragraphs and detect possible passage-to-passage links. The selection of the parameters γ and δ influences the willingness of the system to generate links. For this experiment, we set the parameters γ, δ to 0.4, 0.8 respectively. The subject was asked to decide: (1) if the connection discovered by the link generation method at the granularity of passages was useful (when the system generated a link) (2) whether the decision not to generate link is correct (when the system did not generate a link). The results of this evaluation are reported in Table 3. It can be seen that the system made in 60% $(14\% + 46\%)$ of the cases the correct decision. Most mistakes were made by generating links that were not sufficiently related (24%). This might be improved by using a higher value of γ (lower value of δ).

8 Conclusions

This paper provided a new insight into the use of semantic similarity as a predictor for automatic link generation by performing an investigation in the way people link content. This motivated us in the development of a novel purely content-based approach for automatic generation of links at the granularity of both documents and paragraphs which does not expect semantic similarity and linked-pair likelihood to be directly proportional.

References

Allan, James. 1997. Building hypertext using information retrieval. *Inf. Process. Manage.*, 33:145–159, March.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *JOURNAL OF MACHINE LEARNING RESEARCH*, 3:993–1022.

Chen, Francine, Ayman Farahat, and Thorsten Brants. 2004. Multiple similarity measures and source-pair information in story link detection. In *In HLT-NAACL 2004*, pages 2–7.

Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

Dopichaj, Philipp, Andre Skusa, and Andreas Heß. 2008. Stealing anchors to link the wiki. In Geva et al. (Geva et al., 2009), pages 343–353.

Geva, Shlomo, Jaap Kamps, and Andrew Trotman, editors. 2009. *Advances in Focused Retrieval, 7th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2008, Dagstuhl Castle, Germany, December 15-18, 2008. Revised and Selected Papers*, volume 5631 of *Lecture Notes in Computer Science*. Springer.

Geva, Shlomo. 2007. Gpx: Ad-hoc queries and automated link discovery in the wikipedia. In Fuhr, Norbert, Jaap Kamps, Mounia Lalmas, and Andrew Trotman, editors, *INEX*, volume 4862 of *Lecture Notes in Computer Science*, pages 404–416. Springer.

Granitzer, Michael, Christin Seifert, and Mario Zechner. 2008. Context based wikipedia linking. In Geva et al. (Geva et al., 2009), pages 354–365.

Green, Stephen J. 1998. Automated link generation: can we do better than term repetition? *Comput. Netw. ISDN Syst.*, 30(1-7):75–84.

Green, Stephen J. 1999. Building hypertext links by computing semantic similarity. *IEEE Trans. on Knowl. and Data Eng.*, 11(5):713–730.

He, Jiyin. 2008. Link detection with wikipedia. In Geva et al. (Geva et al., 2009), pages 366–373.

Huang, Wei Che, Andrew Trotman, and Shlomo Geva. 2008. Experiments and evaluation of link discovery in the wikipedia.

Huang, Wei Che, Shlomo Geva, and Andrew Trotman. 2009. Overview of the inex 2009 link the wiki track.

Itakura, Kelly Y. and Charles L. A. Clarke. 2008. University of waterloo at inex 2008: Adhoc, book, and link-the-wiki tracks. In Geva et al. (Geva et al., 2009), pages 132–139.

Jenkinson, Dylan, Kai-Cheung Leung, and Andrew Trotman. 2008. Wikisearching and wikilinking. In Geva et al. (Geva et al., 2009), pages 374–388.

Kolak, Okan and Bill N. Schilit. 2008. Generating links by mining quotations. In *HT '08: Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*, pages 117–126, New York, NY, USA. ACM.

Lu, Wei, Dan Liu, and Zhenzhen Fu. 2008. Csir at inex 2008 link-the-wiki track. In Geva et al. (Geva et al., 2009), pages 389–394.

Manning, Christopher D. and Hinrich Schuetze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1 edition, June.

Manning, Ch. D., P. Raghavan, and H. Schütze. 2008. *Introduction to Information Retrieval*. Cambridge, July.

Widdows, Dominic and Kathleen Ferraro. 2008. Semantic vectors: a scalable open source package and online technology management application. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard Joseph Mariani Jan Odjik Stelios Piperidis Daniel Tapias, editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Wilkinson, Ross and Alan F. Smeaton. 1999. Automatic link generation. *ACM Computing Surveys*, 31.

Zeng, Jihong and Peter A. Bloniarz. 2004. From keywords to links: an automatic approach. *Information Technology: Coding and Computing, International Conference on*, 1:283.

Zhang, Junte and Jaap Kamps. 2008. A content-based link detection approach using the vector space model. In Geva et al. (Geva et al., 2009), pages 395–400.

Dependency-driven Anaphoricity Determination for Coreference Resolution

Fang Kong Guodong Zhou Longhua Qian Qiaoming Zhu[*]
JiangSu Provincial Key Lab for Computer Information Processing Technology
School of Computer Science and Technology Soochow University
{kongfang, gdzhou, qianlonghua, qmzhu}@suda.edu.cn

Abstract

This paper proposes a dependency-driven scheme to dynamically determine the syntactic parse tree structure for tree kernel-based anaphoricity determination in coreference resolution. Given a full syntactic parse tree, it keeps the nodes and the paths related with current mention based on constituent dependencies from both syntactic and semantic perspectives, while removing the noisy information, eventually leading to a dependency-driven dynamic syntactic parse tree (D-DSPT). Evaluation on the ACE 2003 corpus shows that the D-DSPT outperforms all previous parse tree structures on anaphoricity determination, and that applying our anaphoricity determination module in coreference resolution achieves the so far best performance.

1 Introduction

Coreference resolution aims to identify which noun phrases (NPs, or mentions) refer to the same real-world entity in a text. According to Webber (1979), coreference resolution can be decomposed into two complementary sub-tasks: (1) anaphoricity determination, determining whether a given NP is anaphoric or not; and (2) anaphor resolution, linking together multiple mentions of a given entity in the world. Although machine learning approaches have performed reasonably well in coreference resolution without explicit anaphoricity determination (e.g. Soon et al. 2001; Ng and Cardie 2002b; Yang et al. 2003, 2008; Kong et al. 2009), knowledge of NP anaphoricity is expected to much improve the performance of a coreference resolution system, since a non-anaphoric NP does not have an antecedent and therefore does not need to be resolved.

Recently, anaphoricity determination has been drawing more and more attention. One common approach involves the design of some heuristic rules to identify specific types of non-anaphoric NPs, such as pleonastic *it* (e.g. Paice and Husk 1987; Lappin and Leass 1994, Kennedy and Boguraev 1996; Denber 1998) and definite descriptions (e.g. Vieira and Poesio 2000). Alternatively, some studies focus on using statistics to tackle this problem (e.g., Bean and Riloff 1999; Bergsma et al. 2008) and others apply machine learning approaches (e.g. Evans 2001;Ng and Cardie 2002a, 2004,2009; Yang et al. 2005; Denis and Balbridge 2007; Luo 2007; Finkel and Manning 2008; Zhou and Kong 2009).

As a representative, Zhou and Kong (2009) directly employ a tree kernel-based method to automatically mine the non-anaphoric information embedded in the syntactic parse tree. One main advantage of the kernel-based methods is that they are very effective at reducing the burden of feature engineering for structured objects. Indeed, the kernel-based methods have been successfully applied to mine structured information in various NLP applications like syntactic parsing (Collins and Duffy, 2001; Moschitti, 2004), semantic relation extraction (Zelenko et al., 2003; Zhao and Grishman, 2005; Zhou et al. 2007; Qian et al., 2008), semantic role labeling (Moschitti, 2004); coreference resolution (Yang et al., 2006; Zhou et al., 2008). One of the key problems for the kernel-based methods is how to effectively capture the structured information according to the nature of the structured object in the specific task.

This paper advances the state-of-the-art performance in anaphoricity determination by ef-

[*] Corresponding author

fectively capturing the structured syntactic information via a tree kernel-based method. In particular, a dependency-driven scheme is proposed to dynamically determine the syntactic parse tree structure for tree kernel-based anaphoricity determination by exploiting constituent dependencies from both the syntactic and semantic perspectives to keep the necessary information in the parse tree as well as remove the noisy information. Our motivation is to employ critical dependency information in constructing a concise and effective syntactic parse tree structure, specifically targeted for tree kernel-based anaphoricity determination.

The rest of this paper is organized as follows. Section 2 briefly describes the related work on both anaphoricity determination and exploring syntactic parse tree structures in related tasks. Section 3 presents our dependency-driven scheme to determine the syntactic parse tree structure. Section 4 reports the experimental results. Finally, we conclude our work in Section 5.

2 Related Work

This section briefly overviews the related work on both anaphoricity determination and exploring syntactic parse tree structures.

2.1 Anaphoricity Determination

Previous work on anaphoricity determination can be broadly divided into three categories: heuristic rule-based (e.g. Paice and Husk 1987;Lappin and Leass 1994; Kennedy and Boguraev 1996; Denber 1998; Vieira and Poesio 2000; Cherry and Bergsma 2005), statistics-based (e.g. Bean and Riloff 1999; Cherry and Bergsma 2005; Bergsma et al. 2008) and learning-based methods (e.g. Evans 2001; Ng and Cardie 2002a; Ng 2004; Yang et al. 2005; Denis and Balbridge 2007; Luo 2007; Finkel and Manning 2008; Zhou and Kong 2009; Ng 2009).

The heuristic rule-based methods focus on designing some heuristic rules to identify specific types of non-anaphoric NPs. Representative work includes: Paice and Husk (1987), Lappin and Leass (1994) and Kennedy and Boguraev (1996). For example, Kennedy and Boguraev (1996) looked for modal adjectives (e.g. “necessary”) or cognitive verbs (e.g. “It is thought that…” in a set of patterned constructions) in identifying pleonastic *it*.

Among the statistics-based methods, Bean and Riloff (1999) automatically identified existential definite NPs which are non-anaphoric. The intuition behind is that many definite NPs are not anaphoric since their meanings can be understood from general world knowledge, e.g. “the FBI”. They found that existential NPs account for 63% of all definite NPs and 76% of them could be identified by syntactic or lexical means. Cherry and Bergsma (2005) extended the work of Lappin and Leass (1994) for large-scale anaphoricity determination by additionally detecting pleonastic *it*. Bergsma et al. (2008) proposed a distributional method in detecting non-anaphoric pronouns. They first extracted the surrounding context of the pronoun and gathered the distribution of words that occurred within the context from a large corpus, and then identified the pronoun either anaphoric or non-anaphoric based on the word distribution.

Among the learning-based methods, Evans (2001) automatically identified the non-anaphoricity of pronoun *it* using various kinds of lexical and syntactic features. Ng and Cardie (2002a) employed various domain-independent features in identifying anaphoric NPs. They trained an anaphoricity classifier to determine whether a NP was anaphoric or not, and employed an independently-trained coreference resolution system to only resolve those mentions which were classified as anaphoric. Experiments showed that their method improved the performance of coreference resolution by 2.0 and 2.6 to 65.8 and 64.2 in F1-measure on the MUC-6 and MUC-7 corpora, respectively. Ng (2004) examined the representation and optimization issues in computing and using anaphoricity information to improve learning-based coreference resolution. On the basis, he presented a corpus-based approach (Ng, 2009) for achieving global optimization by representing anaphoricity as a feature in coreference resolution. Experiments on the ACE 2003 corpus showed that their method improved the overall performance by 2.8, 2.2 and 4.5 to 54.5, 64.0 and 60.8 in F1-measure on the NWIRE, NPAPER and BNEWS domains, respectively. However, he did not look into the contribution of anaphoricity determination on coreference resolution of different NP types. Yang et al. (2005) made use of non-anaphors to create a special class of training instances in the twin-candidate model (Yang et al. 2003) and improved the performance by 2.9 and 1.6 to 67.3 and 67.2 in F1-measure on the MUC-6 and MUC-7 corpora, respectively. However, their experiments show that eliminating non-anaphors using an anaphoricity determination module in advance harms the performance. Denis and Balbridge (2007) employed an integer linear programming (ILP) formulation for coreference resolution which modeled anaphoricity and coreference as a joint task, such that each local model informed the other for the final assignments. Experiments on the ACE 2003 corpus showed that this joint anaphoricity-coreference ILP formulation improved the F1-measure by 3.7-5.3 on various domains. However, their experiments assume true ACE mentions (i.e. all the ACE mentions are already known from the annotated corpus). Therefore, the actual effect of this joint anaphoricity-coreference ILP formulation on fully automatic coreference resolution is still unclear. Luo (2007) proposed a twin-model for coreference resolution: a link component, which models the coreferential relationship between an anaphor and a candidate antecedent, and a creation component, which models the possibility that a NP was not coreferential with any candidate antecedent. This method combined the probabilities returned by the creation component (an anaphoricity model) with the link component (a coreference model) to score a coreference partition, such that a partition was penalized whenever an anaphoric mention was resolved. Finkel and Manning (2008) showed that transitivity constraints could be incorporated into an ILP-based coreference resolution system and much improved the performance. Zhou and Kong (2009) employed a global learning method in determining the anaphoricity of NPs via a label propagation algorithm to improve learning-based coreference resolution. Experiments on the ACE 2003 corpus demonstrated that this method was very effective. It could improve the F1-measure by 2.4, 3.1 and 4.1 on the NWIRE, NPAPER and BNEWS domains, respectively. Ng (2009) presented a novel approach to the task of anaphoricity determination based on graph minimum cuts and demonstrated the effectiveness in improving a learning-based coreference resolution system.

In summary, although anaphoricity determination plays an important role in coreference resolution and achieves certain success in improving the overall performance of coreference resolution, its contribution is still far from expectation.

2.2 Syntactic Parse Tree Structures

For a tree kernel-based method, one key problem is how to represent and capture the structured syntactic information. During recent years, various tree kernels, such as the convolution tree kernel (Collins and Duffy, 2001), the shallow parse tree kernel (Zelenko et al 2003) and the dependency tree kernel (Culota and Sorensen, 2004), have been proposed in the literature. Among these tree kernels, the convolution tree kernel represents the state-of-the art and has been successfully applied by Collins and Duffy (2002) on syntactic parsing, Zhang et al. (2006) on semantic relation extraction and Yang et al. (2006) on pronoun resolution.

Given a tree kernel, the key issue is how to generate a syntactic parse tree structure for effectively capturing the structured syntactic information. In the literature, various parse tree structures have been proposed and successfully applied in some NLP applications. As a representative, Zhang et al. (2006) investigated five parse tree structures for semantic relation extraction and found that the Shortest Path-enclosed Tree (SPT) achieves the best performance on the 7 relation types of the ACE RDC 2004 corpus. Yang et al. (2006) constructed a document-level syntactic parse tree for an entire text by attaching the parse trees of all its sentences to a new-added upper node and examined three possible parse tree structures (Min-Expansion, Simple-Expansion and Full-Expansion) that contain different substructures of the parse tree for pronoun resolution. Experiments showed that their method achieved certain success on the ACE 2003 corpus and the simple-expansion scheme performs best. However, among the three explored schemes, there exists no obvious overwhelming one, which can well cover structured syntactic information. One problem of Zhang et al. (2006)

and Yang et al. (2006) is that their parse tree structures are context-free and do not consider the information outside the sub-trees. Hence, their ability of exploring structured syntactic information is much limited. Motivated by Zhang et al. (2006) and Yang et al. (2006), Zhou et al. (2007) extended the SPT to become context-sensitive (CS-SPT) by dynamically including necessary predicate-linked path information. Zhou et al. (2008) further proposed a dynamic-expansion scheme to automatically determine a proper parse tree structure for pronoun resolution by taking predicate- and antecedent competitor-related information in consideration. Evaluation on the ACE 2003 corpus showed that the dynamic-expansion scheme can well cover necessary structured information in the parse tree for pronoun resolution. One problem with the above parse tree structures is that they may still contain unnecessary information and also miss some useful context-sensitive information. Qian et al. (2008) dynamically determined the parse tree structure for semantic relation extraction by exploiting constituent dependencies to keep the necessary information in the parse tree as well as remove the noisy information. Evaluation on the ACE RDC 2004 corpus showed that their dynamic syntactic parse tree structure outperforms all previous parse tree structures. However, their solution has the limitation in that the dependencies were found according to some manually-written ad-hoc rules and thus may not be easily applicable to new domains and applications.

This paper proposes a new scheme to dynamically determine the syntactic parse tree structure for anaphoricity determination and systematically studies the application of an explicit anaphoricity determination module in improving coreference resolution.

3 Dependency-driven Dynamic Syntactic Parse Tree

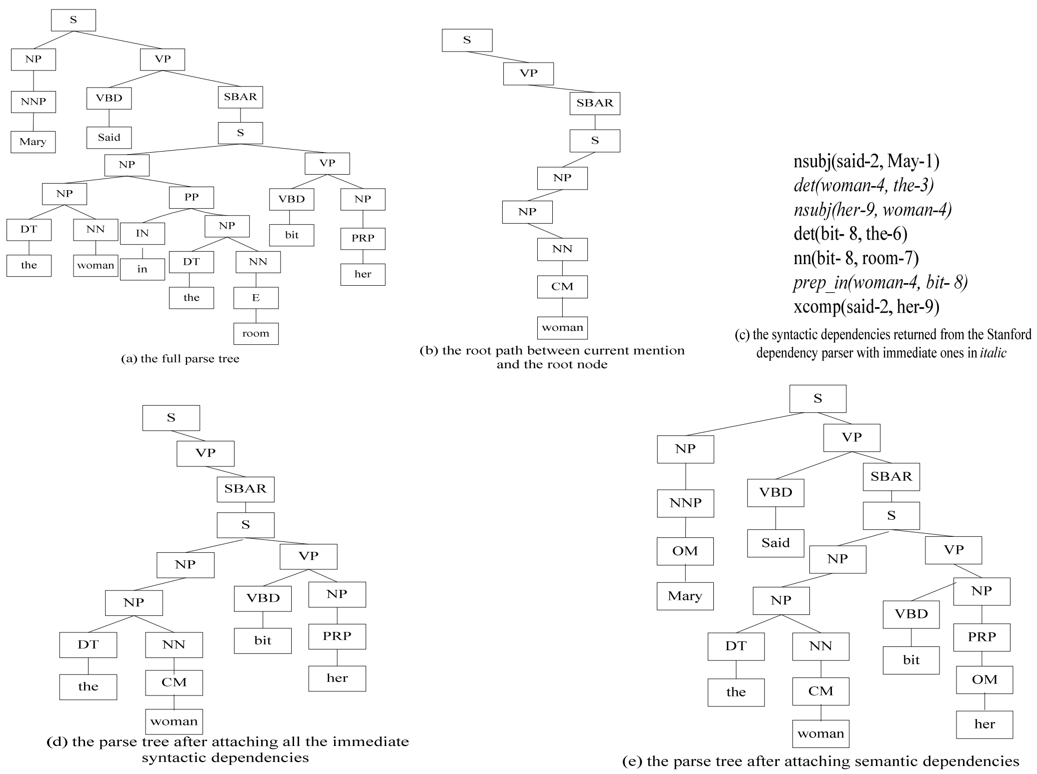
Given a full syntactic parse tree and a NP in consideration, one key issue is how to choose a proper syntactic parse tree structure to well cover structured syntactic information in the tree kernel computation. Generally, the more a syntactic parse tree structure includes, the more structured syntactic information would be available, at the expense of more noisy (or unnecessary) information.

It is well known that dependency information plays a key role in many NLP problems, such as syntactic parsing, semantic role labeling as well as semantic relation extraction. Motivated by Qian et al. (2008) and Zhou et al. (2008), we propose a new scheme to dynamically determine the syntactic parse tree structure for anaphoricity determination by exploiting constituent dependencies from both the syntactic and semantic perspectives to distinguish the necessary evidence from the unnecessary information in the syntactic parse tree. That is, constituent dependencies are explored from two aspects: syntactic dependencies and semantic dependencies.

1) **Syntactic Dependencies:** The Stanford dependency parser[1] is employed as our syntactic dependency parser to automatically extract various syntactic (i.e. grammatical) dependencies between individual words. In this paper, only immediate syntactic dependencies with current mention are considered. The intuition behind is that the immediate syntactic dependencies carry the major contextual information of current mention.
2) **Semantic Dependencies:** A state-of-the-art semantic role labeling (SRL) toolkit (Li et al. 2009) is employed for extracting various semantic dependencies related with current mention. In this paper, semantic dependencies include all the predicates heading any node in the root path from current mention to the root node and their compatible arguments (except those overlapping with current mention).

We name our parse tree structure as a dependency-driven dynamic syntactic parse tree (D-DSPT). The intuition behind is that the dependency information related with current mention in the same sentence plays a critical role in anaphoricity determination. Given the sentence enclosing the mention under consideration, we can get the D-DSPT as follows: (Figure 1 illustrates an example of the D-DSPT generation given the sentence “Mary said the woman in the room bit her” with “woman” as current mention.)

[1] http://nlp.stanford.edu/software/lex-parser.shtml



(a) the full parse tree

(b) the root path between current mention and the root node

nsubj(said-2, May-1)
det(woman-4, the-3)
nsubj(her-9, woman-4)
det(bit- 8, the-6)
nn(bit- 8, room-7)
prep_in(woman-4, bit- 8)
xcomp(said-2, her-9)

(c) the syntactic dependencies returned from the Stanford dependency parser with immediate ones in *italic*

(d) the parse tree after attaching all the immediate syntactic dependencies

(e) the parse tree after attaching semantic dependencies

Figure 1: An example of generating the dependency-driven dynamic syntactic parse tree

1) Generating the full syntactic parse tree of the given sentence using a full syntactic parser. In this paper, the Charniak parser (Charniak 2001) is employed and Figure 1 (a) shows the resulting full parse tree.

2) Keeping only the root path from current mention to the root node of the full parse tree. Figure 1(b) shows the root path corresponding to the current mention "woman". In the following steps, we attach the above two types of dependency information to the root path.

3) Extracting all the syntactic dependencies in the sentence using a syntactic dependency parser, and attaching all the nodes, which have immediate dependency relationship with current mention, and their corresponding paths to the root path. Figure 1(c) illustrates the syntactic dependences extracted from the sentence, where the ones in italic mean immediate dependencies with current mention. Figure 1(d) shows the parse tree structure after considering syntactic dependencies.

4) Attaching all the predicates heading any node in the root path from current mention to the root node and their corresponding paths to the root path. For the example sentence, there are two predicates "said" and "bit", which head the "VP" and "S" nodes in the root path respectively. Therefore, these two predicates and their corresponding paths should be attached to the root path as shown in Figure 1(e). Note that the predicate "bit" and its corresponding path has already been attached in Stop (3). As a result, the predicate-related information can be attached. According to Zhou and Kong (2009), such information is important to definite NP resolution.

5) Extracting the semantic dependencies related with those attached predicates using a (shallow) semantic parser, and attaching all the compatible arguments (except those overlapping with current mention) and their corresponding paths to the root path. For example, as shown in Figure 1(e), since the arguments "Mary" and "her" are compatible with current mention "woman", these two nodes and their corresponding paths are attached while the argument "room" is not since its gender does not agree with current mention.

In this paper, the similarity between two parse trees is measured using a convolution tree kernel, which counts the number of common sub-tree as the syntactic structure similarity between two parse trees. For details, please refer to Collins and Duffy (2001).

4 Experimentation and Discussion

This section evaluates the performance of dependency-driven anaphoricity determination and its application in coreference resolution on the ACE 2003 corpus.

4.1 Experimental Setting

The ACE 2003 corpus contains three domains: newswire (NWIRE), newspaper (NPAPER), and broadcast news (BNEWS). For each domain, there exist two data sets, training and devtest, which are used for training and testing.

For preparation, all the documents in the corpus are preprocessed automatically using a pipeline of NLP components, including tokenization and sentence segmentation, named entity recognition, part-of-speech tagging and noun phrase chunking. Among them, named entity recognition, part-of-speech tagging and noun phrase chunking apply the same state-of-the-art HMM-based engine with error-driven learning capability (Zhou and Su, 2000 & 2002). Our statistics finds that 62.0%, 58.5% and 61.4% of entity mentions are preserved after preprocessing on the NWIRE, NPAPER and BNEWS domains of the ACE 2003 training data respectively while only 89.5%, 89.2% and 94% of entity mentions are preserved after preprocessing on the NWIRE, NPAPER and BNEWS domains of the ACE 2003 devtest data. This indicates the difficulty of coreference resolution. In addition, the corpus is parsed using the Charniak parser for syntactic parsing and the Stanford dependency parser for syntactic dependencies while corresponding semantic dependencies are extracted using a state-of-the-art semantic role labeling toolkit (Li et al. 2009). Finally, we use the SVM-light[2] toolkit with the tree kernel function as the classifier. For comparison purpose, the training parameters C (SVM) and λ (tree kernel) are set to 2.4 and 0.4 respectively, as done in Zhou and Kong (2009).

For anaphoricity determination, we report the performance in Acc^+ and Acc^-, which measure the accuracies of identifying anaphoric NPs and non-anaphoric NPs, respectively. Obviously, higher Acc^+ means that more anaphoric NPs would be identified correctly, while higher Acc^- means that more non-anaphoric NPs would be filtered out. For coreference resolution, we report the performance in terms of recall, precision, and F1-measure using the commonly-used model theoretic MUC scoring program (Vilain et al. 1995). To see whether an improvement is significant, we also conduct significance testing using paired t-test. In this paper, '***', '**' and '*' denote p-values of an improvement smaller than 0.01, in-between (0.01, 0,05] and bigger than 0.05, which mean significantly better, moderately better and slightly better, respectively.

4.2 Experimental Results

Performance of anaphoricity determination

Table 1 presents the performance of anaphoricity determination using the convolution tree kernel on D-DSPT. It shows that our method achieves the accuracies of 83.27/77.13, 86.77/80.25 and 90.02/64.24 on identifying anaphoric/non-anaphoric NPs in the NWIRE, NPAPER and BNEWS domains, respectively. This suggests that our approach can effectively filter out about 75% of non-anaphoric NPs and keep about 85% of anaphoric NPs. In comparison, in the three domains Zhou and Kong (2009) achieve the accuracies of 76.5/82.3, 78.9/81.6 and 74.3/83.2, respectively, using the tree kernel on a dynamically-extended tree (DET). This suggests that their method can filter out about 82% of non-anaphoric NPs and only keep about 76% of anaphoric NPs. In comparison, their method outperforms our method on filtering out more non-anaphoric NPs while our method outperforms their method on keeping more anaphoric NPs in coreference resolution. While a coreference resolution system can detect some non-anaphoric NPs (when failing to find the antecedent candidate), filtering out anaphoric NPs in anaphoricity determination would definitely cause errors and it is almost impossible to recover. Therefore, it is normally more important to keeping more anaphoric NPs than filtering out more non-anaphoric NPs. Table 1 further presents the performance of anaphoricity determination on different NP types. It shows that our method performs best at keeping pronominal NPs and filtering out proper NPs.

[2] http://svmlight.joachims.org/

NP Type	NWIRE		NPAPER		BNEWS	
	Acc$^+$	Acc$^-$	**Acc$^+$**	Acc$^-$	**Acc$^+$**	Acc$^-$
Pronoun	95.07	50.36	96.40	56.44	98.26	54.03
Proper NP	84.61	83.17	83.78	79.62	87.61	71.77
Definite NP	87.17	46.74	82.24	49.18	86.87	53.65
Indefinite NP	86.01	47.52	80.63	48.45	89.71	47.32
Over all	83.27	77.13	86.77	80.25	90.02	64.24

Table 1: Performance of anaphoricity determination using the D-DSPT

Performance Change	NWIRE		NPAPER		BNEWS	
	Acc$^+$	Acc$^-$	**Acc$^+$**	Acc$^-$	**Acc$^+$**	Acc$^-$
D-DSPT	83.27	77.13	86.77	80.25	90.02	64.24
-Syntactic Dependencies	78.67	72.56	80.14	73.74	87.05	60.20
-Semantic Dependencies	81.67	76.74	83.47	77.93	89.58	60.67

Table 2: Contribution of including syntactic and semantic dependencies in D-DSPT on anaphoricity determination

System		NWIRE			NPAPER			BNEWS		
		R%	P%	F	R%	P%	F	R%	P%	F
Without anaphoricity determination (Baseline)	Pronoun	70.8	57.9	63.7	76.5	63.5	69.4	70.0	60.3	64.8
	Proper NP	80.3	80.1	80.2	81.8	83.6	82.7	76.3	76.8	76.6
	Definite NP	35.9	43.4	39.2	43.1	48.5	45.6	47.9	51.9	49.8
	Indefinite NP	40.3	26.3	31.8	39.7	22.9	29.0	23.6	10.7	14.7
	Over all	***55.0***	***63.8***	***59.1***	***62.1***	***65.0***	***63.5***	***53.2***	***60.5***	***56.6***
With D-DSPT -based anaphoricity determination	Pronoun	65.9	70.2	68.0	72.6	78.7	75.5	67.7	75.8	71.5
	Proper NP	80.3	81.0	80.6	81.2	85.1	83.1	76.3	84.4	80.1
	Definite NP	32.3	63.1	42.7	38.4	61.7	47.3	42.5	66.4	51.8
	Indefinite NP	36.4	55.3	43.9	34.7	50.7	41.2	20.3	45.4	28.1
	Over all	***52.4***	***79.6***	***63.2***	***58.1***	***80.3***	***67.4***	***50.1***	***79.8***	***61.6***
With golden anaphoricity determination	Pronoun	68.6	71.5	70.1	75.2	80.4	77.7	69.1	77.8	73.5
	Proper NP	81.7	89.3	85.3	82.6	90.1	86.2	78.6	88.7	83.3
	Definite NP	41.8	85.9	56.2	44.9	85.2	58.8	45.2	87.9	59.7
	Indefinite NP	40.3	67.6	50.5	41.2	65.1	50.5	40.9	50.1	45.1
	Over all	***54.6***	***81.7***	***65.5***	***60.4***	***82.1***	***69.6***	**51.9**	**82.1**	**63.6**

Table 3: Performance of anaphoricity determination on coreference resolution

System		NWIRE			NPAPER			BNEWS		
		R%	P%	F	R%	P%	F	R%	P%	F
Zhou and Kong (2009)	Without anaphoricity determination (Baseline)	*53.1*	*67.4*	*59.4*	*57.7*	*67.0*	*62.1*	*48.0*	*65.9*	*55.5*
	With Dynamically Extended Tree-based anaphoricity determination	*51.6*	*77.2*	*61.8*	*55.2*	*78.6*	*65.2*	*47.5*	*80.3*	*59.6*
Ng (2009)	Without anaphoricity determination (Baseline)	*59.1*	*58.*	*58.6*	*60.8*	*62.6*	*61.7*	*57.7*	*52.6*	*55.0*
	With Graph Minimum Cut-based anaphoricity determination	*54.1*	*69.0*	*60.6*	*57.9*	*71.2*	*63.9*	*53.1*	*67.5*	*59.4*

Table 4: Performance comparison with other systems

Table 2 further presents the contribution of including syntactic and semantic dependencies in the D-DSPT on anaphoricity determination by excluding one or both of them. It shows that both syntactic dependencies and semantic dependencies contribute significantly (***).

Performance of coreference resolution

We have evaluated the effect of our D-DSPT-based anaphoricity determination module on coreference resolution by including it as a preprocessing step to a baseline coreference resolution system without explicit anaphoricity determination, by filtering our those non-anaphoric NPs according to the anaphoricity determination module. Here, the baseline system employs the same set of features, as adopted in the single-candidate model of Yang et al. (2003) and uses a SVM-based classifier with the feature-based RBF kernel. Table 3 presents the detailed performance of the coreference resolution system without anaphoricity determination, with D-DSPT-based anaphoricity determination and. with golden anaphoricity determination. Table 3 shows that:

1) There is a performance gap of 6.4, 6.1 and 7.0 in F1-measure on the NWIRE, NPAPER and BNEWS domain, respectively, between the coreference resolution system with golden anaphoricity determination and the baseline system without anaphoricity determination. This suggests the usefulness of proper anaphoricity determination in coreference resolution. This also agrees with Stoyanov et al. (2009) which measured the impact of golden anaphoricity determination on coreference resolution using only the annotated anaphors in both training and testing.

2) Compared to the baseline system without anaphoricity determination, the D-DSPT-based anaphoricity determination module improves the performance by 4.1(***), 3.9(***) and 5.0(***) to 63.2, 67.4 and 61.6 in F1-measure on the NWIRE, NPAPER and BNEWS domains, respectively, due to a large gain in precision and a much smaller drop in recall. In addition, D-DSPT-based anaphoricity determination can not only much improve the performance of coreference resolution on pronominal NPs (***) but also on definite NPs(***) and indefinite NPs(***) while the improvement on proper NPs can be ignored due to the fact that proper NPs can be well addressed by the simple abbreviation feature in the baseline system.

3) D-DSPT-based anaphoricity determination still lags (2.3, 2.2 and 2.0 on the NWIRE, NPAPER and BNEWS domains, respectively) behind golden anaphoricity determination in improving the overall performance of coreference resolution. This suggests that there exists some room in the performance improvement for anaphoricity determination.

Performance comparison with other systems

Table 4 compares the performance of our system with other systems. Here, Zhou and Kong (2009) use the same set of features with ours in the baseline system and a dynamically-extended tree structure in anaphoricity determination. Ng (2009) uses 33 features as described in Ng (2007) and a graph minimum cut algorithm in anaphoricity determination. It shows that the overall performance of our baseline system is almost as good as that of Zhou and Kong (2009) and a bit better than Ng's (2009).

For overall performance, our coreference resolution system with D-DSPT-based anaphoricity determination much outperforms Zhou and Kong (2009) in F1-measure by 1.4, 2.2 and 2.0 on the NWIRE, NPAPER and BNEWS domains, respectively, due to the better inclusion of dependency information. Detailed evaluation shows that such improvement comes from coreference resolution on both pronominal and definite NPs (Please refer to Table 6 in Zhou and Kong, 2009). Compared with Zhou and Kong (2009) and Ng (2009), our approach achieves the best F1-measure so far for each dataset.

5 Conclusion and Further Work

This paper systematically studies a dependency-driven dynamic syntactic parse tree (DDST) for anaphoricity determination and the application of an explicit anaphoricity determination module in improving learning-based coreference resolution. Evaluation on the ACE 2003 corpus indicates that D-DSPT-based anaphoricity determination much improves the performance of coreference resolution.

To our best knowledge, this paper is the first research which directly explores constituent dependencies in tree kernel-based anaphoricty determination from both syntactic and semantic perspectives.

For further work, we will explore more structured syntactic information in coreference resolution. In addition, we will study the interaction between anaphoricity determination and coreference resolution and better integrate anaphoricity determination with coreference resolution.

Acknowledgments

This research was supported by Projects 60873150, 60970056, and 90920004 under the National Natural Science Foundation of China, Project 200802850006 and 20093201110006 under the Specialized Research Fund for the Doctoral Program of Higher Education of China.

References

D. Bean and E. Riloff 1999. Corpus-based Identification of Non-Anaphoric Noun Phrases. ACL' 1999

S. Bergsma, D. Lin and R. Goebel 2008. Distributional Identification of Non-referential Pronouns. ACL'2008

C. Cherry and S. Bergsma. 2005. An expectation maximization approach to pronoun resolution. CoNLL'2005

M. Collins and N. Duffy. 2001. Covolution kernels for natural language. NIPS' 2001

M. Denber 1998. Automatic Resolution of Anaphoria in English. Technical Report, Eastman KodakCo.

P. Denis and J. Baldridge. 2007. Global, joint determination of anaphoricity and coreference resolution using integer programming. NAACL/HLT'2007

R. Evans 2001. Applying machine learning toward an automatic classification of it. Literary and Linguistic Computing, 16(1):45-57

F. Kong, G.D. Zhou and Q.M. Zhu. 2009 Employing the Centering Theory in Pronoun Resolution from the Semantic Perspective. EMNLP'2009

F. Kong, Y.C. Li, G.D. Zhou and Q.M. Zhu. 2009. Exploring Syntactic Features for Pronoun Resolution Using Context-Sensitive Convolution Tree Kernel. IALP'2009

S. Lappin and J. L. Herbert. 1994. An algorithm for pronominal anaphora resolution. Computational Linguistics, 20(4)

J.H. Li. G.D. Zhou, H. Zhao, Q.M. Zhu and P.D. Qian. Improving nominal SRL in Chinese language with verbal SRL information and automatic predicate recognition. EMNLP '2009

X. Luo. 2007. Coreference or not: A twin model for coreference resolution. NAACL-HLT'2007

V. Ng and C. Cardie 2002. Identify Anaphoric and Non-Anaphoric Noun Phrases to Improve Coreference Resolution. COLING'2002

V. Ng and C. Cardie 2002. Improving machine learning approaches to coreference resolution. ACL'2002

V. Ng 2004. Learning Noun Phrase Anaphoricity to Improve Coreference Resolution: Issues in Representation and Optimization. ACL' 2004

V. Ng 2009. Graph-cut based anaphoricity determination for coreference resolution. NAACL'2009

L.H. Qian, G.D. Zhou, F. Kong, Q.M. Zhu and P.D. Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. COLING'2008

W. M. Soon, H. T. Ng and D. Lim 2001. A machine learning approach to coreference resolution of noun phrase. Computational Linguistics, 27(4):521-544.

V. Stoyanov, N. Gilbert, C. Cardie and E. Riloff. 2009. Conundrums in Noun Phrase Coreference Resolution: Making Sense of the State-of-the Art. ACL'2009

B. L. Webber. 1979. A Formal Approach to Discourse Anaphora. Garland Publishing, Inc.

X.F. Yang, G.D. Zhou, J. Su and C.L. Chew. 2003. Coreference Resolution Using Competition Learning Approach. ACL'2003

X.F. Yang, J. Su and C.L. Chew. 2005. A Twin Candidate Model of Coreference Resolution with Non-Anaphor Identification Capability. IJCNLP'2005

X.F. Yang, J. Su and C.L. Chew. 2006. Kernel-based pronoun resolution with structured syntactic knowledge. COLING-ACL'2006

X.F. Yang, J. Su and C.L. Tan 2008. A Twin-Candidate Model for Learning-Based Anaphora Resolution. Computational Linguistics 34(3):327-356

M. Zhang, J. Zhang, J. Su and G.D. Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. COLING/ACL'2006

S. Zhao and R. Grishman. 2005. Extracting relations with integered information using kernel methods. ACL'2005

D. Zelenko, A. Chinatsu and R. Anthony. 2003. Kernel methods for relation extraction. Machine Learning Researching 3(2003):1083-1106

G.D. Zhou, F. Kong and Q.M. Zhu. 2008. Context-sensitive convolution tree kernel for pronoun resolution. IJCNLP'2008

G.D. Zhou and F. Kong. 2009. Global Learning of Noun Phrase Anaphoricity in Coreference Resolution via Label Propagetion. EMNLP'2009

G.D. Zhou and J. Su. 2002. Named Entity recognition using a HMM-based chunk tagger. ACL'2002

G.D. Zhou, M. Zhang, D.H. Ji and Q.M. Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. EMNLP/CoNLL'2007

Phrase Clustering for Smoothing TM Probabilities – or, How to Extract Paraphrases from Phrase Tables

[1]Roland Kuhn, [1]Boxing Chen, [1]George Foster and [2]Evan Stratford
[1]National Research Council of Canada
[2]University of Waterloo
[1]First.Last@nrc.gc.ca; [2]evan.stratford@gmail.com

Abstract

This paper describes how to cluster together the phrases of a phrase-based statistical machine translation (SMT) system, using information in the phrase table itself. The clustering is symmetric and recursive: it is applied both to source-language and target-language phrases, and the clustering in one language helps determine the clustering in the other. The phrase clusters have many possible uses. This paper looks at one of these uses: smoothing the conditional translation model (TM) probabilities employed by the SMT system. We incorporated phrase-cluster-derived probability estimates into a baseline loglinear feature combination that included relative frequency and lexically-weighted conditional probability estimates. In Chinese-English (C-E) and French-English (F-E) learning curve experiments, we obtained a gain over the baseline in 29 of 30 tests, with a maximum gain of 0.55 BLEU points (though most gains were fairly small). The largest gains came with medium (200-400K sentence pairs) rather than with small (less than 100K sentence pairs) amounts of training data, contrary to what one would expect from the paraphrasing literature. We have only begun to explore the original smoothing approach described here.

1 Introduction and Related Work

The source-language and target-language "phrases" employed by many statistical machine translation (SMT) systems are anomalous: they are arbitrary sequences of contiguous words extracted by complex heuristics from a bilingual corpus, satisfying no formal linguistic criteria. Nevertheless, phrase-based systems perform better than word-based systems (Koehn 2010, pp. 127-129). In this paper, we look at what happens when we cluster together these anomalous but useful entities.

Here, we apply phrase clustering to obtain better estimates for "backward" probability P(**s**|**t**) and "forward" probability P(**t**|**s**), where **s** is a source-language phrase, **t** is a target-language phrase, and phrase pair (**s**,**t**) was seen at least once in training data. The current work is thus related to work on smoothing P(**s**|**t**) and P(**t**|**s**) – see (Foster *et al.*, 2006). The relative frequency estimates for P(**s**|**t**) and P(**t**|**s**) are $P_{RF}(s \mid t) = \#(s,t)/\#t$ and $P_{RF}(t \mid s) = \#(s,t)/\#s$, where *#(s,t)* denotes the number of times phrase pair (**s**,**t**) was observed, *etc*. These estimates are typically smoothed with "lexical" estimates found by breaking phrases **s** and **t** into words. We adopt a different idea, that of smoothing P_{RF}(**s**|**t**) and P_{RF}(**t**|**s**) with estimates obtained from phrases that have similar meanings to **s** and **t**. In our experiments, the two methods were combined, yielding an improvement over lexical smoothing alone – this indicates they provide complementary information. *E.g.*, lexical estimates don't work well for non-compositional phrases like "kick the bucket" - our method might cluster this phrase with "die" and "expire" and thus provide better smoothing. The research that comes closest to ours is the work of Schwenk *et al.* (2007) on continuous space N-gram models, where a neural network is employed to smooth translation probabilities. However, both Schwenk *et al.*'s smoothing technique

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 608–616,
Beijing, August 2010

and the system to which it is applied are quite different from ours.

Phrase clustering is also somewhat related to work on paraphrases for SMT. Key papers in this area include (Bannard and Callison-Burch, 2005), which pioneered the extraction of paraphrases from bilingual parallel corpora, (Callison-Burch *et al.*, 2006) which showed that paraphrase generation could improve SMT performance, (Callison-Burch, 2008) and (Zhao *et al.*, 2008) which showed how to improve the quality of paraphrases, and (Marton *et al.*, 2009) which derived paraphrases from monolingual data using distributional information. Paraphrases typically help SMT systems trained on under 100K sentence pairs the most.

The phrase clustering algorithm in this paper outputs groups of source-language and target-language phrases with similar meanings: paraphrases. However, previous work on paraphrases for SMT has aimed at finding translations for source-language phrases in the system's input that weren't seen during system training. Our approach is completely useless in this situation: it only generates new information for target or source phrases that are already in the system's phrase table. Thus, we find paraphrases for many of the source and target phrases that **are** in the phrase table, while the work cited above looks for paraphrases of source phrases that are **not** in the phrase table.

Our work also differs from most work on paraphrases in that information is extracted not from sources outside the SMT system (*e.g.*, pivot languages or thesauri) but from the system's phrase table. In this respect if no other, it is similar to Chiang's classic work on hierarchical phrase-based systems (Chiang, 2005), though Chiang was mining a very different type of information from phrase tables.

Because of all these differences between work on paraphrasing and the phrase clustering approach, both in terms of the input information and where they are best applied, we did not experimentally compare the two approaches.

2 Deriving Conditional Probabilities from Phrase Clusters

Given phrase clusters in the source and target languages, how would one derive estimates for conditional probabilities P(**s**|**t**) and P(**t**|**s**)? We assume that the clustering is "hard": each source phrase **s** belongs to exactly one cluster C(**s**), and each target phrase **t** belongs to exactly one cluster C(**t**). Some of these clusters will contain singleton phrases, and others will contain more than one phrase. Let "#" denote the total number of observations in the training data associated with a phrase or phrase cluster. *E.g.*, suppose the English cluster C_S contains the three phrases "red", "dark red", and "burgundy", with 50, 25, and 10 observations in the training data respectively – then $\#(C_S) = 85$. Also, let $\#(C_S,C_T)$ be the number of co-occurrences in the training data of source-language cluster C_S and target-language cluster C_T.

The phrase-cluster-based probabilities P_{PC} are:

$$P_{PC}(s \mid t) = P(s \mid C(s)) \times P(C(s) \mid C(t)) = \frac{\#(s)}{\#C(s)} \times \frac{\#(C(s),C(t))}{\#C(t)} \quad (1)$$

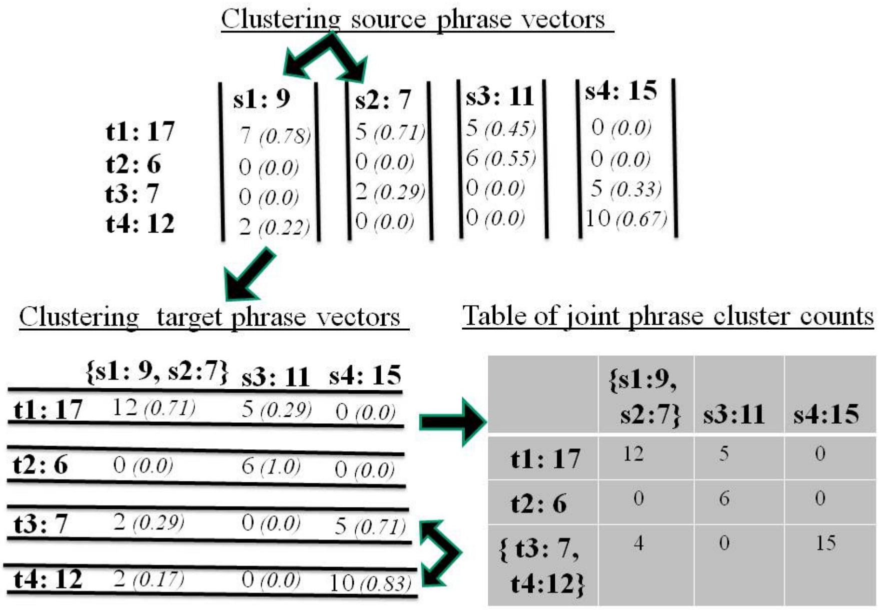
and

$$P_{PC}(t \mid s) = P(t \mid C(t)) \times P(C(t) \mid C(s)) = \frac{\#(t)}{\#C(t)} \times \frac{\#(C(s),C(t))}{\#C(s)} \quad (2)$$

Note that the P_{PC} will often be non-zero where the corresponding relative frequency estimates P_{RF} were zero (the opposite can't happen). Also, the P_{PC} will be most useful where the phrase being conditioned on was seldom seen in the training data. If **t** was seen 1,000 times during training, the P_{RF}(**s**|**t**) are reliable and don't need smoothing; but if **t** was seen 6 times, P_{PC}(**s**|**t**) may yield valuable extra information. The same kind of argument applies to estimation of P(**t**|**s**).

3 Clustering Phrases

We used only information "native" to phrase tables to cluster phrases. Two types of similarity metric between phrases or phrase clusters were employed: count-based metrics and edit-based metrics. The former are based on phrase co-occurrence counts; the latter are based on the word sequences that make up the phrases. Each has its advantages. Count-based metrics can deduce from the similar translations of two phrases that they have similar meanings, despite dissimilarity between the two word sequences – *e.g.*, they can deduce that "red" and "burgundy" belong in the same cluster. However, these metrics are unreliable when total counts are low, since phrase co-occurrences are determined by a noisy alignment process. Edit-based metrics are independent of how often phrases were observed. However, sometimes they can be fooled by phrases that have similar word sequences but different meanings (*e.g.*, "the dog bit the man"


Clustering source phrase vectors
Clustering target phrase vectors
Table of joint phrase cluster counts

	s1: 9	s2: 7	s3: 11	s4: 15
t1: 17	7 *(0.78)*	5 *(0.71)*	5 *(0.45)*	0 *(0.0)*
t2: 6	0 *(0.0)*	0 *(0.0)*	6 *(0.55)*	0 *(0.0)*
t3: 7	0 *(0.0)*	2 *(0.29)*	0 *(0.0)*	5 *(0.33)*
t4: 12	2 *(0.22)*	0 *(0.0)*	0 *(0.0)*	10 *(0.67)*

	{s1: 9, s2:7}	s3: 11	s4: 15
t1: 17	12 *(0.71)*	5 *(0.29)*	0 *(0.0)*
t2: 6	0 *(0.0)*	6 *(1.0)*	0 *(0.0)*
t3: 7	2 *(0.29)*	0 *(0.0)*	5 *(0.71)*
t4: 12	2 *(0.17)*	0 *(0.0)*	10 *(0.83)*

	{s1:9, s2:7}	s3:11	s4:15
t1: 17	12	5	0
t2: 6	0	6	0
{ t3: 7, t4:12}	4	0	15

Figure 1: Example of phrase clustering

and "the man bit the dog", or "walk on the beach" and "don't walk on the beach"). In our experiments, we used a combination of count-based and edit-based metrics to cluster phrases (by simply multiplying the metrics together). However, we invested most of our effort in perfecting the count-based component: our edit-based metric was fairly naïve.

If we rely mainly on count-based similarity between phrases to cluster them, and this kind of similarity is most reliable when phrases have high counts, yet we need phrase-cluster-based estimates most for phrases with low counts, aren't we carrying out clustering on the phrases that need it least? Our hope was that there is a class of phrases with intermediate counts (*e.g.*, with 3-15 observations in the training data) that can be clustered reliably, but still benefit from phrase-cluster-based probability estimates.

3.1 Count-based clustering: overview

Figure 1 shows count-based phrase clustering. One first arbitrarily picks a language (either source or target) and then clusters together some of the phrases in that language. One then switches to the other language and clusters phrases in that language, then switches back to the first one, and so on until enough clustering has taken place.

Each phrase or phrase cluster is represented by the vector of its co-occurrence counts. To calculate the similarity between two phrase clusters, one first normalizes their count vectors. At the top of **Figure 1**, source phrase **s1** occurred 9 times: 7 times aligned with target phrase **t1**, 2 times aligned with **t4**. For source similarity computation, the entry for (**s1**,**t1**) is normalized to 7/9 = 0.78 and the entry for (**s1**,**t4**) is normalized to 2/9 = 0.22 (these normalized values are shown in brackets and italics after the counts).

The two most similar normalized vectors at the top of **Figure 1** are those associated with phrases **s1** and **s2**. These phrases are merged by adding corresponding counts, yielding a new vector associated with the new phrase cluster {**s1**, **s2**}. In real life, one would now do more source-language clustering on the source language side; in this example, we immediately proceed to target-language clustering (carried out in target language space). Note that the target similarity calculations are affected by the previous source clustering (because **s1** and **s2** are now represented by the same coordinate, **t3** and **t4** are now closer than they were in the initial table). In this manner, we can iterate back and forth between the two languages. The final output is a table of joint phrase cluster counts, which is used to estimate the P_{PC} (see previous section).

3.2 Count-based clustering: details

Count-based similarity is computed as follows:

1. Phrase alignment is a noisy process, so we first apply a transformation analogous to *tf-idf* in information retrieval (Salton and McGill, 1986) to phrase cluster

counts. For source similarity computation, each co-occurrence count #(C$_{\mathbf{S}}$,C$_{\mathbf{T}}$) between source cluster C$_{\mathbf{S}}$ and target cluster C$_{\mathbf{T}}$ is multiplied by a factor that reflects the information content of C$_{\mathbf{T}}$. Let #diff(C$_{\mathbf{S}}$) be number of clusters on the source side, and let #[C$_{\mathbf{T}}$>0] for a particular target cluster C$_{\mathbf{T}}$ be the number of source clusters C$_{\mathbf{S}}$ that co-occur with C$_{\mathbf{T}}$. Then let $\#'(C_S,C_T)=\#(C_S,C_T)\times\log(\#\mathit{diff}(C_S)/\#[C_T>0])$. Similarly, for target similarity computation, let $\#'(C_S,C_T)=\#(C_S,C_T)\times\log(\#\mathit{diff}(C_T)/\#[C_S>0])$. *E.g.*, in source similarity computation, if C$_{\mathbf{T}}$ co-occurs with all source clusters, its contribution will be set to zero (because it carries little information).

2. We normalize by dividing each vector of *tf-idf* counts $\#'(C_S,C_T)$ by the total number of observations in the vector.
3. We compute the similarity between each pair of *tf-idf* vectors using either the cosine measure (Salton and McGill, 1986) or one of a family of probabilistic metrics described below.
4. We cluster together the most similar vectors; this involves summing the unmodified counts #(C$_{\mathbf{S}}$,C$_{\mathbf{T}}$) of the vectors (*i.e.*, the *tf-idf* transformation is only applied for the purposes of similarity calculation and is not retained).

Now, we'll describe the probabilistic metrics we considered. For a count vector of dimension D, **u** = (u_1, u_2, …, u_D), define a function $I(\underline{\mathbf{u}})=u_1\times\log(u_1/\sum_i u_i)+\ldots+u_D\times\log(u_D/\sum_i u_i)$. *I*(**u**) is a measure of how well the data in **u** are modeled by the normalized vector ($u_1/\Sigma_i u_i$, …, $u_D/\Sigma_i u_i$). Thus, when two count vectors **u** and **v** are merged (by adding them) we have the following measure of the loss in modeling accuracy:

Probability Loss (PL):

$$PL(\underline{\mathbf{u}},\underline{\mathbf{v}})=I(\underline{\mathbf{u}})+I(\underline{\mathbf{v}})-I(\underline{\mathbf{u}}+\underline{\mathbf{v}})\ . \qquad (3)$$

However, if we choose merges with the lowest PL, we will usually merge only vectors with small counts. We are more interested in the average impact of a merge, so we define

Average Probability Loss (APL):

$$APL(\underline{\mathbf{u}},\underline{\mathbf{v}})=(I(\underline{\mathbf{u}})+I(\underline{\mathbf{v}})-I(\underline{\mathbf{u}}+\underline{\mathbf{v}}))/(\sum_i u_i+\sum_i v_i)\ . \quad (4)$$

In our initial experiments, APL worked better than PL. However, APL had a strange side-effect. Most of the phrase clusters it induced made intuitive sense, but there were typically three or four clusters with large numbers of observations on both language sides that grouped together phrases with wildly disparate meanings. Why does APL induce these "monster clusters"?

Consider two count vectors **u** and **v**. If $\Sigma_i u_i$ is very big and $\Sigma_i v_i$ is small, then *I*(**u**) and *I*(**u** + **v**) will be very similar, and APL will be approximately *I*(**v**) /[$\Sigma_i u_i + \Sigma_i v_i$] which will be close to zero. Thus, the decision will probably be made to merge **u** and **v**, even if they have quite different semantics. The resulting cluster, whose counts are represented by **u** + **v**, is now even bigger and even more likely to swallow up other small count vectors in the next rounds of merging: it becomes a kind of black hole.

To deal with this problem, we devised another metric. Let $I(\underline{\mathbf{u}}\,|\,\underline{\mathbf{v}})=u_1\times\log(v_1/\sum_i v_i)+\ldots+u_D\times\log(v_D/\sum_i v_i)$. This is a measure of how well the counts in **v** predict the distribution of counts in **u**. Then let

Maximum Average Probability Loss (MAPL):

$$MAPL(\underline{\mathbf{u}},\underline{\mathbf{v}})=\max(\frac{I(\underline{\mathbf{u}})-I(\underline{\mathbf{u}}\,|\,\underline{\mathbf{u}}+\underline{\mathbf{v}})}{\sum_i u_i},\frac{I(\underline{\mathbf{v}})-I(\underline{\mathbf{v}}\,|\,\underline{\mathbf{u}}+\underline{\mathbf{v}})}{\sum_i v_i})\ .(5)$$

The first term inside the maximum indicates the average probability loss for an observation in **u** when it is modeled by **u**+**v** instead of **u**; similarly, the second term indicates the average probability loss for an observation in **v**. If we merge vector pairs with the lowest values of MAPL, we will never merge vectors in a way that will cause a large loss to either of the two parents.

In practice, we found that all these metrics worked better when multiplied by the Dice coefficient based distance. For **u** and **v**, this is $Dice(\underline{\mathbf{u}},\underline{\mathbf{v}})=1-\frac{2\times|\underline{\mathbf{u}}\cap\underline{\mathbf{v}}|}{|\underline{\mathbf{u}}|+|\underline{\mathbf{v}}|}$, where "|**u**|" means the number of non-zero count entries in **u**, and "$|\underline{\mathbf{u}}\cap\underline{\mathbf{v}}|$" is the number of count entries that are non-zero in **u** and **v.**

3.3 Edit-based similarity

In most of our experiments, count-based metrics were combined with edit-based metrics; we put little effort into optimizing the edit metrics. Let MCWS stand for "maximum common word sequence". For phrases p_1 and p_2, we define

$$Edit(p_1, p_2) = 1 - \frac{2 \times len(MCWS(p_1, p_2))}{len(p_1) + len(p_2)}. \quad (6)$$

where *len*() returns the number of words. This metric doesn't take word identities into account; in future work, we may weight differences involving content words more heavily.

We also defined an edit-based metric for distance between phrase clusters. Let cluster 1 have phrases "red" (10); "burgundy" (5); "resembling scarlet" (2) and cluster 2 have "dark burgundy" (7); "scarlet" (3) (number of observations in brackets). What is the edit distance between clusters 1 and 2? We defined the distance as that between the two phrases with the most observations in each cluster. Thus, distance between clusters 1 and 2 would be Edit("red", "dark burgundy")=1.0. Other definitions are possible.

3.4 Examples of phrase clusters

Figure 2 shows an English phrase cluster learned during C-E experiments by a metric combining count-based and edit-based information. Each phrase is followed by its count in brackets; we don't show phrases with low counts. Since our edit distance sees words as atoms (it doesn't know about morphology), the phrases containing "emancipating" were clustered with phrases containing "emancipation" based on count information, rather than because of the common stem.

Figure 3 shows part of a French phrase cluster learned during F-E experiments by the same mixed metric. The surface forms are quite varied, but most of the phrases mean "to assure or to guarantee that something will happen". An interesting exception is "pas faire" – it means not to do something ("pas" is negative). This illustrates why we need a better edit distance that heavily weights negative words.

emancipating (247), emancipate (167), emancipate our (73), emancipating thinking (67), emancipate our minds (46), further emancipate (45), emancipate the (38), emancipate the mind (38), emancipating minds (33), emancipate their (32), emancipate their minds (27), emancipating our minds (24), emancipating our (21), emancipate our mind (21), further emancipate our (19), emancipate our thinking (14), further emancipate their (11), emancipating the minds (9), emancipate thinking (8), unfettering (8) ...

Figure 2: partial English phrase cluster

garantir que (64), assurer que (46), veiller à ce que (27), afin de garantir (24), faire en sorte (19), de garantir que (16), afin de garantir que (14), faire des (14), de veiller à ce (14), s' assurer que (13), de veiller à ce que (13), pour garantir que (13), de faire en sorte (8), de faire en sorte que (7), à garantir que (6), pas faire (5), de veiller (5)…

Figure 3: partial French phrase cluster

4 Experiments

We carried out experiments on a standard one-pass phrase-based SMT system with a phrase table derived from merged counts of symmetrized IBM2 and HMM alignments; the system has both lexicalized and distance-based distortion components (there is a 7-word distortion limit) and employs cube pruning (Huang and Chiang, 2007). The baseline is a loglinear feature combination that includes language models, the distortion components, relative frequency estimators $P_{RF}(\mathbf{s}|\mathbf{t})$ and $P_{RF}(\mathbf{t}|\mathbf{s})$ and lexical weight estimators $P_{LW}(\mathbf{s}|\mathbf{t})$ and $P_{LW}(\mathbf{t}|\mathbf{s})$. The $P_{LW}()$ components are based on (Zens and Ney, 2004); Foster *et al.* (2006) found this to be the most effective lexical smoothing technique. The phrase-cluster-based components $P_{PC}(\mathbf{s}|\mathbf{t})$ and $P_{PC}(\mathbf{t}|\mathbf{s})$ are incorporated as additional loglinear feature functions. Weights on feature functions are found by lattice MERT (Macherey *et al.*, 2008).

4.1 Data

We evaluated our method on C-E and F-E tasks. For each pair, we carried out experiments on training corpora of different sizes. C-E data were from the NIST[1] 2009 evaluation; all the allowed bilingual corpora except the *UN corpus*, *Hong Kong Hansard and Hong Kong Law corpus* were used to estimate the translation model. For C-E, we trained two 5-gram language models: the first on the English side of the parallel data, and the second on the English *Gigaword corpus*.

Our C-E development set is made up mainly of data from the NIST 2005 test set; it also includes some balanced-genre web-text from the NIST training material. Evaluation was performed on the NIST 2006 and 2008 test sets. **Table 1** gives figures for training, development and test corpora for C-E tasks; |S| is the number of sentences, and |W| is the number of words. There are four references for dev and test sets.

[1] http://www.nist.gov/speech/tests/mt

			Chi	Eng
All parallel Train		|S|	3.3M	
		|W|	68.2M	66.5M
Dev		|S|	1,506	1,506×4
Test	NIST06	|S|	1,664	1,664×4
	NIST08	|S|	1,357	1,357×4
Gigaword		|S|	-	11.7M

Table 1: Statistics for Chinese-to-English tasks.

			Fre	Eng
Train	Europarl	|S|	1.6M	
		|W|	51.3M	46.6M
Dev	2008	|S|	2,051	
Test	2009	|S|	2,525	
	2010	|S|	2,489	
GigaFrEn		|S|	-	22.5M

Table 2: Statistics for French-to-English tasks.

Lang (#sent)		C-E (3.3M)		F-E (1.6M)	
		#count-1	#other	#count-1	#other
Src	Before clustering	11.3M	5.7M	28.1M	21.2M
	After clustering	11.3M	5.3M	28.1M	19.3M
	#clustered	0	0.4M	0	1.9M
Tgt	Before clustering	11.9M	6.0M	25.6M	20.4M
	After clustering	11.9M	5.6M	25.6M	18.5M
	#clustered	0	0.4M	0	1.9M

Table 3: # phrase classes before & after clustering.

For F-E tasks, we used WMT 2010[2] F-E track data sets. Parallel *Europarl* data are used for training; WMT Newstest 2008 set is the dev set, and WMT Newstest 2009 and 2010 are the test sets. One reference is provided for each source input sentence. Two language models are used in this task: one is the English side of the parallel data, and the second is the English side of the *GigaFrEn* corpus. **Table 2** summarizes the training, development and test corpora for F-E tasks.

4.2 Amount of clustering and metric

For both C-E and E-F, we assumed that phrases seen only once in training data couldn't be clustered reliably, so we prevented these "count 1" phrases from participating in clustering. The key clustering parameter is the number of merge operations per iteration, given as a percentage of the number of potential same-language phrase pairs satisfying a simple criterion (some overlap in translations to the other language). Preliminary tests involving the FBIS corpus (about 8% of the C-E data) caused us to set this parameter at 5%. For C-E, we first clustered Chinese with this 5% value, then English with the same amount. For F-E, we first clustered French, then English, using 5% in both cases.

Table 3 shows the results. Only 2-4% of the total phrases in each language end up in a cluster (that's 6.5-9% of eligible phrases, *i.e.*, of phrases that aren't "count 1"). However, about 20-25% of translation probabilities are smoothed for both language pairs. Based on these preliminary tests, we decided to use $Edit \times Dice \times MAPL$ ($Edit \times DMAPL$) as our metric (though $Edit \times Cosine$ was a close runner-up).

4.3 Results and discussion

Our evaluation metric is IBM BLEU (Papineni *et al.*, 2002), which performs case-insensitive matching of n-grams up to $n = 4$. Our first experiment evaluated the effects of the phrase clustering features given various amounts of training data. **Figure 4** gives the BLEU score improvements for the two language pairs, with results for each pair averaged over two test sets (training data size shown as #sentences). The improvement is largest for medium amounts of training data. Since the F-E training data has more words per sentence than C-E, the two peaks would have been closer together if we'd put #words on the x axis: improvements for both tasks peak around 6-8 M English words. For more details, refer to **Table 4** and **Table 5**. The biggest improvement is 0.55 BLEU for the NIST06 test. More importantly, cluster features yield gains in 29 of 30 experiments. Surprisingly, a reviewer asked if we'd done significance tests on the individual results shown in **Tables 4** and **5.** Most likely, many of these individual results are insignificant, but so what? Based on the tables, the probability of the null hypothesis that our method has no effect is equivalent to that of tossing a fair coin 30 times and getting 29 heads (if we adopt an independence approximation).

In the research on paraphrases cited earlier, paraphrases tend to be most helpful for <u>small</u> amounts of training data. By contrast, our approach seems to be most helpful for <u>medium</u> amounts of training data (200-400K sentence

[2] http://www.statmt.org/wmt10/

Data size	Nist06			Nist08		
	Baseline	+phrase-clustering	Improv.	Baseline	+phrase-clustering	Improv.
25K	21.66	21.88	0.22	15.80	15.99	0.19
50K	23.23	23.43	0.20	17.69	17.84	0.15
100K	25.83	26.24	0.41	20.08	20.27	0.19
200K	27.80	28.26	0.46	21.28	21.58	0.30
400K	29.61	30.16	**0.55**	23.37	23.75	**0.38**
800K	30.87	31.17	0.30	24.41	24.65	0.24
1.6M	32.94	33.10	0.16	25.61	25.72	0.11
3.3M	33.59	33.64	0.05	26.84	26.85	0.01

Table 4: BLEU(%) scores for C-E with the various training corpora, including baseline results, results for with phrase clustering, and the absolute improvements. Corpus size is measured in sentences.

Data size	Newstest2009			Newstest2010		
	Baseline	+phrase-clustering	Improv.	Baseline	+phrase-clustering	Improv.
25K	20.21	20.37	0.16	20.54	20.73	0.19
50K	21.25	21.44	0.19	21.95	22.11	0.16
100K	22.56	22.86	0.30	23.44	23.69	0.25
200K	23.67	24.02	**0.35**	24.31	24.71	**0.40**
400K	24.36	24.50	0.14	25.28	25.46	0.18
800K	24.92	24.97	0.05	25.80	25.90	0.10
1.6M	25.47	25.47	0.00	26.35	26.37	0.02

Table 5: BLEU(%) scores for F-E with the various training corpora, including baseline results without phrase clustering feature, results for phrase clustering, and the absolute improvements.

pairs). We attribute this to the properties of count-based clustering. When there is little training data, clustering is unreliable; when there is much data, clustering is reliable but unneeded, because most relative frequencies are well-estimated. In between, phrase cluster probability estimates are both reliable and useful.

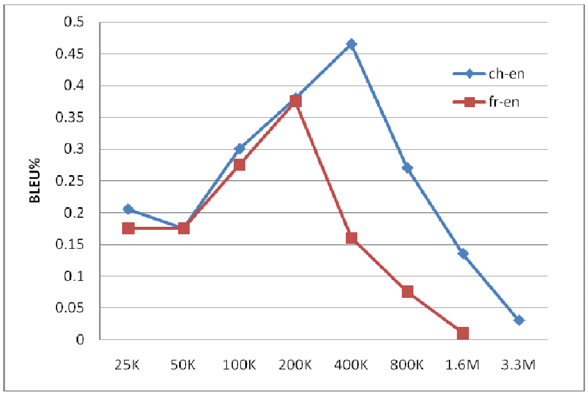


Figure 4: Average BLEU improvement for C-E and F-E tasks (each averaged over two tests) *vs*. #training sent.

Finally, we carried out experiments to see if some of our earlier decisions were correct. Were we right to use DMAPL instead of cosine as the count-based component of our metric? Experiments with $Edit \times DMAPL$ *vs.* $Edit \times Cosine$ on 400K C-E (tested on NIST06 and NIST08) and on 200K F-E (tested on Newstest2009 and 2010) showed a tiny advantage for $Edit \times DMAPL$ of about 0.06 BLEU. So we probably didn't make the wrong decision here (though it doesn't matter much). Were we right to include the *Edit* component? Carrying out analogous experiments with $Edit \times DMAPL$ *vs.* *DMAPL*, we found that dropping *Edit* caused a loss of 0.1-0.2 BLEU for all four test sets. Here again, we made the right decision.

In a final experiment, we allowed "count 1" phrases to participate in clustering (using $Edit \times DMAPL$). The resulting C-E system had somewhat more clustered phrases than the previous one (for both Chinese and English, about 3.5% of phrases were in clusters compared to 2.5% in the previous system). To our surprise, this led to a slight improvement in BLEU: the 400K C-E system now yielded 30.25 on NIST06 (up 0.09) and 23.88 on NIST08 (up 0.13). The F-E system where "count 1" clustering is allowed also had more phrases in clusters than the system where it's prohibited (the former has just under 10% of French and English phrases in clusters *vs.*

4% for the latter). For F-E, the 200K system allowing “count 1” clustering again yielded a slightly higher BLEU: 24.07 on Newstest2009 and 24.76 on Newstest2010 (up 0.05 in both cases). Thus, our decision not to allow “count 1” phrases to participate in clustering in the Table 4 and 5 experiments appears to have been a mistake. We suspect we can greatly improve handling of “count 1” phrases – *e.g.*, by weighting the Edit component of the similarity metric more heavily when assigning these phrases to clusters.

5 Conclusion and Future Work

We have shown that source-language and target-language phrases in the phrase table can be clustered, and that these clusters can be used to smooth “forward” and “backward” estimates P(**t**|**s**) and P(**s**|**t**), yielding modest but consistent BLEU gains over a baseline that included lexical smoothing. Though our experiments were done on a phrase-based system, this method could also be applied to hierarchical phrase-based SMT and syntactic SMT systems. There are several possibilities for future work based on new applications for phrase clusters:

- In the experiments above, we used phrase clusters to smooth P(t|s) and P(s|t) when the pair (s,t) was observed in training data. However, the phrase clusters often give non-zero probabilities for P(t|s) and P(s|t) when s and t were both in the training data, but didn’t co-occur. We could allow the decoder to consider such “invented” phrase pairs (s,t).
- Phrase clusters could be used to construct target language models (LMs) in which the basic unit is a phrase cluster rather than a word. For instance, a tri-cluster model would estimate the probability of phrase p at time i as a function of its phrase cluster, Ci(p), and the two preceding phrase clusters Ci-1 and Ci-2: $P(\mathbf{p}) = f(\mathbf{p} \mid C_i(\mathbf{p})) \times f(C_i \mid C_{i-1}C_{i-2})$.
- Lexicalized distortion models could be modified so as to condition distortion events on phrase clusters.
- We could build SMT grammars in which the terminals are phrases and the parents of terminals are phrase clusters.

The phrase clustering algorithm described above could be improved in several ways:

- In the above, the edit distance between phrases and between phrase clusters was crudely defined. If we improve edit distance, it will have an especially large impact on “count 1” phrases, for which count-based metrics are unreliable and which are a large proportion of all phrases. The edit distance between two phrases weighted all words equally: preferably, weights for word substitution, insertion, or deletion would be learned from purely count-derived phrase clusters (content words and negative words might have heavier weights than other words). The edit distance between two phrase clusters was defined as the edit distance between the phrases with the most observations in each cluster. E.g., distance to the phrase cluster in Figure 2 is defined as the phrase edit distance to “emancipating”. Instead, one could allow a cluster to be characterized by (e.g.) up to three phrases, and let distance between two clusters be the minimum or average pairwise edit distance between these characteristic phrases.
- To cluster phrases, we only used information derived from phrase tables. In future, we could also use the kind of information used in work on paraphrases, such as the context surrounding phrases in monolingual corpora, entries in thesauri, and information from pivot languages.
- The phrase clustering above was “hard”: each phrase in either language belongs to exactly one cluster. We could modify our algorithms to carry out “soft” clustering. For instance, we could interpolate the probabilities associated with a phrase with probabilities from its neighbours.
- Clustering is a primitive way of finding latent structure in the table of joint phrase counts. One could apply principal component analysis or a related algorithm to this table.

References

C. Bannard and C. Callison-Burch. “Paraphrasing with Bilingual Parallel Corpora”. *Proc. ACL*, pp. 597-604, Ann Arbor, USA, June 2005.

C. Callison-Burch, P. Koehn, and M. Osborne. “Improved Statistical Machine Translation Using Paraphrases”. *Proc. HLT/NAACL*, pp. 17-24, New York City, USA, June 2006.

C. Callison-Burch. “Syntactic Constraints on Paraphrases Extracted from Parallel Corpora”. *Proc. EMNLP*, pp. 196-205, Honolulu, USA, October 2008.

D. Chiang. “A hierarchical phrase-based model for statistical machine translation”. *Proc. ACL*, pp. 263-270, Ann Arbor, USA, June 2005.

G. Foster, R. Kuhn, and H. Johnson. “Phrasetable smoothing for statistical machine translation”. *Proc. EMNLP*, pp. 53-61, Sydney, Australia, July 2006.

L. Huang and D. Chiang. “Forest Rescoring: Faster Decoding with Integrated Language Models”. *Proc. ACL*, pp. 144-151, Prague, Czech Republic, June 2007.

P. Koehn. 2010. *Statistical Machine Translation.* Cambridge University Press, Cambridge, UK.

W. Macherey, F. Och, I. Thayer, and J. Uszkoreit. “Lattice-based Minimum Error Rate Training for Statistical Machine Translation”. *Proc. EMNLP*, pp. 725-734, Honolulu, USA, October 2008.

Y. Marton, C. Callison-Burch, and Philip Resnik. “Improved Statistical Machine Translation Using Monolingually-Derived Paraphrases”. *Proc. EMNLP*, pp. 381-390, Singapore, August 2009.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. “Bleu: a method for automatic evaluation of machine translation”. *Proc. ACL*, pp. 311–318, Philadelphia, July 2002.

G. Salton and M. McGill. 1986. *Introduction to Modern Information Retrieval.* McGraw-Hill Inc., New York, USA.

H. Schwenk, M. Costa-jussà, and J. Fonollosa. “Smooth Bilingual N-gram Translation”. *Proc. Joint EMNLP/CoNLL*, pp. 430-438, Prague, Czech Republic, June 2007.

R. Zens and H. Ney. “Improvements in phrase-based statistical machine translation”. *Proc. ACL/HLT*, pp. 257-264, Boston, USA, May 2004.

S. Zhao, H. Wang, T. Liu, and S. Li. “Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora”. *Proc. ACL/HLT*, pp. 780-788, Columbus, USA, June 2008.

Revisiting Context-based Projection Methods for Term-Translation Spotting in Comparable Corpora

Audrey Laroche
OLST – Dép. de linguistique et de traduction
Université de Montréal
audrey.laroche@umontreal.ca

Philippe Langlais
RALI – DIRO
Université de Montréal
felipe@iro.umontreal.ca

Abstract

Context-based projection methods for identifying the translation of terms in comparable corpora has attracted a lot of attention in the community, *e.g.* (Fung, 1998; Rapp, 1999). Surprisingly, none of those works have systematically investigated the impact of the many parameters controlling their approach. The present study aims at doing just this. As a test-case, we address the task of translating terms of the medical domain by exploiting pages mined from Wikipedia. One interesting outcome of this study is that significant gains can be obtained by using an association measure that is rarely used in practice.

1 Introduction

Identifying translations of terms in comparable corpora is a challenge that has attracted many researchers. A popular idea that emerged for solving this problem is based on the assumption that the context of a term and its translation share similarities that can be used to rank translation candidates (Fung, 1998; Rapp, 1999). Many variants of this idea have been implemented.

While a few studies have investigated pattern matching approaches to compare source and target contexts (Fung, 1995; Diab and Finch, 2000; Yu and Tsujii, 2009), most variants make use of a bilingual lexicon in order to translate the words of the context of a term (often called *seed words*). Déjean et al. (2005) instead use a bilingual thesaurus for translating these.

Another distinction between approaches lies in the way the context is defined. The most common practice, the so-called window-based approach, defines the context words as those cooccuring significantly with the source term within windows centered around the term.[1] Some studies have reported gains by considering syntactically motivated co-occurrences. Yu and Tsujii (2009) propose a resource-intensive strategy which requires both source and target dependency parsers, while Otero (2007) investigates a lighter approach where a few hand coded regular expressions based on POS tags simulate source parsing. The latter approach only requires a POS tagger of the source and the target languages as well as a small parallel corpus in order to project the source regular expressions.

Naturally, studies differ in the way each co-occurrence (either window or syntax-based) is weighted, and a plethora of association scores have been investigated and compared, the likelihood score (Dunning, 1993) being among the most popular. Also, different similarity measures have been proposed for ranking target context vectors, among which the popular cosine measure.

The goal of the different authors who investigate context-projection approaches also varies. Some studies are tackling the problem of identifying the translation of general words (Rapp, 1999; Otero, 2007; Yu and Tsujii, 2009) while others are addressing the translation of domain specific terms. Among the latter, many are translating single-word terms (Chiao and Zweigenbaum, 2002; Déjean et al., 2005; Prochasson et

[1] A stoplist is typically used in order to prevent function words from populating the context vectors.

al., 2009), while others are tackling the translation of multi-word terms (Daille and Morin, 2005). The type of discourse might as well be of concern in some of the studies dedicated to bilingual terminology mining. For instance, Morin et al. (2007) distinguish popular science versus scientific terms, while Saralegi et al. (2008) target popular science terms only.

The present discussion only focuses on a few number of representative studies. Still, it is already striking that a direct comparison of them is difficult, if not impossible. Differences in resources being used (in quantities, in domains, etc.), in technical choices made (similarity measures, context vector computation, etc.) and in objectives (general versus terminological dictionary extraction) prevent one from establishing a clear landscape of the various approaches.

Indeed, many studies provide some figures that help to appreciate the influence of some parameters in a given experimental setting. Notably, Otero (2008) studies no less than 7 similarity measures for ranking context vectors while comparing window and syntax-based methods. Morin et al. (2007) consider both the log-likelihood and the mutual information association scores as well as the Jaccard and the cosine similarity measures.

Ideally, a benchmark on which researchers could run their translation finder would ease the comparison of the different approaches. However, designing such a benchmark that would satisfy the evaluation purposes of all the researchers is far too ambitious a goal for this contribution. Instead, we investigate the impact of some major factors influencing projection-based approaches on a task of translating 5,000 terms of the medical domain (the most studied domain), making use of French and English Wikipedia pages extracted monolingually thanks to an information retrieval engine. While the present work does not investigate all the parameters that could potentially impact results, we believe it constitutes the most complete and systematic comparison made so far with variants of the context-based projection approach.

In the remainder of this paper, we describe the projection-based approach to translation spotting in Section 2 and detail the parameters that directly influence its performance. The experimental protocol we followed is described in Section 3 and we analyze our results in Section 4. We discuss the main results in the light of previous work and propose some future avenues in Section 5.

2 Projection-based variants

The approach we investigate for identifying term translations in comparable corpora is similar to (Rapp, 1999) and many others. We describe in the following the different steps it encompasses and the parameters we are considering in the light of typical choices made in the literature.

2.1 Approach

Step 1 A comparable corpus is constructed for each term to translate. In this study, the source and target corpora are sets of Wikipedia pages related to the source term (S) and its reference translation (T) respectively (see Section 3.1). The degree of corpus preprocessing varies greatly from one study to another. Complex linguistic tools such as terminological extractors (Daille and Morin, 2005), parsers (Yu and Tsujii, 2009) or lemmatizers (Rapp, 1999) are sometimes used.

In our case, the only preprocessing that takes place is the deletion of the Wikipedia symbols pertaining to its particular syntax (*e.g.* `[[ ]]`).[2] It is to be noted that, for the sake of simplicity and generality, our implementation does not exploit interlanguage links nor structural elements specific to Wikipedia documents, as opposed to (Yu and Tsujii, 2009).

Step 2 A context vector v_s for the source term S is built (see Figure 1 for a made-up example). This vector contains the words that are in the context of the occurrences of S and are strongly correlated to S. The definition of "context" is one of the parameters whose best value we want to find. Context length can be based on a number of units, for instance 3 sentences (Daille and Morin, 2005), windows of 3 (Rapp, 1999) or 25 words (Prochasson et al., 2009), etc. It is an important parameter of the projection-based approach. Should the context length be too small, we would miss words that would be relevant in finding the translation. On the other hand, if the context is too large, it

[2] We used a set of about 40 regular expressions to do this.

might contain too much noise. At this step, a stoplist made of function words is applied in order to filter out context words and reduce noise in the context vector.

Additionally, an association measure is used to score the strength of correlation between S and the words in its contexts; it serves to normalize corpus frequencies. Words that have a high association score with S are more prominent in the context vector. The association measure is the second important parameter we want to study. As already noted, most authors use the log-likelihood ratio to measure the association between collocates; some, like (Rapp, 1999), informally compare the performance of a small number of association measures, or combine the results obtained with different association measures (Daille and Morin, 2005).

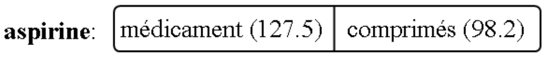


Figure 1: Step 2

Step 3 Words in v_s are projected into the target language with the help of the bilingual seed lexicon (Figure 2). Each word in v_s which is present in the bilingual lexicon is translated, and those translations define the projected context vector v_p. Words that are not found in the bilingual lexicon are simply ignored. The size of the seed lexicon and its content are therefore two important parameters of the approach. In previous studies, seed lexicons vary between 16,000 (Rapp, 1999) and 65,000 (Déjean et al., 2005) entries, a typical size being around 20,000 (Fung, 1998; Chiao and Zweigenbaum, 2002; Daille and Morin, 2005).

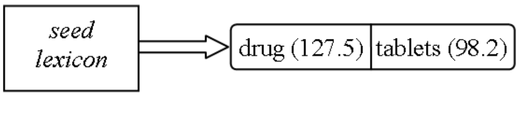


Figure 2: Step 3

Step 4 Context vectors v_t are computed for each candidate term in the target language corpus (Figure 3). The dimension of the target-vector space is defined to be the one induced by the projection mechanism described in Step 3. The context vector v_t of each candidate term is computed as in Step 2. Therefore, in Step 4, the parameters of context definition and association measure are important and take the same values as those in Step 2. Note that in this study, on top of all single terms, we also consider target bigrams as potential candidates (99.5 % of our reference target terms are composed of at most two words). As such, our method can handle complex terms (of up to two words), as opposed to most previous studies, without having to resort to a separate terminological extraction as in (Daille and Morin, 2005).

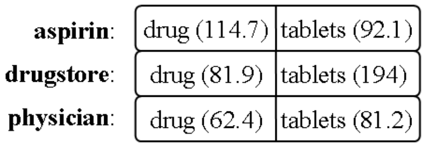


Figure 3: Step 4

Step 5 Context vectors v_t are ranked in decreasing order of their similarity with v_p (Figure 4). The similarity measure between context vectors varies among studies: city-block measure (Rapp, 1999), cosine (Fung, 1998; Chiao and Zweigenbaum, 2002; Daille and Morin, 2005; Prochasson et al., 2009), Dice or Jaccard indexes (Chiao and Zweigenbaum, 2002; Daille and Morin, 2005), etc. It is among the parameters whose effect we experimentally evaluate.

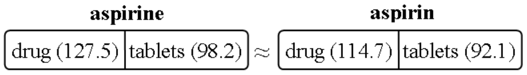


Figure 4: Step 5

2.2 Parameters studied

The five steps we described involve many parameters, the values of which can influence at varying degrees the performance of a translation spotter. In the current study, we considered the following parameter values.

Context We considered contexts defined as the current sentence or the current paragraph involving S. We also considered windows of 5 and 25 words on both sides of S.

Association measure Following the aforementioned studies, we implemented these popular measures: pointwise mutual information (PMI), log-likelihood ratio (LL) and chi-square (χ^2). We also implemented the discounted log-odds (LO) described by (Evert, 2005, p. 86) in his work on collocation mining. To our knowledge, this association measure has not been used yet in translation spotting. It is computed as:

$$\text{odds-ratio}_{disc} = \log \frac{(O_{11} + \frac{1}{2})(O_{22} + \frac{1}{2})}{(O_{12} + \frac{1}{2})(O_{21} + \frac{1}{2})}$$

where O_{ij} are the cells of the 2×2 contingency matrix of a word token s cooccurring with the term S within a given window size.[3]

Similarity measure We implemented four measures: city-block, cosine, as well as Dice and Jaccard indexes (Jurafsky and Martin, 2008, p. 666). Our implementations of Dice and Jaccard are identical to the *DiceMin* and *JaccardMin* similarity measures reported in (Otero, 2008) and which outperformed the other five metrics he tested.

Seed lexicon We investigated the impact of both the size of the lexicon and its content. We started our study with a mixed lexicon of around 5,000 word entries: roughly 2,000 of them belong to the medical domain, while the other entries belong to the general language. We also considered mixed lexicons of 7,000, 9,000 and 11,000 entries (where 2,000 entries are related to the medical domain), as well as a 5,000-entry general language only lexicon.

2.3 Cognate heuristic

Many authors are embedding heuristics in order to improve their approach. For instance, Chiao and Zweigenbaum (2002) propose to integrate a reverse translation spotting strategy in order to improve precision. Prochasson et al. (2009) boost the strength of context words that happen to be transliterated in the other language. A somehow generalized version of this heuristic has been described in (Shao and Ng, 2004).

In this work, we examine the performance of the best configuration of parameters we found, combined with a simple heuristic based on graphic similarity between source and target terms, similar to the orthographic features in (Haghighi et al., 2008)'s generative model. This is very specific to our task where medical terms often (but not always) share Latin or Greek roots, such as *microvillosités* in French and *microvilli* in English.

In this heuristic, translation candidates which are cognates of the source term are ranked first among the list of translation candidates. In our implementation, two words are cognates if their first four characters are identical (Simard et al., 1992). One interesting note concerns the word-order mismatch typically observed in French and English complex terms, such as in *ADN mitochondrial* (French) and *mitochondrial DNA* (English). We do treat this case adequately.

3 Experimental protocol

In order to pinpoint the best configuration of values for the parameters identified in Section 2.2, four series of experiments were carried out. In all of them, the task consists of spotting translation candidates for each source language term using the resources[4] described below. The quality of the results is evaluated with the help of the metrics described in Section 3.2.

3.1 Resources

Corpora The comparable corpora are made of the (at most) 50 French and English Wikipedia documents that are the most relevant to the source term and to its reference translation respectively. These documents are retrieved with the NLGbAse Information Retrieval tool.[5] The average token count of all the 50-document corpora as well as the average frequency of the source and target terms in these corpora for our four series of experiments are listed in Table 1.

[3] For instance, O_{21} stands for the number of windows containing S but not s.

[4] Our resources are available at `http://olst.ling.umontreal.ca/~audrey/coling2010/`. They were acquired as described in (Rubino, 2009).

[5] `http://nlgbase.org/`

	Experiment			
	1	2	3	4
Tokens$_s$	89,431	73,809	42,762	90,328
Tokens$_t$	52,002	27,517	12,891	38,929
$\|S\|$	296	184	66	306
$\|T\|$	542	255	104	404

Table 1: 50-document corpora averages

The corpora are somewhat small (most corpora in previous studies are made of at least a million words). We believe this is more representative of a task where we try to translate domain specific terms. Some of the Wikipedia documents may contain a handful of parallel sentences (Smith et al., 2010), but this information is not used in our approach. The construction of the corpus involves a bias in that the reference translations are used to obtain the most relevant target language documents. However, since our objective is to compare the relative performance of different sets of parameters, this does not affect our results. In fact, as per (Déjean et al., 2005) (whose comparable corpora are English and German abstracts), the use of such an "ideal" corpus is common (as in (Chiao and Zweigenbaum, 2002), where the corpus is built from a specific query).

Seed lexicon The mixed seed lexicon we use is taken from the Heymans Institute of Pharmacology's *Multilingual glossary of technical and popular medical terms*.[6] Random general language entries from the `FreeLang`[7] project are also incorporated into the lexicon for some of our experiments.

Reference translations The test set is composed of 5,000 nominal single and multi-word pairs of French and English terms from the MeSH (*Medical Subject Heading*) thesaurus.[8]

3.2 Evaluation metrics

The performance of each set of parameters in the experiments is evaluated with Top N precision (P_N), recall (R_N) and F-measure (F_N), as well as Mean Average Precision (MAP). Precision is the number of correct translations (at most 1 per source term) divided by the number of terms for which our system gave at least one answer; recall is equal to the ratio of correct translations to the total number of terms. F-measure is the harmonic mean of precision and recall:

$$\text{F-measure} = \frac{2 \times (precision \times recall)}{(precision + recall)}$$

The MAP represents in a single figure the quality of a system according to various recall levels (Manning et al., 2008, p. 147–148):

$$\text{MAP(Q)} = \frac{1}{|Q|} \sum_{|Q|}^{j=1} \frac{1}{m_j} \sum_{m_j}^{k=1} Precision(R_{jk})$$

where $|Q|$ is the number of terms to be translated, m_j is the number of reference translations for the j^{th} term (always 1 in our case), and $Precision(R_{jk})$ is 0 if the reference translation is not found for the j^{th} term or $1/r$ if it is (r is the rank of the reference translation in the translation candidates).

4 Experiments

In Experiment 1, 500 single and multi-word terms must be translated from French to English using each of the 64 possible configurations of these parameters: context definition, association measure and similarity measure. In Experiment 2, we submit to the 8 best variants 1,500 new terms to determine with greater confidence the best 2, which are again tested on the last 3,000 of the test terms (Experiment 3). In Experiment 4, using 1,350 frequent terms, we examine the effects of seed lexicon size and specificity and we apply a heuristic based on cognates.

4.1 Experiment 1

The results of the first series of experiments on 500 terms can be analysed from the point of view of each of the parameters whose values varied among 64 configurations (Section 2.2). The maximal MAP reached for each parametric value is given in Table 2.

The most notable result is that, of the four association measures studied, the log-odds ratio is

[6] `http://users.ugent.be/~rvdstich/eugloss/welcome.html`

[7] `http://www.freelang.net/`

[8] `http://www.nlm.nih.gov/mesh/`

	Param.	Value	Best MAP	In config.
association	LO	**0.536**	sentence_cosine
	LL	0.413	sentence_Dice
	PMI	0.299	sentence_city-block
	χ^2	0.179	sentence_Dice
similarity	cosine	**0.536**	sentence_LO
	Dice	0.520	sentence_LO
	Jaccard	0.520	sentence_LO
	city-block	0.415	sentence_LO
context	sentence	**0.536**	cosine_LO
	paragraph	0.460	cosine_LO
	25 words	0.454	cosine_LO
	5 words	0.361	Dice_LO

Table 2: Best MAP in Experiment 1

significantly superior to the others in every variant. There is as much as 34 % difference between LO and other measures for Top 1 recall. This is interesting since most previous works use the log-likelihood, and none use LO. Our best results for LO (with cosine_sentence) and LL (with Dice_sentence) are in Table 3. Note that the oracle recall is 93 % (7 % of the source and target terms were not in the corpus).

Assoc.	R1	R20	P1	P20	F1	F20	MAP
LO	**39.4**	**84.8**	**42.3**	**91.0**	**40.8**	**87.8**	**0.536**
LL	29.0	75.2	31.3	81.0	30.1	78.0	0.413

Table 3: Best LO and LL configurations scores

Another relevant observation is that the parameters interact with each other. When the similarity measure is cosine, PMI results in higher Top 1 F-scores than LL, but the Top 20 F-scores are better with LL. PMI is better than LL when using city-block as a similarity measure, but LL is better than PMI when using Dice and Jaccard indexes. χ^2 gives off the worst MAP in all but 4 of the 64 parametric configurations.

As for similarity measures, the Dice and Jaccard indexes have identical performances, in accordance with the fact that they are equivalent (Otero, 2008).[9] Influences among parameters are also observable in the performance of similarity measures. When the association measure is LO, the cosine measure gives slightly better Top 1 F-scores, while the Dice index performs slightly better with regards to Top 20 F-scores. Dice is better when the association measure is LL, with a Top 1 F-score gain of about 15 % compared to the cosine.

Again, in the case of context definitions, relative performances depend on the other parameters and on the number of top translation candidates considered. With LO, sentence contexts have the highest Top 1 F-measures, while Top 20 F-measures are highest with paragraphs, and 5-word contexts are the worst.

4.2 Experiment 2

The best parametric values found in Experiment 1 were put to the test on 1,500 different test terms for scale-up verification. Along with LO, which was the best association measure in the previous experiment, we used LL to double-check its relative inefficiency. For all of the 8 configurations evaluated, LL's recall, precision and MAP remain worse than LO's. In particular, LO's MAP scores with the cosine measure are more than twice as high as LL's (respectively 0.33 and 0.124 for sentence contexts). As in Experiment 1, the Dice index is significantly better for LL compared to the cosine, but not for LO. In the case of LO, sentence contexts have better Top 1 performances than paragraphs, and vice versa for Top 20 performances (see Table 4; oracle recall is 93.5 %). Hence, paragraph contexts would be more useful in tasks consisting of proposing candidate translations to lexicographers, while sentences would be more appropriate for automatic bilingual lexicon construction.

Ctx	R1	R20	P1	P20	F1	F20	MAP
Sent.	**23.1**	63.9	**27.8**	76.6	**25.23**	69.68	**0.336**
Parag.	20.1	**70.0**	22.9	**79.7**	21.41	**74.54**	0.325

Table 4: LO_Dice configuration scores

The cosine and Dice similarity measures have similar performances when LO is used. Moreover, we observe the effect of source and target term frequencies in corpus. As seen in Table 1, these frequencies are on average about half smaller in Experiment 2 as they are in Experiment 1, which results in significantly lower performances for all

[9] For this reason, whenever "Dice" is mentioned from this point on, it also applies to the Jaccard index.

8 variants. As Figure 5 shows for the variant LO_cosine_sentence, terms that are more frequent have a greater chance of being correctly translated at better ranks.

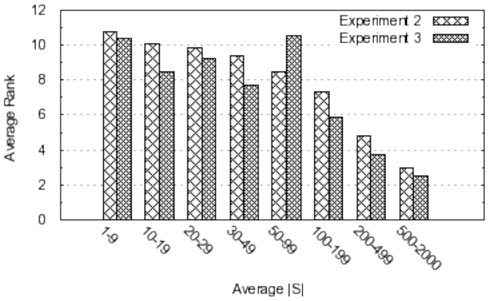


Figure 5: Average rank of correct translation according to average source term frequency

However, the relative performance of the different parametric configurations still holds.

4.3 Experiment 3

In Experiment 3, we evaluate the two best configurations from Experiment 2 with 3,000 new terms in order to verify the relative performance of the cosine and Dice similarity measures. As Table 5 shows, cosine has slightly better Top 1 figures, while Dice is a little better when considering the Top 20 translation candidates. Therefore, as previously mentioned, the choice of similarity measure (cosine or Dice) should depend on the goal of translation spotting. Note that the scores in Experiment 3 are much lower than those of Experiments 1 and 2 because of low term frequencies in the corpus (see Table 1 and Figure 5). Also, oracle recall is only 71.1 %.

Sim.	R1	R20	P1	P20	F1	F20	MAP
Cosine	**9.8**	28.1	**20.7**	59.4	**13.3**	38.15	0.232
Dice	9.4	**28.9**	19.8	**61.2**	12.75	**39.26**	**0.286**

Table 5: LO_sentence configuration scores

4.4 Experiment 4

In the last series of experiments, we examine the influence of the bilingual seed lexicon specificity and size, using the 1,350 terms which have source and target frequencies $\geq$ 30 from the 1,500 and 3,000 sets used in Experiments 2 and 3 (oracle recall: 100 %). We tested the different lexicons (see Section 2.2) on the 4 parametric configurations made of sentence contexts, LO or LL association measures, and cosine or Dice similarity measures.

Yet again, LO is better than LL. MAP scores for LO in all variants are comprised in [0.466–0.489]; LL MAPs vary between 0.135 and 0.146 when the cosine is used and between 0.348 and 0.380 when the Dice index is used.

According to our results, translation spotting is more accurate when the seed lexicon contains (5,000) entries from both the medical domain and general language instead of general language words only, but only by a very small margin. Table 6 shows the results for the configuration LO_cosine_sentence. The fact that the difference is so small could be explained by our resources' properties. The reference translations from MeSH contain terms that are also used in other domains or in the general language, *e.g.* terms from the category "people" (Névéol and Ozdowska, 2006). Wikipedia documents retrieved by using those references may in turn not belong to the medical domain, in which case medical terms from the seed lexicon are not appropriate. Still, the relatively good performance of the general language-only lexicon supports (Déjean et al., 2005, p. 119)'s claim that general language words are useful when spotting translations of domain specific terms, since the latter can appear in generic contexts.

Lex.	R1	R20	P1	P20	F1	F20	MAP
Gen. + med.	**39.3**	87.0	**39.6**	87.6	**39.4**	87.3	**0.473**
Gen. only	38.8	**88.1**	39.0	**88.5**	38.9	**88.3**	0.471

Table 6: LO_cosine_sentence configuration scores

Lexicon sizes tested are 5,000 (the mixed lexicon used in previous experiments), 7,000, 9,000 and 11,000 entries. The performance (based on MAP) is better when 7,000- and 9,000-entry lexicons are used, because more source language context words can be taken into account. However, when the lexicon reaches 11,000, Top 1 MAP scores and F-measures are slightly lower than those obtained with the 7,000-entry one. This may happen because the lexicon is increased with general language words; 9,000 of the 11,000 entries are not from the medical domain, making it harder for the context words to be specific. It would be interesting to study the specificity of context vectors built from the source corpus. Still, the differences in scores are small, as Table 7 shows (see Table 6 for the results obtained with 5,000 entries). This is because, in our implementation, context vector size is limited to 20, as in (Daille and Morin, 2005), in order to reduce processing time. The influence of context vector sizes should be studied.

Lex. size	R1	R20	P1	P20	F1	F20	MAP
7,000	**41.5**	88.8	**41.6**	89.1	**41.5**	88.9	0.488
9,000	40.9	89.3	41.1	89.7	41.0	89.5	**0.489**
11,000	40.1	**89.8**	40.2	**90.1**	40.1	**89.9**	0.484

Table 7: LO_cosine_sentence configuration scores

The parameters related to the seed lexicon do not have as great an impact on the performance as the choice of association measure does: the biggest difference in F-measures for Experiment 4 is 2.9 %. At this point, linguistic-based heuristics such as graphic similarity should be used to significantly increase performance. We applied the cognate heuristic (Section 2.3) on the Top 20 translation candidates given by the variant LO_sentence_9,000-entry lexicon using cosine and Dice similarity measures. Without the heuristic, Top 1 performances are better with cosine, while Dice is better for Top 20. Applying the cognate heuristic makes the Top 1 precision go from 41.1 % to 55.2 % in the case of cosine, and from 39.6 % to 53.9 % in the case of Dice.

5 Discussion

Our results show that using the log-odds ratio as the association measure allows for significantly better translation spotting than the log-likelihood. A closer look at the translation candidates obtained when using LL, the most popular association measure in projection-based approaches, shows that they are often collocates of the reference translation. Therefore, LL may fare better in an indirect approach, like the one in (Daille and Morin, 2005).

Moreover, we have seen that the cosine similarity measure and sentence contexts give more correct top translation candidates, at least when LO is used. Indeed, the values of the different parameters influence one another in most cases. Parameters related to the seed lexicon (size, domain specificity) are not of great influence on the performance, but this may in part be due to our resources and the way they were built.

The highest Top 1 precision, 55.2 %, was reached with the following parameters: sentence contexts, LO, cosine and a 9,000-entry mixed lexicon, with the use of a cognate heuristic.

In future works, other parameters which influence the performance will be studied, among which the use of a terminological extractor to treat complex terms (Daille and Morin, 2005), more contextual window configurations, and the use of syntactic information in combination with lexical information (Yu and Tsujii, 2009). It would also be interesting to compare the projection-based approaches to (Haghighi et al., 2008)'s generative model for bilingual lexicon acquisition from monolingual corpora.

One latent outcome of this work is that Wikipedia is surprisingly suitable for mining medical terms. We plan to check its adequacy for other domains and verify that LO remains a better association measure for different corpora and domains.

Acknowledgments

We are deeply grateful to Raphaël Rubino who provided us with the data material we have been using in this study. We thank the anonymous reviewers for their suggestions.

References

Chiao, Yun-Chuang and Pierre Zweigenbaum. 2002. Looking for candidate translational equivalents in specialized, comparable corpora. In *19th International Conference on Computational Linguistics*, pages 1208–1212.

Daille, Béatrice and Emmanuel Morin. 2005. French-English terminology extraction from comparable corpora. In *2nd International Joint Conference on Natural Language Processing*, pages 707–718.

Déjean, Hervé, Éric Gaussier, Jean-Michel Renders, and Fatiha Sadat. 2005. Automatic processing of multilingual medical terminology: Applications to thesaurus enrichment and cross-language information retrieval. *Artificial Intelligence in Medicine*, 33(2):111–124. Elsevier Science, New York.

Diab, Mona and Steve Finch. 2000. A statistical word-level translation model for comparable corpora. In *Proceedings of the Conference on Content-Based Multimedia Information Access*.

Dunning, Ted. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

Evert, Stefan. 2005. *The Statistics of Word Cooccurrences. Word Pairs and Collocations*. Ph.D. thesis, Universität Stuttgart.

Fung, Pascale. 1995. A pattern matching method for finding noun and proper noun translations from noisy parallel corpora. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 236–243.

Fung, Pascale. 1998. A statistical view on bilingual lexicon extraction: From parallel corpora to non-parallel corpora. In *3rd Conference of the Association for Machine Translation in the Americas*, pages 1–17.

Haghighi, Aria, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Human Language Technology and Association for Computational Linguistics*, pages 771–779.

Jurafsky, Daniel and James H. Martin. 2008. *Speech and Language Processing*. Prentice-Hall.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Morin, Emmanuel, Béatrice Daille, Koichi Takeuchi, and Kyo Kageura. 2007. Bilingual terminology mining — using brain, not brawn comparable corpora. In *45th Annual Meeting of the Association for Computational Linguistics*, pages 664–671.

Névéol, Aurélie and Sylwia Ozdowska. 2006. Terminologie médicale bilingue anglais/français: usages cliniques et bilingues. *Glottopol*, 8.

Otero, Pablo Gamallo. 2007. Learning bilingual lexicons from comparable English and Spanish corpora. In *Machine Translation Summit 2007*, pages 191–198.

Otero, Pablo Gamallo. 2008. Evaluating two different methods for the task of extracting bilingual lexicons from comparable corpora. In *1st Workshop Building and Using Comparable Corpora*.

Prochasson, Emmanuel, Emmanuel Morin, and Kyo Kageura. 2009. Anchor points for bilingual lexicon extraction from small comparable corpora. In *Machine Translation Summit XII*, pages 284–291.

Rapp, Reinhard. 1999. Automatic identification of word translations from unrelated English and German corpora. In *37th Annual Meeting of the Association for Computational Linguistics*, pages 66–70.

Rubino, Raphaël. 2009. Exploring context variation and lexicon coverage in projection-based approach for term translation. In *Proceedings of the Student Research Workshop associated with RANLP–09*, pages 66–70.

Saralegi, X., I. San Vicente, and A. Gurrutxaga. 2008. Automatic extraction of bilingual terms from comparable corpora in a popular science domain. In *1st Workshop Building and Using Comparable Corpora*.

Shao, Li and Hwee Tou Ng. 2004. Mining new word translations from comparable corpora. In *20th International Conference on Computational Linguistics*, pages 618–624.

Simard, Michel, George Foster, and Pierre Isabelle. 1992. Using cognates to align sentences in bilingual corpora. In *4th Conference on Theoretical and Methodological Issues in Machine Translation*, pages 67–81.

Smith, Jason R., Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*, pages 403–411.

Yu, Kun and Junichi Tsujii. 2009. Bilingual dictionary extraction from Wikipedia. In *Machine Translation Summit XII*.

Constituent Reordering and Syntax Models for English-to-Japanese Statistical Machine Translation

Young-Suk Lee
IBM Research
ysuklee@us.ibm.com

Bing Zhao
IBM Research
zhaob@us.ibm.com

Xiaoqiang Luo
IBM Research
xiaoluo@us.ibm.com

Abstract

We present a constituent parsing-based reordering technique that improves the performance of the state-of-the-art English-to-Japanese phrase translation system that includes distortion models by 4.76 BLEU points. The phrase translation model with reordering applied at the pre-processing stage outperforms a syntax-based translation system that incorporates a phrase translation model, a hierarchical phrase-based translation model and a tree-to-string grammar. We also show that combining constituent reordering and the syntax model improves the translation quality by additional 0.84 BLEU points.

1 Introduction

Since the seminal work by (Wu, 1997) and (Yamada and Knight, 2001), there have been great advances in syntax-based statistical machine translation to accurately model the word order distortion between the source and the target languages.

Compared with the IBM source-channel models (Brown et al., 1994) and the phrase translation models (Koehn et al., 2003), (Och and Ney, 2004) which are good at capturing local reordering within empirical phrases, syntax-based models have been effective in capturing the long-range reordering between language pairs with very different word orders like Japanese-English (Yamada and Knight, 2001), Chinese-English (Chiang, 2005) and Urdu-English (Zollmann et al. 2008), (Callison-Burch et al. 2010).

However, (Xu et al., 2009) show that applying dependency parsing-based reordering as pre-processing (pre-ordering hereafter) to phrase translation models produces translation qualities significantly better than a hierarchical phrase-based translation model (Hiero hereafter) implemented in (Zollman and Venugopal, 2006) for English-to-Japanese translation. They also report that the two models result in comparable translation qualities for English-to-Korean/Hindi/Turkish/Urdu, underpinning the limitations of syntax-based models for handling long-range reordering exhibited by the strictly head-final Subject-Object-Verb (SOV) order languages like Japanese and the largely head-initial Subject-Verb-Object (SVO) order languages like English.

In this paper, we present a novel constituent parsing-based reordering technique that uses manually written context free (CFG hereafter) and context sensitive grammar (CSG hereafter) rules. The technique improves the performance of the state-of-the-art English-to-Japanese phrase translation system that includes distortion models by 4.76 BLEU points. The phrase translation model with constituent pre-ordering consistently outperforms a syntax-based translation system that integrates features from a phrase translation model, Hiero and a tree-to-string grammar. We also achieve an additional 0.84 BLEU point improvement by applying an extended set of reordering rules that incorporate new rules learned from the syntax model for decoding.

The rest of the paper is organized as follows. In Section 2, we summarize previous work related to this paper. In Section 3, we give an overview of the syntax model with which we compare the performance of a phrase translation model with pre-ordering. We also discuss a chart-based decoder used in all of our experiments. In Section 4, we describe the constituent parsing-based reordering rules. We show the impact of pre-ordering on a phrase translation model and compare its performance with the syntax model. In Section 5, we discuss experimental results from the combination of syntax model and pre-ordering. Finally in Section 6, we discuss future work.

2 Related Work

Along the traditions of unsupervised learning by (Wu, 1997), (Chiang, 2005) presents a model that uses hierarchical phrases, Hiero. The model is a synchronous context free grammar learned from a parallel corpus without any linguistic annotations and is applied to Chinese-to-English translation. (Galley and Manning, 2008) propose a hierarchical phrase reordering model that uses shift-reduce parsing.

In line with the syntax-based model of (Yamada and Knight, 2001) that transforms a source language parse tree into a target language string for Japanese-English translation, linguistically motivated syntactic features have been directly incorporated into both modeling and decoding. (Liu, et. al. 2006), (Zhao and Al-Onaizan, 2008) propose a source tree to target string grammar (tree-to-string grammar hereafter) in order to utilize the source language parsing information for translation. (Liu, et. al. 2007) propose packed forest to allow ambiguities in the source structure for the tree-to-string grammar. (Ding and Palmer, 2005) and (Zhang et. al., 2006) propose a tree-to-tree grammar, which generates the target tree structure from the high-precision source syntax. (Shen, et. al., 2008) propose a string to dependency tree grammar to use the target syntax when the target is English for which parsing is more accurate than other languages. (Marcu et al., 2006) introduce a syntax model that uses syntactified target language phrases. (Chang and Toutanova, 2007) propose a global discriminative statistical word order model that combines syntactic and surface movement information, which improves the translation quality by 2.4 BLEU points in English-to-Japanese translation. (Zollmann, et. al., 2008) compare various translation models and report that the syntax augmented model works better for Chinese-to-English and Urdu-to-English, but not for Arabic-to-English translation. (Carreras and Collins, 2009) propose a highly flexible reordering operations during tree adjoining grammar parsing for German-English translation. (Callison-Burch et al., 2010) report a dramatic impact of syntactic translation models on Urdu-to-English translation.

Besides the approaches which integrate the syntactic features into translation models, there are approaches showing improvements via pre-ordering for model training and decoding. (Xia and McCord, 2004), (Collins et al., 2005) and (Wang, et. al. 2007) apply pre-ordering to the training data according to language-pair specific reordering rules to improve the translation qualities of French-English, German-English and Chinese-English, respectively. (Habash, 2007) uses syntactic preprocessing for Arabic-to-English translation. (Xu et al., 2009) use a dependency parsing-based pre-ordering to improve translation qualities of English to five SOV languages including Japanese.

The current work is related to (Xu et al., 2009) in terms of the language pair and translation models explored. However, we use constituent parsing with hierarchical rules, while (Xu et al., 2009) use dependency parsing with precedence rules. The two approaches have different rule coverage and result in different word orders especially for phrases headed by verbs and prepositions. We also present techniques for combining the syntax model with tree-to-string grammar and pre-ordering for additional performance improvement. The total improvement by the current techniques over the state-of-the-art phrase translation model is 5.6 BLEU points, which is an improvement gap not attested elsewhere with reordering approaches.

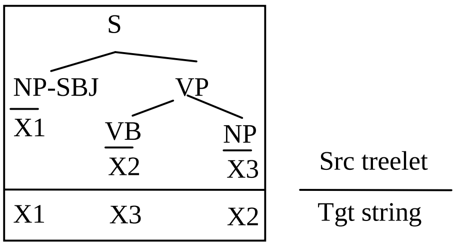
3 Syntax Model and Chart-Based Decoder

In this section, we give an overview of the syntax model incorporating a tree-to-string grammar. We will compare the syntax model performance with a phrase translation model that uses the pre-ordering technique we propose in Section 4. We also describe the chart-based decoder that we use in all of the experiments reported in this paper.

3.1 Tree-to-String Grammar

Tree-to-string grammar utilizes the source language parse to model reordering probabilities from a source tree to the target string (Liu et. al., 2006), (Liu et. al., 2007), (Zhao and Al-Onaizan, 2008) so that long distance word reordering becomes local in the parse tree.

Reordering patterns of the source language syntax and their probabilities are automatically learned from the word-aligned source-parsed parallel data and incorporated as a tree-to-string grammar for decoding. Source side parsing and the resulting reordering patterns bound the search space. Parsing also assigns linguistic labels to the chunk, e.g. NP-SBJ, and allows statistics to be clustered reasonably. Each synchronous context free grammar (SCFG) rewriting rule rewrites a source treelet into a target string, with both sides containing hiero-style variables. For instance, the rule [X, VP] [X, VB] [X,NP] → [X, NP] [X, VB] rewrites a VP with two constituents VB and NP into an NP VB order in the target, shown below.



The tree-to-string grammar introduces possible search space to generate an accurate word order, which is refined on the basis of supports from other models. However, if the correct word order cannot be generated by the tree-to-string grammar, the system can resort to rules from Hiero or a phrase translation model for extended rule coverage.

3.2 Chart-based Decoder

We use a chart-based decoder − a template decoder that generalizes over various decoding schemes in terms of the dot-product in Earley-style parsing (Earley, 1970) − to support various decoding schemes such as phrase, Hiero (Chiang, 2005), Tree-to-String, and the mixture of all of the above.

This framework allows one to strictly compare different decoding schemes using the same feature and parameter setups. For the experimental results in Sections 4 & 5, we applied (1) phrase decoding for the baseline phrase translation system that includes distortion models, (2) Hiero decoding for the Hiero system that incorporates a phrase translation model, and (3) Tree-to-string decoding for the syntax-based systems that incorporate features from phrase translation, Hiero and tree-to-string grammar models.

The decoder seeks the best hypothesis e^* according to the Bayesian decision rule (1):

$$e^* = \underset{\{e,d \in D\}}{\arg\min}\, \phi(e) \cdot \phi(d) \qquad (1)$$

d is one derivation path, rewriting the source tree into the target string via the probabilistic synchronous context free tree-to-string grammar (PSCFG). $\phi(e)$ is the cost functions computed from general n-gram language models. In this work, we use two sets of interpolated 5-gram language models. $\phi(d)$ is a vector of cost functions defined on the derivation sequence. We have integrated 18 cost functions ranging from the basic relative frequencies and IBM model-1 scores to counters for different types of rules including blocks, glue, Hiero, and tree-to-string grammar rules. Additional cost functions are also integrated into the decoder, including measuring the function/content-word mismatch between source and target, similar to (Chiang et. al., 2009) and length distribution for non-terminals in (Shen et. al., 2009).

4 Parsing and Reordering Rules

We apply a set of manually acquired reordering rules to the parsing output from a constituent parser to pre-order the data for model training and decoding.

4.1 Parsing with Functional Tags

We use a maximum entropy English parser (Ratnaparkhi, 1999) trained on OntoNotes (Hovy, 2006) data. OntoNotes data include most of the Wall Street Journal data in Penn Treebank (Marcus et al., 1993) and additional data from broadcast conversation, broadcast news and web log.

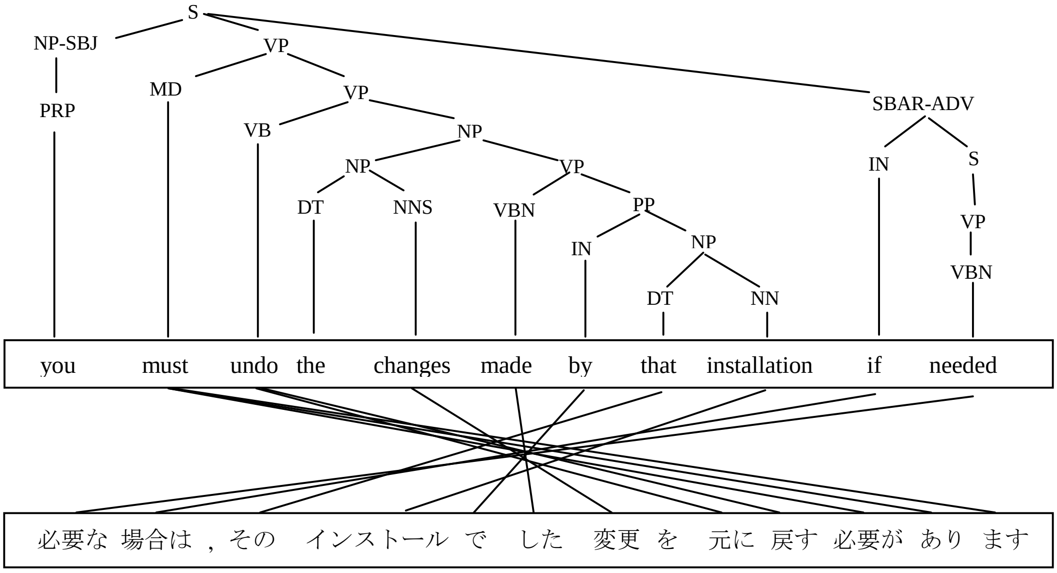

S
NP-SBJ
VP
PRP
MD
VP
VB
NP
NP
VP
DT
NNS
VBN
PP
IN
NP
DT
NN
SBAR-ADV
IN
S
VP
VBN
you must undo the changes made by that installation if needed
必要な 場合は , その インストール で した 変更 を 元に 戻す 必要が あり ます

Figure 1. Parse Tree and Word Alignment before Reordering

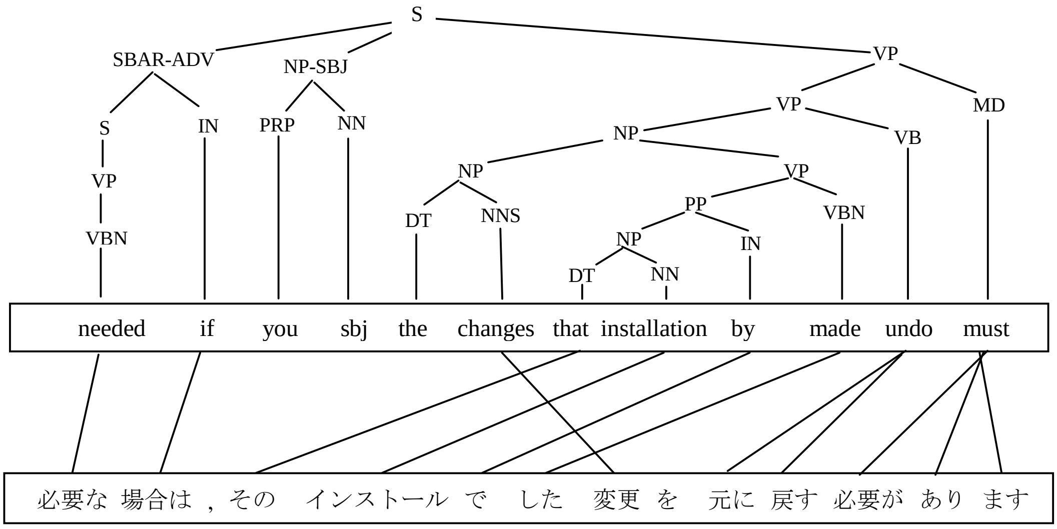

S
SBAR-ADV
NP-SBJ
VP
S
IN
PRP
NN
VP
MD
VP
NP
VB
VBN
NP
VP
DT
NNS
PP
VBN
NP
IN
DT
NN
needed if you sbj the changes that installation by made undo must
必要な 場合は , その インストール で した 変更 を 元に 戻す 必要が あり ます

Figure 2. Parse Tree and Word Alignment after Reordering

The parser is trained with all of the functional and part-of-speech (POS) tags in the original distribution: total 59 POS tags and 364 phrase labels.

We use functional tags since reordering decisions for machine translation are highly influenced by the function of a phrase, as will be shown later in this section. An example parse tree with functional tags is given at the top half of Figure 1. NP-SBJ indicates a subject noun phrase, SBAR-ADV, an adverbial clause.

4.2 Structural Divergence between English and Japanese

Japanese is a strictly head-final language, i.e. the head is located at the end of a phrase. This leads to a high degree of distortions with English, which is largely head initial.

The word order contrast between the two languages is illustrated by the human word alignment at the bottom half of Figure 1. All instances of word alignments are non-monotonic except for the sequence *that installation*, which is monotonically aligned to the Japanese morpheme sequence その インストール. Note that there are no word boundaries in Japanese written text, and we apply Japanese morpheme segmentation to obtain morpheme sequences in the figure. All of the Japanese examples in this paper are presented with morpheme segmentation.

The manual reordering rules are written by a person who is proficient with English and Japanese/Korean grammars, mostly on the basis of perusing parsed English texts.

4.3 CFG Reordering Rules

Our reordering rules are mostly CFG rules and divided into head and modifier rules.

Head reordering rules in Table 1 move verbs and prepositions from the phrase initial to the phrase final positions (Rules 1-11). Reordering of the head phrase in an adverbial clause also belongs to this group (Rules 12-14). The label sequences in **Before RO** and **After RO** are the immediate children of the **Parent Node** before and after reordering. VBX stands for VB, VBZ, VBP, VBD, VBN and VBG. XP^+ stands for one or more POS and/or phrase labels such as MD, VBX, NP, PP, VP, etc. In 2 & 4, RB is the tag for negation *not*. In 5, RP is the tag for a verb particle.

Modifier reordering rules in Table 2 move modified phrases from the phrase initial to the phrase final positions within an NP (Rules 1-3). They also include placement of NP, PP, ADVP within a VP (Rules 4 & 5). The subscripts in a rule, e.g. PP_1 and PP_2 in Rule 3, indicate the distinctness of each phrase sharing the same label.

4.4 CSG Reordering Rules

Some reordering rules cannot be captured by CFG rules, and we resort to CSG rules.[1]

	Parent Node	**Before RO**	**After RO**
1	VP	MD VP	VP MD
2	VP	MD RB VP	VP MD RB
3	VP	VBX XP^+	XP^+ VBX
4	VP	VBX RB XP^+	XP^+ VBX RB
5	VP	VBX RP XP^+	XP^+ VBX RP
6	ADJP-PRD	JJ XP^+	XP^+ JJ
7	PP	IN NP	NP IN
8	PP	IN S	S IN
9	SBAR-TMP	IN S	S IN
10	SBAR-ADV	IN S	S IN
11	SBAR-PRP	IN S	S IN
12	SBAR-TMP	WHADVP S	S WHADVP
13	SBAR-ADV	WHADVP S	S WHADVP
14	SBAR-PRP	WHADVP S	S WHADVP

Table 1. Head Reordering Rules

	Parent Node	**Before RO**	**After RO**
1	NP	NP SBAR	SBAR NP
2	NP	NP PP	PP NP
3	NP	NP PP_1 PP_2	PP_1 PP_2 NP
4	VP	VBX NP PP	PP NP VBX
5	VP	VBX NP ADVP-TMP PP	PP NP ADVP-TMP VBX

Table 2. Modifier Reordering Rules

For instance, in the parse tree and word alignment in Figure 1, the last two English words *if needed* under SBAR-ADV is aligned to the first two Japanese words 必要な 場合は. In order to change the English order to the corresponding Japanese order, SBAR-ADV dominated by the VP should move across the VP to sentence initial position, as shown in the top half of Figure 2, requiring a CSG rule.

The adverbial clause reordering in Figure 2 is denoted as Rule 1 in Table 3, which lists two other CSG rules, Rule 2 & 3.[2] The subscripts in Table 3 are interpreted in the same way as those in Table 2.

[1] These CSG rules apply to trees of depth two or more, and the applications are dependent on surrounding contexts. Therefore, they are different from CFG rules which apply only to trees of depth one, and TSG (tree substitution grammar) rules for which variables are independently substituted by substitution. The readers are referred to (Joshi and Schabes, 1997) for formal definitions of various grammar formalisms.

[2] Rule 3 is applied after all CFG rules, see Section 4.6. Therefore, VBX's are located at the end of each corresponding VP.

	Before RO → After RO
1	(S XP_1^+ (VP XP_2^+ SBAR-ADV)) → (S SBAR-ADV XP_1^+ (VP XP_2^+))
2	(S XP_1^+ (VP (XP_2^+ SBAR-ADV))) → (S XP_1^+ SBAR-ADV (VP (XP_2^+)))
3	(VP_1 ADVP-MNR (VP_2 XP^+ VBX_2) VBX_1) →(VP_1 (VP_2 XP^+ ADVP-MNR VBX_2) VBX_1)

Table 3. CSG Reordering Rules

ADVP-MNR stands for a manner adverbial phrase such as *explicitly* in the following: *The software version has been explicitly verified as working*. Rule 3 in Table 3 indicates that a ADVP-MNR has to immediately precede a verb in Japanese, resulting in the substring '*...as working explicitly verified...*' after reordering.

Note that functional tags allow us to write reordering rules specific to semantic phrases. For instance, in Rule 1, SBAR-ADV under VP moves to the sentence initial position under S, but an SBAR without any functional tags do not. It typically stays within a VP as the complement of the verb.

4.5 Subject Marker Insertion

Japanese extensively uses case particles that denote the role of the preceding noun phrase, for example, as subject, object, etc. We insert *sbj*, denoting the subject marker, at the end of a subject noun phrase NP-SBJ. The inserted subject marker *sbj* mostly gets translated into the subject particle が or は in Japanese.[3]

4.6 Reordering Rule Application

The rules are applied categorically, sequentially and recursively. CSG Rules 1 and 2 in Table 3 are applied before all of the CFG rules. Among CFG rules, the modifier rules in Table 2 are applied before the head rules in Table 1. CSG Rule 3 in Table 3 is applied last, followed by the subject marker insertion operation.

CFG head and modifier rules are applied recursively. The top half of Figure 2 is the parse tree obtained by applying reordering rules to the parse tree in Figure 1. After reordering, the word alignment becomes mostly monotonic, as shown at the bottom half of Figure 2.

4.7 Experimental Results

All systems are trained on a parallel corpus, primarily from the Information Technology (IT) domain and evaluated on the data from the same domain. The training data statistics is in Table 4 and the evaluation data statistics is in Table 5. Japanese tokens are morphemes and English tokens are punctuation tokenized words.

Corpus Stats	English	Japanese
sentence count	3,358,635	3,358,635
token count	57,231,649	68,725,865
vocabulary size	242,712	348,221

Table 4. Training Corpus Statistics

Data Sets	Sentence Count	Token Count
Tuning	600	11,761
DevTest	437	8,158
Eval	600	11,463

Table 5. Evaluation Data Statistics

We measure the translation quality with IBM BLEU (Papineni et al., 2002) up to 4 grams, using 2 reference translations, BLEUr2n4. For BLEU score computation, we character-segment Kanji and Kana sequences in the reference and the machine translation output. Various system performances are shown in Table 6.

Models	Tuning	DevTest	Eval
Phrase (BL)	0.5102	0.5330	0.5486
Hiero	0.5385	0.5574	0.5724
Syntax	0.5561	0.5777	0.5863
Phrase+RO1	0.5681	0.5793	0.5962

Table 6. Model Performances (BLEUr2n4)

Phrase (BL) is the baseline phrase translation system that incorporates lexical distortion models (Al-Onaizan and Papineni, 2006). Hiero is the hierarchical phrase-based system (Chiang, 2006) that incorporates the phrase translation model. Syntax is the syntax model described in Section 3, which incorporates the phrase translation, Hiero and tree-to-string grammar models. Phrase+RO1 is the phrase translation model with pre-ordering for system training and decoding, using the rules described in this section. Phrase+RO1 improves the translation quality of the baseline model by 4.76 BLEU points and outperforms the syntax model by over 0.9 BLEU points.

[3] The subject marker insertion is analogous to the insertion operation in (Yamada and Knight, 2001), which covers a wide range of insertion of case particles and verb inflections in general.

5 Constituent Reordering and Syntax Model Combined

Translation qualities of systems that combine the syntax model and pre-ordering are shown in Table 7. Syntax+RO1 indicates the syntax model with pre-ordering discussed in Section 4. Syntax+RO2 indicates the syntax model with a more extensive pre-ordering for decoding discussed below .

Models	Tuning	DevTest	Eval
Phrase+RO1	0.5681	0.5793	0.5962
Syntax+RO1	0.5742	0.5802	0.6003
Syntax+RO2	0.5769	0.5880	0.6046

Table 7. Syntax Model with Pre-ordering

Analyses of the syntax model in Table 6 revealed that automatically learned rules by the tree-to-string grammar include new rules not covered by the manually written rules, some of which are shown in Table 8.

Parent Node	**Before RO**	**After RO**
ADJP-PRD	RB JJ PP	PP RB JJ
ADVP-TMP	RB PP	PP RB
ADVP	ADVP PP	PP ADVP
NP	NP VP	VP NP

Table 8. New CFG rules automatically learned by Tree-to-String grammar

We augment the manual rules with the new automatically learned rules. We call this extended set of reordering rules RO2. We use the manual reordering rules RO1 for model training, but use the extended rules RO2 for decoding. And we obtain the translation output Syntax+RO2 in Table 7. Syntax+RO2 outperforms Phrase+RO1 by 0.84 BLEU points, and Syntax+RO1 by 0.43 BLEU points.

In Table 9, we show the ratio of each rule type preserved in the derivation of one-best translation output of the following two models: Syntax and Syntax+RO2. In the table, 'Blocks' indicate phrases from the phrase translation model and 'Glue Rules' denote the default grammar rule for monotone decoding.

The syntax model without pre-ordering (Syntax) heavily utilizes the Hiero and tree-to-string grammar rules, whereas the syntax model with pre-ordering (Syntax+RO2) mostly depends on monotone decoding with blocks and glue rules.

Rule Type	Syntax	Syntax+RO2
Blocks	46.3%	51.2%
Glue Rules	6.0%	37.3%
Hiero Rules	18.3%	1.3%
Tree-to-String	29.4%	10.2%

Table 9. Ratio of each rule type preserved in the translation derivation of Syntax and Syntax+RO2

6 Summary and Future Research

We have proposed a constituent pre-ordering technique for English-to-Japanese translation. The technique improves the performance of the state-of-the-art phrase translation models by 4.76 BLEU points and outperforms a syntax-based translation system that incorporates a phrase translation model, Hiero and a tree-to-string grammar. We have also shown that combining constituent pre-ordering and the syntax model improves the translation quality by additional 0.84 BLEU points.

While achieving solid performance improvement over the existing translation models for English-to-Japanese translation, our work has revealed some limitations of syntax models both in terms of grammar representations and modeling. Whereas many syntax models are based on CFG rules for probability acquisition, the current research shows that there are various types of reordering that require the generative capacity beyond CFG. While most of the reordering rules for changing the English order to the Japanese order (and vice versa) should apply categorically,[4] often the probabilities of tree-to-string grammar rules are not high enough to survive in the translation derivations.

As for the reordering rules that require the generative capacity beyond CFG, we may model mildly context sensitive grammars such as tree adjoining grammars (Joshi and Schabes, 1997), as in (Carreras and Collins, 2009). The

[4] Assuming that the parses are correct, the head reordering rules in Table 1 have to apply categorically to change the English order into the Japanese order because English is head initial and Japanese is head final without any exceptions. Similarly, most of the modifier reordering rules in Table 2 have to apply categorically because most modifiers follow the modified head phrase in English, e.g. a relative clause modifier follows the head noun phrase, a prepositional phrase modifier follows the head noun phrase, etc., whereas modifier phrases precede the modified head phrases in Japanese.

extended domain of locality of tree adjoining grammars should suffice to capture non-CFG reordering rules for many language pairs. Alternatively, we can adopt enriched feature representations so that a tree of depth one can actually convey information on a tree of several depths, such as parent annotation of (Klein and Manning, 2003).

Regarding the issue of modeling, we can introduce a rich set of features, as in (Ittycheriah and Roukos, 2007), the weights of which are trained to ensure that the tree-to-string grammar rules generating the accurate target orders are assigned probabilities high enough not to get pruned out in the translation derivation.

Acknowledgement

We would like to acknowledge IBM RTTS (Realtime Translation Systems) team for technical discussions on the topic and the provision of linguistic resources. We also would like to thank IBM SMT (Statistical Machine Translation) team for various software tools and the anonymous reviewers for their helpful comments .

References

Y. Al-Onaizan and K. Papineni. 2006. Distortion models for statistical machine translation. *Proceedings of ACL-COLING*. Pages 529-536.

C. Baker, S. Bethard, M. Bloodgood, R. Brown, C. Callison-Burch, G. Coppersmith, B. Dorr, W. Filardo, K. Giles, A. Irvine, M. Kayser, L. Levin, J. Martineau, J. Mayfield, S. Miller, A. Phillips, A. Philpot, C. Piatko, L. Schwartz, D. Zajic. 2010. Semantically Informed Machine Translation (SIMT). Final Report of the 2009 Summer Camp for Applied Language Exploration.

P. Brown, V. Della Pietra, S. Della Pietra, and R. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation, *Computational Linguistics*, 19(2):263−311.

X. Carreras and M. Collins. 2009. Non-projective parsing for statistical machine translation. *Proceedings of the 2009 EMNLP*. Pages 200-209.

P. Chang and C. Toutanova. 2007. A Discriminative Syntactic Word Order Model for Machine Translation. *Proceedings of ACL*. Pages 9-16.

D. Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. *Proceedings of ACL*. Pages 263-270.

D. Chiang, W. Wang and Kevin Knight. 2009. 11,001 new features for statistical machine translation. *Proceedings of HLT-NAACL*. Pages 218-226.

M. Collins, P. Koehn, I. Kucerova. 2005. Clause Restructuring for Statistical Machine Translation. *Proceedings of ACL*. Pages 531-540.

Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. *Proceedings of ACL*. Pages 541-548.

J. Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*. Vol. 13. Pages 94–102.

M. Galley and C. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. *Proceedings of EMNLP*.

N. Habash. 2007. Syntactic Preprocessing for Statistical Machine Translation. *Proceedings of the Machine Translation Summit*.

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw and R. Weischedel. 2006. OntoNotes: The 90% Solution. *Proceedings of HLT*. Pages 57-60.

A. Ittycheriah and S. Roukos. 2007. Direct Translation Model 2. *Proceedings of HLT-NAACL*.

A. Joshi and Y. Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and K. Salomaa, editors, *Handbook of Formal Languages*, volume 3. Springer.

D. Klein and C. Manning. 2003. Accurate Unlexicalized Parsing. *Proceedings of 41st ACL*.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation, *Proceedings of HLT−NAACL*. Pages 48−54.

Y. Liu, Q. Liu and S. Lin. 2006. Tree-to-string alignment template for statistical machine translation. *Proceedings of ACL-COLING*.

Y. Liu, Y. Huang, Q. Liu, and S. Lin. 2007. Forest-to-string statistical translation rules. *Proceedings of the 45th ACL*.

D. Marcu, W. Wang, A. Echihabi and K. Knight. 2006. SPMT: Statistical Machine Translation with Syntactified Target Language Phrases. *Proceedings of EMNLP*. Pages 44-52.

M. Marcus, B. Santorini and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: the Penn Treebank. *Computational Linguistics,* 19(2): 313-330.

F. J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*: Vol. 30. Pages 417– 449.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. *Proceddings of ACL*. Pages 311–318.

A. Ratnaparkhi. 1999. Learning to Parse Natural Language with Maximum Entropy Models. *Machine Learning: Vol. 34*. Pages 151-178.

L. Shen, J. Xu and R. Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. *Proceedings of ACL.*

L. Shen, J. Xu, B. Zhang, S. Matsoukas and Ralph Weischedel. 2009. Effective Use of Linguistic and Contextual Information for Statistical Machine Translation. *Proceedings of EMNLP*.

C. Wang, M. Collins, P. Koehn. 2007. Chinese Syntactic Reordering for Statistical Machine Translation. *Proceedings of EMNLP-CoNLL*.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3): 377-404.

F. Xia and M. McCord. 2004. Improving a Statistical MT System with Automatically Learned Rewrite Patterns. *Proceedings of COLING*.

P. Xu, J. Kang, M. Ringgaard, F. Och. 2009. Using a dependency parser to improve SMT for subject-verb-object languages. *Proceedings of HLT-NAACL*.

K. Yamada and K. Knight. 2001. A Syntax-based Statistical Translation Model. *Proceedings of the 39th ACL*. Pages 523-530.

H. Zhang, L. Huang, D. Gildea and K. Knight. 2006. Synchronous binarization for machine translation. *Proceedings of the HLT-NAACL*. Pages 256-263.

B. Zhao and Y. Al-onaizan. 2008. Generalizing Local and Non-Local Word-Reordering Patterns for Syntax-Based Machine Translation. *Proceedings of EMNLP*. Pages 572-581.

A. Zollmann and A. Venugopal. 2006. Syntax augmented machine translation via chart parsing. *Proceedings of NAACL 2006 -Workshop on statistical machine translation*.

A. Zollmann, A. Venugopal, F. Och and J. Ponte. 2008. A Systematic Comparison of Phrase-Based, Hierarchical and Syntax-Augmented MT. *Proceedings of COLING*.

Sentiment Classification and Polarity Shifting

Shoushan Li[†‡] **Sophia Yat Mei Lee**[†] **Ying Chen**[†] **Chu-Ren Huang**[†] **Guodong Zhou**[‡]

[†]Department of CBS
The Hong Kong Polytechnic University
{shoushan.li, sophiaym, chenying3176, churenhuang}@gmail.com

[‡] Natural Language Processing Lab
School of Computer Science and Technology
Soochow University
gdzhou@suda.edu.cn

Abstract

Polarity shifting marked by various linguistic structures has been a challenge to automatic sentiment classification. In this paper, we propose a machine learning approach to incorporate polarity shifting information into a document-level sentiment classification system. First, a feature selection method is adopted to automatically generate the training data for a binary classifier on polarity shifting detection of sentences. Then, by using the obtained binary classifier, each document in the original polarity classification training data is split into two partitions, polarity-shifted and polarity-unshifted, which are used to train two base classifiers respectively for further classifier combination. The experimental results across four different domains demonstrate the effectiveness of our approach.

1 Introduction

Sentiment classification is a special task of text classification whose objective is to classify a text according to the sentimental polarities of opinions it contains (Pang et al., 2002), e.g., *favorable* or *unfavorable*, *positive* or *negative*. This task has received considerable interests in the computational linguistic community due to its potential applications.

In the literature, machine learning approaches have dominated the research in sentiment classification and achieved the state-of-the-art performance (e.g., Kennedy and Inkpen, 2006; Pang et al., 2002). In a typical machine learning approach, a document (text) is modeled as a bag-of-words, i.e. a set of content words without any word order or syntactic relation information. In other words, the underlying assumption is that the sentimental orientation of the whole text depends on the sum of the sentimental polarities of content words. Although this assumption is reasonable and has led to initial success, it is linguistically unsound since many function words and constructions can shift the sentimental polarities of a text. For example, in the sentence '*The chair is not comfortable*', the polarity of the word '*comfortable*' is positive while the polarity of the whole sentence is reversed because of the negation word '*not*'. Therefore, the overall sentiment of a document is not necessarily the sum of the content parts (Turney, 2002). This phenomenon is one main reason why machine learning approaches fail under some circumstances.

As a typical case of polarity shifting, negation has been paid close attention and widely studied in the literature (Na et al., 2004; Wilson et al., 2009; Kennedy and Inkpen, 2006). Generally, there are two steps to incorporate negation information into a system: negation detection and negation classification. For negation detection, some negation trigger words, such as '*no*', '*not*', and '*never*', are usually applied to recognize negation phrases or sentences. As for negation classification, one way to import negation information is to directly reverse the polarity of the words which contain negation trigger words as far as term-counting approaches are considered (Kennedy and Inkpen, 2006). An alternative way is to add some negation features (e.g., negation bigrams or negation phrases) into

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 635–643,
Beijing, August 2010

machine learning approaches (Na et al., 2004). Such approaches have achieved certain success.

There are, however, some shortcomings with current approaches in incorporating negation information. In terms of negation detection, firstly, the negation trigger word dictionary is either manually constructed or relies on existing resources. This leads to certain limitations concerning the quality and coverage of the dictionary. Secondly, it is difficult to adapt negation detection to other languages due to its language dependence nature of negation constructions and words. Thirdly, apart from negation, many other phenomena, e.g., contrast transition with trigger words like '*but*', '*however*', and '*nevertheless*', can shift the sentimental polarity of a phrase or sentence. Therefore, considering negation alone is inadequate to deal with the polarity shifting problem, especially for document-level sentiment classification.

In terms of negation classification, although it is easy for term-counting approaches to integrate negation information, they rarely outperform a machine learning baseline (Kennedy and Inkpen, 2006). Even for machine learning approaches, although negation information is sometimes effective for local cases (e.g., *not good*), it fails on long-distance cases (e.g., *I don't think it is good*).

In this paper, we first propose a feature selection method to automatically generate a large scale polarity shifting training data for polarity shifting detection of sentences. Then, a classifier combination method is presented for incorporating polarity shifting information. Compared with previous ones, our approach highlights the following advantages：First of all, we apply a binary classifier to detect polarity shifting rather than merely relying on trigger words or phrases. This enables our approach to handle different kinds of polarity shifting phenomena. More importantly, a feature selection method is presented to automatically generate the labeled training data for polarity shifting detection of sentences.

The remainder of this paper is organized as follows. Section 2 introduces the related work of sentiment classification. Section 3 presents our approach in details. Experimental results are presented and analyzed in Section 4. Finally, Section 5 draws the conclusion and outlines the future work.

2 Related Work

Generally, sentiment classification can be performed at four different levels: word level (Wiebe, 2000), phrase level (Wilson et al., 2009), sentence level (Kim and Hovy, 2004; Liu et al., 2005), and document level (Turney, 2002; Pang et al., 2002; Pang and Lee, 2004; Riloff et al., 2006). This paper focuses on document-level sentiment classification.

In the literature, there are mainly two kinds of approaches on document-level sentiment classification: term-counting approaches (lexicon-based) and machine learning approaches (corpus-based). Term-counting approaches usually involve deriving a sentiment measure by calculating the total number of negative and positive terms (Turney, 2002; Kim and Hovy, 2004; Kennedy and Inkpen, 2006). Machine learning approaches recast the sentiment classification problem as a statistical classification task (Pang and Lee, 2004). Compared to term-counting approaches, machine learning approaches usually achieve much better performance (Pang et al., 2002; Kennedy and Inkpen, 2006), and have been adopted to more complicated scenarios, such as domain adaptation (Blitzer et al., 2007), multi-domain learning (Li and Zong, 2008) and semi-supervised learning (Wan, 2009; Dasgupta and Ng, 2009) for sentiment classification.

Polarity shifting plays a crucial role in phrase-level, sentence-level, and document-level sentiment classification. However, most of previous studies merely focus on negation shifting (polarity shifting caused by the negation structure). As one pioneer research on sentiment classification, Pang et al. (2002) propose a machine learning approach to tackle negation shifting by adding the tag 'not' to every word between a negation trigger word/phrase (e.g., *not*, *isn't*, *didn't*, etc.) and the first punctuation mark following the negation trigger word/phrase. To their disappointment, considering negation shifting has a negligible effect and even slightly harms the overall performance. Kennedy and Inkpen (2006) explore negation shifting by incorporating negation bigrams as additional features into machine learning approaches. The experimental results show that considering sentiment shifting greatly improves the performance of term-counting approaches but only slightly improves the performance of machine learning approaches. Other studies such as Na et al. (2004), Ding et al. (2008), and Wilson et al. (2009) also explore negation shifting and achieve some improvements[1]. Nonetheless, as far as machine learning approaches are concerned, the improvement is rather insignificant (normally less than 1%). More recently, Ikeda et al. (2008) first propose a machine learning approach to detect polarity shifting for sentence-level sentiment classification, based on a manually-constructed dictionary containing thousands of positive and negative sentimental words, and then adopt a term-counting approach to incorporate polarity shifting information.

3 Sentiment Classification with Polarity Shifting Detection

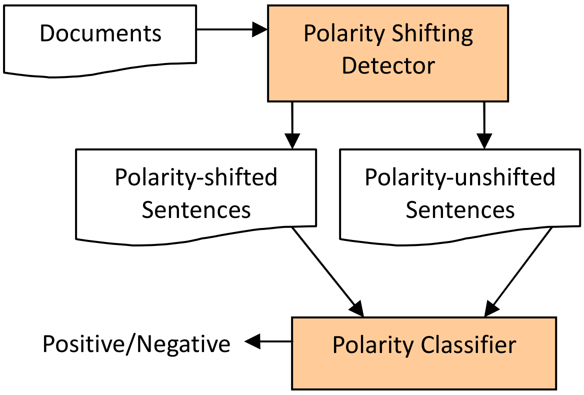

Documents
Polarity Shifting Detector
Polarity-shifted Sentences
Polarity-unshifted Sentences
Positive/Negative
Polarity Classifier

Figure 1: General framework of our approach

The motivation of our approach is to improve the performance of sentiment classification by robust treatment of sentiment polarity shifting between sentences. With the help of a binary classifier, the sentences in a document are divided into two parts: sentences which contain polarity shifting structures and sentences without any polarity shifting structure. Figure 1 illustrates the general framework of our approach. Note that this framework is a general one, that is, different polarity shifting detection methods can be applied to differentiate polarity-shifted sentences from those polarity-unshifted sentences and different polarity classification methods can be adopted to incorporate sentiment shifting information. For clarification, the training data used for polarity shifting detection and polarity classification are referred to as the polarity shifting training data and the polarity classification training data, respectively.

3.1 Polarity Shifting Detection

In this paper, polarity shifting means that the polarity of a sentence is different from the polarity expressed by the sum of the content words in the sentence. For example, in the sentence "*I am not disappointed*", the negation structure makes the polarity of the word '*disappointed*' different from that of the whole sentence (*negative* vs. *positive*). Apart from the negation structure, many other linguistic structures allow polarity shifting, such as contrast transition, modals, and pre-suppositional items (Polanyi and Zaenen, 2006). We refer these structures as polarity shifting structures.

One of the great challenges in building a polarity shifting detector lies on the lack of relevant training data since manually creating a large scale corpus of polarity shifting sentences is time-consuming and labor-intensive. Ikeda et al. (2008) propose an automatic way for collecting the polarity shifting training data based on a manually-constructed large-scale dictionary. Instead, we adopt a feature selection method to build a large scale training corpus of polarity shifting sentences, given only the already available document-level polarity classification training data. With the help of the feature selection method, the top-ranked word features with strong sentimental polarity orientation, e.g., '*great*', '*love*', '*worst*' are first chosen as the polarity trigger words. Then, those sentences with the top-ranked polarity trigger words in both categories of positive and negative documents are selected. Finally, those candidate sentences taking opposite-polarity compared to the containing trigger word are deemed as polarity-shifted.

The basic idea of automatically generating the polarity shifting training data is based on the assumption that the real polarity of a word or phrase is decided by the major polarity category where the word or phrase appears more often. As a result, the sentences in the

[1] Note that Ding et al. (2006) also consider *but*-clause, another important structure for sentiment shifting. Wilson et al. (2009) use conjunctive and dependency relations among polarity words.

frequently-occurring category would be seen as polarity-unshifted while the sentences in the infrequently-occurring category would be seen as polarity-shifted.

In the literature, various feature selection methods, such as Mutual Information (MI), Information Gain (IG) and Bi-Normal Separation (BNS) (Yang and Pedersen, 1997; Forman 2003), have been employed to cope with the problem of the high-dimensional feature space which is normal in sentiment classification.

In this paper, we employ the theoretical framework, proposed by Li et al. (2009), including two basic measurements, i.e. *frequency measurement* and *ratio measurement*, where the first measures, the document frequency of a term in one category, and the second measures, the ratio between the document frequency in one category and other categories. In particular, a novel method called Weighed Frequency and Odds (WFO) is proposed to incorporate both basic measurements:

$$WFO(t,c_i)=P(t\,|\,c_i)^{\lambda}\{\max(0,\log\frac{P(t\,|\,c_i)}{P(t\,|\,\overline{c_i})})\}^{1-\lambda}$$

where $P(t\,|\,c_i)$ denotes the probability that a document *x* contains the term *t* with the condition that *x* belongs to category c_i ; $P(t\,|\,\overline{c_i})$ denotes the probability that a document *x* contains the term *t* with the condition that *x* does not belong to category c_i. The left part of the formula $P(t\,|\,c_i)$ implies the first basic measurement and the right part $\log(P(t\,|\,c_i)/P(t\,|\,\overline{c_i}))$ implies the second one. The parameter λ $(0\le\lambda\le1)$ is thus to tune the weight between the two basic measurements. Especially, when λ equals 0, the WFO method fades to the MI method which fully prefers the second basic measurement.

Figure 2 illustrates our algorithm for automatically generating the polarity shifting training data where c_1 and c_2 denote the two sentimental orientation categories, i.e. negative and positive. *Step A* segments a document into sentences with punctuations. Besides, two special words, ‘*but*’ and ‘*and’*, are used to further segment some contrast transition structures and compound sentences. *Step B* employs the WFO method to rank all features including the words. *Step D* extracts those polarity-shifted and polarity-unshifted sentences containing t_{top-i} where $N_{\max}$ denotes the upper-limit number of sentences in each category of the polarity shifting training data and *#(x)* denotes the total number of the elements in *x*. Apart from that, the first word in the following sentence is also included to capture a common kind of long-distance polarity shifting structure: contrast transition. Thus, important trigger words like ‘*however*’ and ‘*but*’ may be considered. Finally, *Step E* guarantees the balance between the two categories of the polarity shifting training data.

Given the polarity shifting training data, we apply SVM classification algorithm to train a polarity-shifting detector with word unigram features.

```
Input:
    The polarity classification training data: the negative sentimental document set $D_{c_1}$ and the positive sentimental document set $D_{c_2}$.
Output:
    The polarity shifting training data: the polarity-unshifted sentence set $S_{unshift}$ and the polarity-shifted sentence set $S_{shift}$.
Procedure:
A.  Segment documents $D_{c_1}$ and $D_{c_2}$ to single sentences $S_{c_1}$ and $S_{c_2}$.
B.  Apply feature selection on the polarity classification training data and get the ranked features, $(t_{top-1},...,t_{top-i},...,t_{top-N})$
C.  $S_{shift}$ = {}, $S_{unshift}$ = {}
D.  For $t_{top-i}$ in $(t_{top-1},...,t_{top-i},...,t_{top-N})$ :
    D1) if #($S_{shift}$)> $N_{\max}$ : break
    D2) Collect all sentences $S_{top-i,c_1}$ and $S_{top-i,c_2}$ which contain $t_{top-i}$ from $S_{c_1}$ and $S_{c_2}$ respectively
    D3) if #($S_{top-i,c_1}$)>#($S_{top-i,c_2}$):
                put $S_{top-i,c_2}$ into $S_{shift}$
                put $S_{top-i,c_1}$ into $S_{unshift}$
            else:
                put $S_{top-i,c_1}$ into $S_{shift}$
                put $S_{top-i,c_2}$ into $S_{unshift}$
E.  Randomly select $N_{\max}$ sentences from $S_{unshift}$ as the output of $S_{unshift}$
```

Figure 2: The algorithm for automatically generating the polarity shifting training data

3.2 Polarity Classification with Classifier Combination

After polarity shifting detection, each document in the polarity classification training data is divided into two parts, one containing polarity-shifted sentences and the other containing polarity-unshifted sentences, which are used to form the polarity-shifted training data and the polarity-unshifted training data. In this way, two different polarity classifiers, f_1 and f_2 , can be trained on the polarity-shifted training data and the polarity-unshifted training data respectively. Along with classifier f_3 , trained on all original polarity classification training data, we now have three base classifiers in hand for possible classifier combination via a multiple classifier system.

The key issue in constructing a multiple classifier system (MCS) is to find a suitable way to combine the outputs of the base classifiers. In MCS literature, various methods are available for combining the outputs, such as fixed rules including the voting rule, the product rule and the sum rule (Kittler et al., 1998) and trained rules including the weighted sum rule (Fumera and Roli, 2005) and the meta-learning approaches (Vilalta and Drissi, 2002). In this study, we employ the product rule, a popular fixed rule, and stacking (Džeroski and Ženko, 2004), a well-known trained rule, to combine the outputs.

Formally, each base classifier provides some kind of confidence measurements, e.g., posterior probabilities of the test sample belonging to each class. Formally, each base classifier f_l $(l = 1, 2, 3)$ assigns a test sample (denoted as x_l) a posterior probability vector $\vec{P}(x_l)$:

$$\vec{P}(x_l) = (p(c_1 \mid x_l), p(c_2 \mid x_l))^t$$

where $p(c_1 \mid x_l)$ denotes the probability that the l-th base classifier considers the sample belonging c_1 .

The product rule combines the base classifiers by multiplying the posterior possibilities and using the multiplied possibility for decision, i.e.

$$assign \quad y \to c_j \; when\; j = \arg\max_i \prod_{l=1}^{3} p(c_i \mid x_l)$$

Stacking belongs to well-known meta-learning (Vilalta and Drissi, 2002). The key idea behind meta-learning is to train a meta-classifier with input attributes that are the outputs of the base classifiers. Hence, meta-learning usually needs some development data for generating the meta-training data. Let x' denote a feature vector of a sample from the development data. The output of the l-th base classifier f_l on this sample is the probability distribution over the category set $\{c_1, c_2\}$, i.e.

$$\vec{P}(x'_l) = (p(c_1 \mid x'_l), p_l(c_2 \mid x'_l))$$

A meta-classifier can be trained using the development data with the meta-level feature vector $x^{meta} \in R^{2\times3}$

$$x^{meta} = (\vec{P}(x'_{l=1}), \vec{P}(x'_{l=2}), \vec{P}(x'_{l=3}))$$

Stacking is a specific meta-learning rule, in which a leave-one-out or a cross-validation procedure on the training data is applied to generate the meta-training data instead of using extra development data. In our experiments, we perform stacking with 10-fold cross-validation to generate the meta-training data.

4 Experimentation

4.1 Experimental Setting

The experiments are carried out on product reviews from four domains: books, DVDs, electronics, and kitchen appliances (Blitzer et al., 2007)[2]. Each domain contains 1000 positive and 1000 negative reviews.

For sentiment classification, all classifiers including the polarity shifting detector, three base classifiers and the meta-classifier in stacking are trained by SVM using the SVM-light tool [3] with Logistic Regression method for probability measuring (Platt, 1999).

In all the experiments, each dataset is randomly and evenly split into two subsets: 50% documents as the training data and the remaining 50% as the test data. The features include word unigrams and bigrams with Boolean weights.

4.2 Experimental Results on Polarity Shifting Data

To better understand the polarity shifting phenomena in document-level sentiment classification, we randomly investigate 200

[2] This data set is collected by Blitzer et al. (2007): http://www.seas.upenn.edu/~mdredze/datasets/sentiment/

[3] It is available at: http://svmlight.joachims.org/

polarity-shifted sentences, together with their contexts (i.e. the sentences before and after it), automatically generated by the WFO ($\lambda = 0$) feature selection method. We find that nearly half of the automatically generated polarity-shifted sentences are actually polarity-unshifted sentences or difficult to decide. That is to say, the polarity shifting training data is noisy to some extent. One main reason is that some automatically selected trigger words do not really contain sentiment information, e.g., ‘*hear*’, ‘*information*’ etc. Another reason is that some reversed opinion is given in a review without any explicit polarity shifting structures.

To gain more insights, we manually checked 100 sentences which are explicitly polarity-shifted and can also be judged by human according to their contexts. Table 1 presents some typical structures causing polarity shifting. It shows that the most common polarity shifting type is Explicit Negation (37%), usually expressed by trigger words such as ‘*not*’, ‘*no*’, or ‘*without*’, e.g., in the sentence ‘*I am not happy with this flashcard at all*’. Another common type of polarity shifting is Contrast Transition (20%), expressed by trigger words such as ‘*however*’, e.g., in the sentence ‘*It is large and stylish, however, I cannot recommend it because of the lid*’. Other less common yet productive polarity shifting types include Exception and Until. Exception structure is usually expressed by the trigger phrase ‘*the only*’ to indicate the one and only advantage of the product, e.g., in the sentence ‘*The only thing that I like about it is that bamboo is a renewable resource*’. Until structure is often expressed by the trigger word ‘*until*’ to show the reversed polarity, e.g. in the sentence ‘*This unit was a great addition until the probe went bad after only a few months*’.

Polarity Shifting Structures	Trigger Words/Phrases	Distribution (%)
Explicit Negation	*not, no, without*	37
Contrast Transition	*but, however, unfortunately*	20
Implicit Negation	*avoid, hardly,*	7
False Impression	*look, seem*	6
Likelihood	*probably, perhaps*	5
Counter-factual	*should, would*	5
Exception	*the only*	5
Until	*until*	3

Table 1: Statistics on various polarity shifting structures

4.3 Experimental Results on Polarity Classification

For comparison, several classifiers with different classification methods are developed.

1) Baseline classifier, which applies SVM with all unigrams and bigrams. Note that it also serves as a base classifier in the following combined classifiers.

2) Base classifier 1, a base classifier for the classifier combination method. It works on the polarity-unshifted data.

3) Base classifier 2, another base classifier for the classifier combination method. It works on the polarity-shifted data.

4) Negation classifier, which applies SVM with all unigrams and bigrams plus negation bigrams. It is a natural extension of the baseline classifier with the consideration of negation bigrams. In this study, the negation bigrams are collected using some negation trigger words, such as ‘*not*’ and ‘*never*’. If a negation trigger word is found in a sentence, each word in the sentence is attached with the word ‘*_not*’ to form a negation bigram.

5) Product classifier, which combines the baseline classifier, the base classifier 1 and the base classifier 2 using the product rule.

6) Stacking classifier, a combined classifier similar to the **Product classifier**. It uses the stacking classifier combination method instead of the product rule.

Please note that we do not compare our approach with the one as proposed in Ikeda et al. (2008) due to the absence of a manually-collected sentiment dictionary. Besides, it is well known that a combination strategy itself is capable of improving the classification performance. To justify whether the improvement is due to the combination strategy or our polarity shifting detection or both, we first randomly split the training data into two portions and train two base classifiers on each portion, then apply the stacking method to combine them along with the baseline classifier. The corresponding results are shown as ‘Random+Stacking’ in Table 2. Finally, in our experiments, *t*-test is performed to evaluate the significance of the performance improvement between two systems employing different methods (Yang and Liu, 1999).

Domain	Baseline	Base Classifier 1	Base Classifier 2	Negation Classifier	Random + Stacking	Shifting + Product	Shifting + Stacking
Book	0.755	0.756	0.670	0.759	0.764	0.772	**0.785**
DVD	0.750	0.743	0.667	0.748	0.759	0.768	**0.770**
Electronic	0.779	0.786	0.711	0.785	0.789	0.820	**0.830**
Kitchen	0.818	0.814	0.683	0.826	0.835	0.840	**0.849**

Table 2: Performance comparison of different classifiers with equally-splitting between training and test data

Performance comparison of different classifiers

Table 2 shows the accuracy results of different methods using 2000 polarity shifted sentences and 2000 polarity-unshifted sentences to train the polarity shifting detector (N_{max}=2000). Compared to the baseline classifier, it shows that: 1) The base classifier 1, which only uses the polarity-unshifted sentences as the training data, achieves similar performance. 2) The base classifier 2 achieves much lower performance due to much fewer sentences involved. 3) Including negation bigrams usually allows insignificant improvements (*p-value*>0.1), which is consistent with most of previous works (Pang et al., 2002; Kennedy and Inkpen, 2006). 4) Both the product and stacking classifiers with polarity shifting detection significantly improve the performance (*p-value*<0.05). Compared to the product rule, the stacking classifier is preferable, probably due to the performance unbalance among the individual classifiers, e.g., the performance of the base classifier 2 is much lower than the other two. Although stacking with two randomly generated base classifiers, i.e. "Random + Stacking", also consistently outperforms the baseline classifier, the improvements are much lower than what has been achieved by our approach. This suggests that both the classifier combination strategy and polarity shifting detection contribute to the overall performance improvement.

Effect of WFO feature selection method

Figure 3 presents the accuracy curve of the stacking classifier when using different Lambda (λ) values in the WFO feature selection method. It shows that those feature selection methods which prefer frequency information, e.g., MI and BNS, are better in automatically generating the polarity shifting training data. This is reasonable since high frequency terms, e.g., '*is*', '*it*', '*a*', etc., tend to obey our assumption that the real polarity of one top term should belong to the polarity category where the term appears frequently.

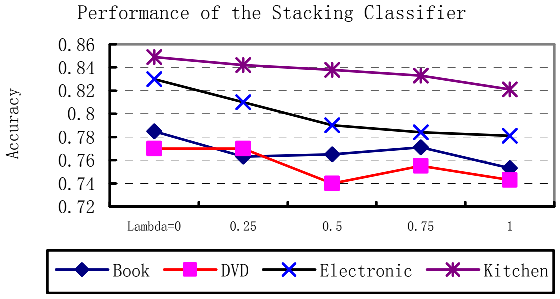


Figure 3: Performance of the stacking classifier using WFO with different Lambda (λ) values

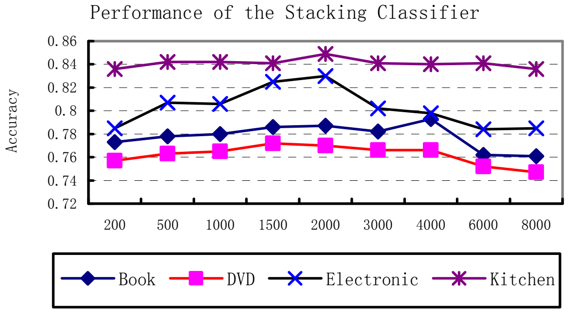


Figure 4: Performance of the stacking classifier over different sizes of the polarity shifting training data (with N_{max} sentences in each category)

Effect of a classifier over different sizes of the polarity shifting training data

Another factor which might influence the overall performance is the size of the polarity shifting training data. Figure 4 presents the overall performance on different numbers of the polarity shifting sentences when using the stacking classifier. It shows that 1000 to 4000 sentences are enough for the performance improvement. When the number is too large, the noisy training data may harm polarity shifting detection. When the number is too small, it is not enough for the automatically generated polarity shifting training data to capture various polarity shifting structures.

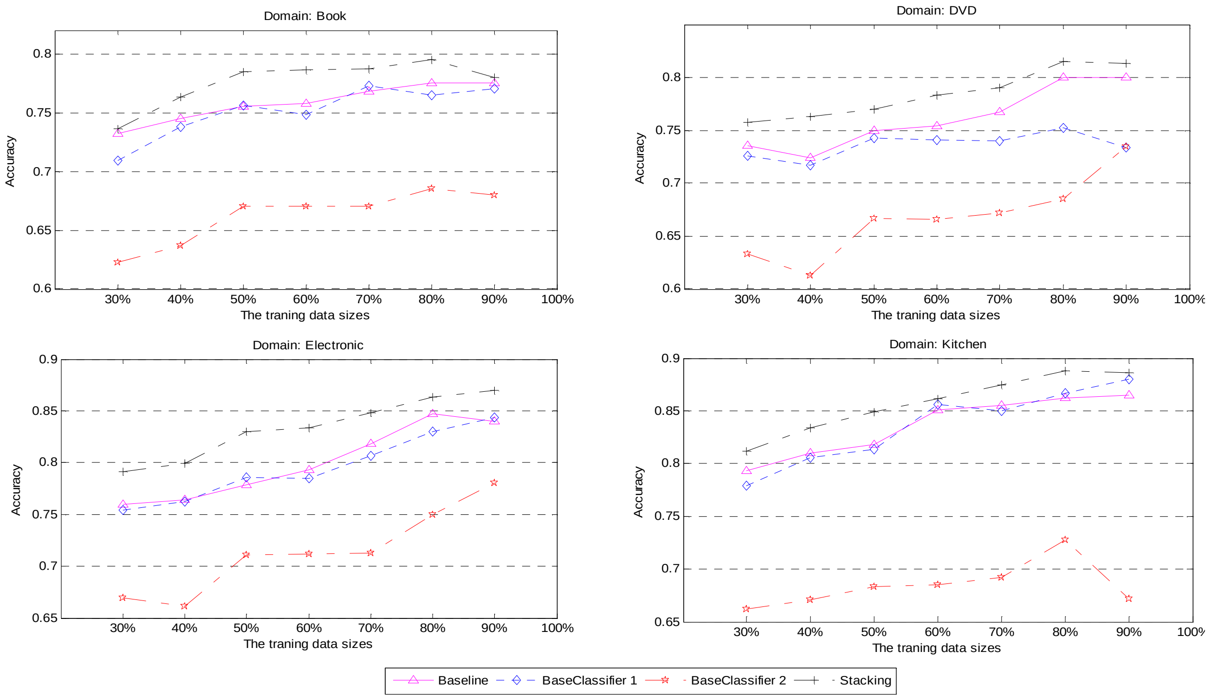


Figure 5: Performance of different classifiers over different sizes of the polarity classification training data

Effect of different classifiers over different sizes of the polarity classification training data

Figure 5 shows the classification results of different classifiers with varying sizes of the polarity classification training data. It shows that our approach is able to improve the overall performance robustly. We also notice the big difference between the performance of the baseline classifier and that of the base classifier 1 when using 30% training data in Book domain and 90% training data in DVD domain. Detailed exploration of the polarity shifting sentences in the training data shows that this difference is mainly attributed to the poor performance of the polarity shifting detector. Even so, the stacking classifier guarantees no worse performance than the baseline classifier.

5 Conclusion and Future Work

In this paper, we propose a novel approach to incorporate polarity shifting information into document-level sentiment classification. In our approach, we first propose a machine-learning-based classifier to detect polarity shifting and then apply two classifier combination methods to perform polarity classification. Particularly, the polarity shifting training data is automatically generated through a feature selection method. As shown in our experimental results, our approach is able to consistently improve the overall performance across different domains and training data sizes, although the automatically generated polarity shifting training data is prone to noise. Furthermore, we conclude that those feature selection methods, which prefer frequency information, e.g., MI and BNS, are good choices for generating the polarity shifting training data.

In our future work, we will explore better ways in generating less-noisy polarity shifting training data. In addition, since our approach is language-independent, it is readily applicable to sentiment classification tasks in other languages.

For availability of the automatically generated polarity shifting training data, please contact the first author (for research purpose only).

Acknowledgments

This research work has been partially supported by Start-up Grant for Newly Appointed Professors, No. 1-BBZM in the Hong Kong Polytechnic University and two NSFC grants, No. 60873150 and No. 90920004. We also thank the three anonymous reviewers for their helpful comments.

References

Blitzer J., M. Dredze, and F. Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. In *Proceedings of ACL-07*.

Dasgupta S. and V. Ng. 2009. Mine the Easy and Classify the Hard: Experiments with Automatic Sentiment Classification. In *Proceedings of ACL-IJCNLP-09*.

Ding X., B. Liu, and P. Yu. 2008. A Holistic Lexicon-based Approach to Opinion Mining. In *Proceedings of the International Conference on Web Search and Web Data Mining, WSDM-08*.

Džeroski S. and B. Ženko. 2004. Is Combining Classifiers with Stacking Better than Selecting the Best One? *Machine Learning*, vol.54(3), pp.255-273, 2004.

Forman G. 2003. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. *The Journal of Machine Learning Research*, 3(1), pp.1289-1305.

Fumera G. and F. Roli. 2005. A Theoretical and Experimental Analysis of Linear Combiners for Multiple Classifier Systems. *IEEE Trans. PAMI*, vol.27, pp.942–956, 2005

Ikeda D., H. Takamura, L. Ratinov, and M. Okumura. 2008. Learning to Shift the Polarity of Words for Sentiment Classification. In *Proceedings of IJCNLP-08*.

Kennedy, A. and D. Inkpen. 2006. Sentiment Classification of Movie Reviews using Contextual Valence Shifters. *Computational Intelligence*, vol.22(2), pp.110-125, 2006.

Kim S. and E. Hovy. 2004. Determining the Sentiment of Opinions. In *Proceedings of COLING-04*.

Kittler J., M. Hatef, R. Duin, and J. Matas. 1998. On Combining Classifiers. *IEEE Trans. PAMI*, vol.20, pp.226-239, 1998

Li S., R. Xia, C. Zong, and C. Huang. 2009. A Framework of Feature Selection Methods for Text Categorization. In *Proceedings of ACL-IJCNLP-09*.

Li S. and C. Zong. 2008. Multi-domain Sentiment Classification. In *Proceedings of ACL-08: HLT*, short paper.

Liu B., M. Hu, and J. Cheng. 2005. Opinion Observer: Analyzing and Comparing Opinions on the Web. In *Proceedings of WWW-05*.

Na J., H. Sui, C. Khoo, S. Chan, and Y. Zhou. 2004. Effectiveness of Simple Linguistic Processing in Automatic Sentiment Classification of Product Reviews. In *Conference of the International Society for Knowledge Organization (ISKO-04)*.

Pang B. and L. Lee. 2004. A Sentimental Education: Sentiment Analysis using Subjectivity Summarization based on Minimum Cuts. In *Proceedings of ACL-04*.

Pang B., L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Proceedings of EMNLP-02*.

Platt J. 1999. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In: A. Smola, P. Bartlett, B. Schoelkopf and D. Schuurmans (Eds.): *Advances in Large Margin Classiers*. MIT Press, Cambridge, 61–74.

Polanyi L. and A. Zaenen. 2006. Contextual Valence Shifters. *Computing attitude and affect in text: Theory and application*. Springer Verlag.

Riloff E., S. Patwardhan, and J. Wiebe. 2006. Feature Subsumption for Opinion Analysis. In *Proceedings of EMNLP-06*.

Turney P. 2002. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In *Proceedings of ACL-02*.

Vilalta R. and Y. Drissi. 2002. A Perspective View and Survey of Meta-learning. *Artificial Intelligence Review*, 18(2), pp. 77–95.

Wan X. 2009. Co-Training for Cross-Lingual Sentiment Classification. In *Proceedings of ACL-IJCNLP-09*.

Wiebe J. 2000. Learning Subjective Adjectives from Corpora. In *Proceedings of AAAI-2000*.

Wilson T., J. Wiebe, and P. Hoffmann. 2009. Recognizing Contextual Polarity: An Exploration of Features for Phrase-Level Sentiment Analysis. *Computational Linguistics*, vol.35(3), pp.399-433, 2009.

Yang Y. and X. Liu, X. 1999. A Re-Examination of Text Categorization methods. In *Proceedings of SIGIR-99*.

Yang Y. and J. Pedersen. 1997. A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of ICML-97*.

Improving Corpus Comparability for Bilingual Lexicon Extraction from Comparable Corpora

Bo Li, Eric Gaussier
Laboratoire d'Informatique de Grenoble (LIG)
Université de Grenoble
firstname.lastname@imag.fr

Abstract

Previous work on bilingual lexicon extraction from comparable corpora aimed at finding a good representation for the usage patterns of source and target words and at comparing these patterns efficiently. In this paper, we try to work it out in another way: improving the quality of the comparable corpus from which the bilingual lexicon has to be extracted. To do so, we propose a measure of comparability and a strategy to improve the quality of a given corpus through an iterative construction process. Our approach, being general, can be used with any existing bilingual lexicon extraction method. We show here that it leads to a significant improvement over standard bilingual lexicon extraction methods.

1 Introduction

Bilingual dictionaries are an essential resource in many multilingual natural language processing (NLP) tasks such as machine translation (Och and Ney, 2003) and cross-language information retrieval (CLIR) (Ballesteros and Croft, 1997). Hand-coded dictionaries are of high quality, but expensive to build and researchers have tried, since the end of the 1980s, to automatically extract bilingual lexicons from parallel corpora (see (Chen, 1993; Kay and Röscheisen, 1993; Melamed, 1997a; Melamed, 1997b) for early work). Parallel corpora are however difficult to get at in several domains, and the majority of bilingual collections are comparable and not parallel. Due to their low cost of acquisition, several researchers have tried to exploit such comparable corpora for bilingual lexicon extraction (Fung and McKeown, 1997; Fung and Yee, 1998; Rapp, 1999; Déjean et al., 2002; Gaussier et al., 2004; Robitaille et al., 2006; Morin et al., 2007; Yu and Tsujii, 2009). The notion of comparability is however a loose one, and comparable corpora range from lowly comparable ones to highly comparable ones and parallel ones. For data-driven NLP techniques, using better corpora often leads to better results, a fact which should be true for the task of bilingual lexicon extraction. This point has largely been ignored in previous work on the subject. In this paper, we develop a well-founded strategy to improve the quality of a comparable corpus, so as to improve in turn the quality of the bilingual lexicon extracted. To do so, we first propose a measure of comparability which we then use in a method to improve the quality of the existing corpus.

The remainder of the paper is organized as follows: Section 2 introduces the experimental materials used for the different evaluations; comparability measures are then presented and evaluated in Section 3; in Section 4, we detail and evaluate a strategy to improve the quality of a given corpus while preserving its vocabulary; the method used for bilingual lexicon extraction is then described and evaluated in Section 5. Section 6 is then devoted to a discussion, prior to the conclusion given in Section 7.

2 Experimental Materials

For the experiments reported here, several corpora were used: the parallel English-French *Europarl* corpus (Koehn, 2005), the TREC

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 644–652,
Beijing, August 2010

(http://trec.nist.gov/) *Associated Press* corpus (*AP*, English) and the corpora used in the multilingual track of CLEF (http://www.clef-campaign.org) which includes the *Los Angeles Times* (*LAT94*, English), *Glasgow Herald* (*GH95*, English), *Le Monde* (*MON94*, French), *SDA French 94* (*SDA94*, French) and *SDA French 95* (*SDA95*, French). In addition to these existing corpora, two monolingual corpora from the Wikipedia dump[1] were built. For English, all the articles below the root category *Society* with a depth less than 4[2] were retained. For French, all the articles with a depth less than 7 below the category *Société* are extracted. As a result, the English corpus *Wiki-En* consists of 367,918 documents and the French one *Wiki-Fr* consists of 378,297 documents.

The bilingual dictionary used in our experiments is constructed from an online dictionary. It consists of 33,372 distinct English words and 27,733 distinct French words, which constitutes 75,845 translation pairs. Standard preprocessing steps: tokenization, POS-tagging and lemmatization are performed on all the linguistic resources. We will directly work on lemmatized forms of content words (nouns, verbs, adjectives, adverbs).

3 Measuring Comparability

As far as we can tell, there are no practical measures with which we can judge the degree of comparability of a bilingual corpus. In this paper, we propose a comparability measure based on the expectation of finding the translation for each word in the corpus. The measure is light-weighted and does not depend on complex resources like the machine translation system. For convenience, the following discussions will be made in the context of the English-French comparable corpus.

3.1 The Comparability Measure

For the comparable corpus $\mathcal{C}$, if we consider the translation process from the English part $\mathcal{C}_e$ to the French part $\mathcal{C}_f$, a comparability measure M_{ef} can be defined on the basis of the expectation of finding, for each English word w_e in the vocabulary $\mathcal{C}_e^v$ of $\mathcal{C}_e$, its translation in the vocabulary $\mathcal{C}_f^v$ of $\mathcal{C}_f$. Let σ be a function indicating whether a translation from the translation set $\mathcal{T}_w$ of w is found in the vocabulary $\mathcal{C}^v$ of a corpus $\mathcal{C}$, i.e.:

$$\sigma(w, \mathcal{C}^v) = \begin{cases} 1 & \text{iff } \mathcal{T}_w \cap \mathcal{C}^v \neq \emptyset \\ 0 & \text{else} \end{cases}$$

M_{ef} is then defined as:

$$\begin{aligned} M_{ef}(\mathcal{C}_e, \mathcal{C}_f) &= \mathbb{E}(\sigma(w, \mathcal{C}_f^v) | w \in \mathcal{C}_e^v) \\ &= \sum_{w \in \mathcal{C}_e^v} \underbrace{\sigma(w, \mathcal{C}_f^v) \cdot Pr(w \in \mathcal{C}_e^v)}_{A_w} \\ &= \frac{|\mathcal{C}_e^v|}{|\mathcal{C}_e^v \cap \mathcal{D}_e^v|} \sum_{w \in \mathcal{C}_e^v \cap \mathcal{D}_e^v} A_w \end{aligned}$$

where $\mathcal{D}_e^v$ is the English part of a given, independent bilingual dictionary $\mathcal{D}$, and where the last equality is based on the fact that, the comparable corpus and the bilingual dictionary being independent of one another, the probability of finding the translation in $\mathcal{C}_f^v$ of a word w is the same for w is in $\mathcal{C}_e^v \cap \mathcal{D}_e^v$ and in $\mathcal{C}_e^v \backslash \mathcal{D}_e^v$[3]. Furthermore, the presence of common words suggests that one should rely on a presence/absence criterion rather than on the number of occurrences to avoid a bias towards common words. Given the natural language text, our evaluation will show that the simple presence/absence criterion can perform very well. This leads to $Pr(w \in \mathcal{C}_e^v) = 1/|\mathcal{C}_e^v|$, and finally to:

$$M_{ef}(\mathcal{C}_e, \mathcal{C}_f) = \frac{1}{|\mathcal{C}_e^v \cap \mathcal{D}_e^v|} \sum_{w \in \mathcal{C}_e^v \cap \mathcal{D}_e^v} \sigma(w, \mathcal{C}_f^v)$$

This formula shows that M_{ef} is actually the proportion of English words translated in the French part of the comparable corpus. Similarly, the counterpart of M_{ef}, M_{fe}, is defined as:

$$M_{fe}(\mathcal{C}_e, \mathcal{C}_f) = \frac{1}{|\mathcal{C}_f^v \cap \mathcal{D}_f^v|} \sum_{w \in \mathcal{C}_f^v \cap \mathcal{D}_f^v} \sigma(w, \mathcal{C}_e^v)$$

[1] The Wikipedia dump files can be downloaded at http://download.wikimedia.org. In this paper, we use the English dump file on July 13, 2009 and the French dump file on July 7, 2009.

[2] There are several cycles in the category tree of Wikipedia. It is thus necessary to define a threshold on the depth to make the iterative process feasible.

[3] The fact can be reliable only when a substantial part of the corpus vocabulary is covered by the dictionary. Fortunately, the constraint is satisfied in most applications where the common but not the specialized corpora like the medical corpora are involved.

and measures the proportion of French words in $\mathcal{C}_f^v$ translated in the English part of the comparable corpus. A symmetric version of these measures is obtained by considering the proportion of the words (both English and French) for which a translation can be found in the corpus:

$$M(\mathcal{C}_e, \mathcal{C}_f) = \frac{\sum_{w \in \mathcal{C}_e^v \cap \mathcal{D}_e^v} \sigma(w, \mathcal{C}_f^v) + \sum_{w \in \mathcal{C}_f^v \cap \mathcal{D}_f^v} \sigma(w, \mathcal{C}_e^v)}{|\mathcal{C}_e^v \cap \mathcal{D}_e^v| + |\mathcal{C}_f^v \cap \mathcal{D}_f^v|}$$

We now present an evaluation of these measures on artificial test corpora.

3.2 Validation

In order to test the comparability measures, we developed gold-standard comparability scores from the *Europarl* and *AP* corpora. We start from the parallel corpus, *Europarl*, of which we degrade the comparability by gradually importing some documents from either *Europarl* or *AP*. Three groups (G_a, G_b, G_c) of comparable corpora are built in this fashion. Each group consists of test corpora with a gold-standard comparability ranging, arbitrarily, from 0 to 1 and corresponding to the proportion of documents in "parallel" translation. The first group G_a is built from *Europarl* only. First, the *Europarl* corpus is split into 10 equal parts, leading to 10 parallel corpora (P_1, P_2, ..., P_{10}) with a gold-standard comparability arbitrarily set to 1. Then for each parallel corpus, e.g. P_i, we replace a certain proportion p of the English part with documents of the same size from another parallel corpus $P_j (j \neq i)$, producing the new corpus P_i' with less comparability which is the gold-standard comparability $1 - p$. For each P_i, as p increases, we obtain several comparable corpora with a decreasing gold-standard comparability score. All the P_i and their descendant corpora constitute the group G_a. The only difference between G_b and G_a is that, in G_b, the replacement in P_i is done with documents from the *AP* corpus and not from *Europarl*. In G_c, we start with 10 final, comparable corpora P_i' from G_a. These corpora have a gold-standard comparability of 0 in G_a, and of 1 in G_c. Then each P_i' is further degraded by replacing certain portions with documents from the *AP* corpus.

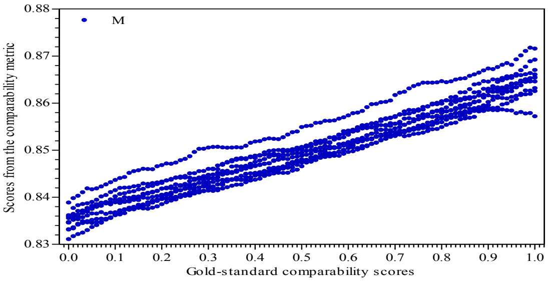


Figure 1: Evolution of M wrt gold-standard on the corpus group G_c (x-axis: gold-standard scores, y-axis: *M* scores)

We then computed, for each comparable corpus in each group, its comparability according to one of the comparability measures. Figure 1 plots the measure M for ten comparable corpora and their descendants from G_c, according to their gold-standard comparability scores. As one can note, the measure M is able to capture almost all the differences in comparability and is strongly correlated with the gold-standard. The correlation between the different measures and the gold-standard is finally computed with Pearson correlation coefficient. The results obtained are listed in Table 1. As one can note, M_{fe} performs worst among the three measures, the reason being that the process to construct G_b and G_c yields unbalanced bilingual corpora, the English section being larger than the French one. Translations of French words are still likely to be found in the English corpus, even though the corpora are not comparable. On all the 3 groups, M performs best and correlates very well with the gold standard, meaning that M was able to capture all the differences in comparability artificially introduced in the degradation process we have considered. This is the measure we will retain in the following parts.

	M_{ef}	M_{fe}	M
G_a	0.897	0.770	0.936
G_b	0.955	0.190	0.979
G_c	0.940	-0.595	0.960

Table 1: Correlation scores for the different comparability measures on the 3 groups of corpora

Having established a measure for the degree of comparability of bilingual corpora, we now turn to the problem of improving the quality of comparable corpora.

4 Improving Corpus Quality

We here try to improve the quality of a given corpus $\mathcal{C}$, which we will refer to as the *base corpus*, by extracting the highly comparable subpart $\mathcal{C}_H$ which is above a certain degree of comparability η (Step 1), and by enriching the lowly comparable part $\mathcal{C}_L$ with texts from other sources (Step 2). As we are interested in extracting information related to the vocabulary of the base corpus, we want the newly built corpus to contain a substantial part of the base corpus. This can be achieved by preserving in Step 1 as many documents from the base corpus as possible (e.g. by considering low values of η), and by using in step 2 sources close to the base corpus.

4.1 Step 1: Extracting $\mathcal{C}_H$

The strategy consisting of building all the possible sub-corpora of a given size from a given comparable corpora is not realistic as soon as the number of documents making up the corpora is larger than a few thousands. In such cases, better ways for extracting subparts have to be designed. The strategy we have adopted here aims at efficiently extracting a subpart of $\mathcal{C}$ above a certain degree of comparability and is based on the following property.

Property 1. *Let d_e^1 and d_e^2 (resp. d_f^1 and d_f^2) be two English (resp. French) documents from a bilingual corpus $\mathcal{C}$. We consider, as before, that the bilingual dictionary $\mathcal{D}$ is independent from $\mathcal{C}$. Let $(d_e^{1'}, d_f^{1'})$ be such that: $d_e^{1'} \subseteq d_e^1$, $d_f^{1'} \subseteq d_f^1$, which means $d_e^{1'}$ is a subpart of d_e^1 and $d_f^{1'}$ is a subpart of d_f^1.*
We assume:

$$\textit{(i)} \qquad \frac{|d_e^1 \cup d_e^2|}{|d_e^2|} = \frac{|d_f^1 \cup d_f^2|}{|d_f^2|}$$

$$\textit{(ii)} \quad M_{ef}(d_e^{1'}, d_f^1) \geq M_{ef}(d_e^2, d_f^2)$$

$$M_{fe}(d_e^1, d_f^{1'}) \geq M_{fe}(d_e^2, d_f^2)$$

Then:

$$M(d_e^2, d_f^2) \leq M(d_e^1 \cup d_e^2, d_f^1 \cup d_f^2)$$

Proof [sketch]: Let $B = (d_e^1 \cup d_e^2) \cap \mathcal{D}_e^v) \backslash (d_e^2 \cap \mathcal{D}_e^v)$. One can show, by exploiting condition (ii), that:

$$\sum_{w \in B} \sigma(w, d_f^1 \cup d_f^2) \geq |B| M_{ef}(d_e^2, d_f^2)$$

and similarly that:

$$\sum_{w \in d_e^2 \cap \mathcal{D}_e^v} \sigma(w, d_f^1 \cup d_f^2) \geq |d_e^2 \cap \mathcal{D}_e^v| M_{ef}(d_e^2, d_f^2)$$

Then exploiting condition (i), and the independence between the corpus and the dictionary, one arrives at:

$$\frac{\sum_{w \in (d_e^1 \cup d_e^2) \cap \mathcal{D}_e^v} \sigma(w, d_f^1 \cup d_f^2)}{|(d_e^1 \cup d_e^2) \cap \mathcal{D}_e^v| + |(d_f^1 \cup d_f^2) \cap \mathcal{D}_f^v|}$$
$$\geq \frac{|d_e^2 \cap \mathcal{D}_e^v| M_{ef}(d_e^2, d_f^2)}{|d_e^2 \cap \mathcal{D}_e^v| + |d_f^2 \cap \mathcal{D}_f^v|}$$

The same development on M_{fe} completes the proof. □

Property 1 shows that one can incrementally extract from a bilingual corpus a subpart with a guaranteed minimum degree of comparability η by iteratively adding new elements, provided (a) that the new elements have a degree of comparability of at least η and (b) that they are less comparable than the currently extracted subpart (conditions (ii)). This strategy is described in Algorithm 1. Since the degree of comparability is always above a certain threshold and since the new documents selected (d_e^2, d_f^2) are the most comparable among the remaining documents, condition (i) is likely to be satisfied, as this condition states that the increase in the vocabulary from the second documents to the union of the two is the same in both languages. Similarly, considering new elements by decreasing comparability scores is a necessary step for the satisfaction of condition (ii), which states that the current subpart should be uniformly more comparable than the element to be added. Hence, the conditions for property 1 to hold are met in Algorithm 1, which finally yields a corpus with a degree of comparability of at least η.

4.2 Step 2: Enriching $\mathcal{C}_L$

This step tries to absorb knowledge from other resources, which will be called *external corpus*,

```
Algorithm 1
Input:
    English document set $\mathcal{C}_e^d$ of $\mathcal{C}$
    French document set $\mathcal{C}_f^d$ of $\mathcal{C}$
    Threshold $\eta$
Output:
    $\mathcal{C}_H$, consisting of the English document set $\mathcal{S}_e$
    and the French document set $\mathcal{S}_f$
Initialize $\mathcal{S}_e = \emptyset, \mathcal{S}_f = \emptyset$, temp $= 0$;
repeat
    $(d_e, d_f) = \underset{d_e \in \mathcal{C}_e^d, d_f \in \mathcal{C}_f^d}{\arg\max} M(d_e, d_f)$;
    temp $= \max_{d_e \in \mathcal{C}_e^d, d_f \in \mathcal{C}_f^d} M(d_e, d_f)$;
    if temp $\geq \eta$ then
        Add $d_e$ into $\mathcal{S}_e$ and add $d_f$ into $\mathcal{S}_f$;
        $\mathcal{C}_e^d = \mathcal{C}_e^d \backslash d_e$, $\mathcal{C}_f^d = \mathcal{C}_f^d \backslash d_f$;
    end if
until $\mathcal{C}_e^d = \emptyset$ or $\mathcal{C}_f^d = \emptyset$ or temp $< \eta$
return $\mathcal{C}_H$;
```

to enrich the lowly comparable part $\mathcal{C}_L$ which is the left part in $\mathcal{C}$ during the creation of $\mathcal{C}_H$. One choice for obtaining the external corpus $\mathcal{C}_T$ is to fetch documents which are likely to be comparable from the Internet. In this case, we first extract representative words for each document in $\mathcal{C}_L$, translate them using the bilingual dictionary and retrieve associated documents via a search engine. An alternative approach is of course to use existing bilingual corpora. Once $\mathcal{C}_T$ has been constructed, the lowly comparable part $\mathcal{C}_L$ can be enriched in exactly the same way as in section 4.1: First, Algorithm 1 is used on the English part of $\mathcal{C}_L$ and the French part of $\mathcal{C}_T$ to get the high-quality document pairs. Then the French part of $\mathcal{C}_L$ is enriched with the English part of $\mathcal{C}_T$ by the same algorithm. All the high-quality document pairs are then added to $\mathcal{C}_H$ to constitute the final result.

4.3 Validation

We use here *GH95* and *SDA95* as the base corpus $\mathcal{C}^0$. In order to illustrate that the efficiency of the proposed algorithm is not confined to a specific external resource, we consider two external resources: (a) $\mathcal{C}_T^1$ made of *LAT94*, *MON94* and *SDA94*, and (b) $\mathcal{C}_T^2$ consisting of *Wiki-En* and *Wiki-Fr*. The number of documents in all the corpora after elimination of short documents (< 30 words) is listed in Table 2.

	$\mathcal{C}^0$	$\mathcal{C}_T^1$	$\mathcal{C}_T^2$
English	55,989	109,476	367,918
French	42,463	87,086	378,297

Table 2: The size of the corpora in the experiments

For the extraction of the highly comparable part $\mathcal{C}_H$ from the base corpus $\mathcal{C}^0$, we set η to 0.3 so as to extract a substantial subpart of $\mathcal{C}^0$. After this step, corresponding to Algorithm 1, we have 20,124 English-French document pairs in $\mathcal{C}_H$. The second step is to enrich the lowly comparable part $\mathcal{C}_L$ of the base corpus documents from the external resources $\mathcal{C}_T^1$ and $\mathcal{C}_T^2$. The final corpora we obtain consist of 46,996 document pairs for $\mathcal{C}^1$ (with $\mathcal{C}_T^1$) and of 54,402 document pairs for $\mathcal{C}^2$ (with $\mathcal{C}_T^2$), size similar to the one of $\mathcal{C}^0$. The proportion of documents (columns “D-e” and “D-f”), sentences (columns “S-e” and “S-f”) and vocabulary (columns “V-e” and “V-f”) of $\mathcal{C}^0$ found in $\mathcal{C}^1$ and $\mathcal{C}^2$ is given in Table 3. As one can note, the final corpora obtained through the method presented above preserve most of the information from the base corpus. Especially for the vocabulary, the final corpora cover nearly all the vocabulary of the base corpus. Considering the comparability scores, the comparability of $\mathcal{C}^1$ is 0.912 and the one of $\mathcal{C}^2$ is 0.916. Both of them are more comparable than the base corpus of which the comparability is 0.882.

From these results of the intrinsic evaluation, one can conclude that the strategy developed to improve the corpus quality while preserving most of its information is efficient: The corpora obtained here, $\mathcal{C}^1$ and $\mathcal{C}^2$, are more comparable than the base corpus $\mathcal{C}^0$ and preserve most of its information. We now turn to the problem of extracting bilingual lexicons from these corpora.

5 Bilingual Lexicon Extraction

Following standard practice in bilingual lexicon extraction from comparable corpora, we rely on the approach proposed by Fung and Yee (1998). In this approach, each word w is represented as a

	D-e	D-f	S-e	S-f	V-e	V-f
$\mathcal{C}^1$	0.669	0.698	0.821	0.805	0.937	0.981
$\mathcal{C}^2$	0.785	0.719	0.893	0.807	0.968	0.987

Table 3: Proportion of documents, sentences and vocabulary of $\mathcal{C}^0$ covered by the result corpora

context vector consisting of the weight $a(w^c)$ of each context word w^c, the context being extracted from a window running through the corpus. Once context vectors for English and French words have been constructed, a general bilingual dictionary $\mathcal{D}$ can be used to bridge them by accumulating the contributions from words that are translation of each other. Standard similarity measures, as the cosine or the Jaccard coefficient, can then be applied to compute the similarity between vectors. For example, the cosine leads to:

$$s_c(w_e, w_f) = \frac{\sum_{(w_e^c, w_f^c)\in\mathcal{D}} a(w_e^c)a(w_f^c)}{\|\overrightarrow{w_e}\| \cdot \|\overrightarrow{w_f}\|} \quad (1)$$

5.1 Using Algorithm 1 pseudo-Alignments

The process we have defined in the previous section to improve the quality of a given corpus while preserving its vocabulary makes use of highly comparable document pairs, and thus provides some loose alignments between the two corpora. One can thus try to leverage the above approach to bilingual lexicon extraction by re-weighting $s_c(w_e, w_f)$ by a quantity which is large if w_e and w_f appear in many document pairs with a high comparability score, and small otherwise. In this section, we can not use the alignments in algorithm 1 directly because the alignments in the comparable corpus should not be 1 to 1 and we did not try to find the precise 1 to 1 alignments in algorithm 1.

Let η be the threshold used in algorithm 1 to construct the improved corpus and let $\phi(d_e, d_f)$ be defined as:

$$\phi(d_e, d_f) = \begin{cases} 1 & \text{iff } M(d_e, d_f) \geq \eta \\ 0 & \text{else} \end{cases}$$

Let $\mathcal{H}_e$ (resp. $\mathcal{H}_f$) be the set of documents containing word w_e (resp. w_f). We define the joint probability of w_e and w_f as being proportional to the number of comparable document pairs they belong to, where two documents are comparable if their comparability score is above η, that is:

$$p(w_e, w_f) \propto \sum_{d_e \in \mathcal{H}_e, d_f \in \mathcal{H}_f} \phi(d_e, d_f)$$

The marginal probability $p(w_e)$ can then be written as:

$$\begin{aligned} p(w_e) &\propto \sum_{w_f \in \mathcal{C}_f^v} p(w_e, w_f) \\ &\propto \sum_{d_e \in \mathcal{H}_e} \sum_{d_f \in \mathcal{C}_f^d} |d_f| \cdot \phi(d_e, d_f) \end{aligned}$$

Assuming that all d_f in $\mathcal{C}_f^d$ have roughly the same vocabulary size and all d_e have the same number of comparable counterparts in $\mathcal{C}_f^d$, then the marginal probability can be simplified as: $p(w_e) \propto |\mathcal{H}_e|$. By resorting to the exponential of the point-wise mutual information, one finally obtains the following weight:

$$\begin{aligned} \pi(w_e, w_f) &= \frac{p(w_e, w_f)}{p(w_e) \cdot p(w_f)} \\ &\propto \frac{1}{|\mathcal{H}_e| \cdot |\mathcal{H}_f|} \sum_{d_e \in \mathcal{H}_e, d_f \in \mathcal{H}_f} \phi(d_e, d_f) \end{aligned}$$

which has the desired property: It is large if the two words appear in comparable document pairs more often than chance would predict, and small otherwise. We thus obtain the revised similarity score for w_e and w_f:

$$s_{cr}(w_e, w_f) = s_c(w_e, w_f) \cdot \pi(w_e, w_f) \quad (2)$$

5.2 Validation

In order to measure the performance of the bilingual lexicon extraction method presented above, we divided the original dictionary into 2 parts: 10% of the English words (3,338 words) together with their translations are randomly chosen and used as the evaluation set, the remaining words (30,034 words) being used to compute context vectors and similarity between them. In this study, the weight $a(w^c)$ used in the context vectors (see above) are taken to be the tf-idf score of w^c: $a(w^c)$ = tf-idf(w^c). English words not present in $\mathcal{C}_e^v$ or with no translation in $\mathcal{C}_f^v$ are excluded from the evaluation set. For each English word in the evaluation set, all the French words in $\mathcal{C}_f^v$ are then ranked according to their similarity with the English word (using either equation 1 or 2). To evaluate the quality of the lexicons extracted, we first retain for each English word its N first translations, and then measure the precision of the lists obtained, which amounts in this case to the proportion of lists containing the correct translation (in case of multiple translations, a list is deemed to contain the correct translation as soon as one of the possible translations is present). This evaluation procedure has been used in previous work (e.g. (Gaussier et al., 2004)) and is now standard for the evaluation of lexicons extracted from comparable corpora. In this study, N is set to 20. Furthermore, several studies have shown that it is easier to find the correct translations for frequent words than for infrequent ones (Pekar et al., 2006). To take this fact into account, we distinguished different frequency ranges to assess the validity of our approach for all frequency ranges. Words with frequency less than 100 are defined as low-frequency words (W_L), whereas words with frequency larger than 400 are high-frequency words (W_H), and words with frequency in between are medium-frequency words (W_M).

We then tested the standard method based on the cosine similarity (equation 1) on the corpora $\mathcal{C}^0$, $\mathcal{C}_H$, $\mathcal{C}'_H$, $\mathcal{C}^1$ and $\mathcal{C}^2$. The results obtained are displayed in Table 4, and correspond to columns 2-6. They show that the standard approach performs significantly better on the improved corpora $\mathcal{C}^1$/$\mathcal{C}^2$ than on the base corpus $\mathcal{C}^0$. The overall precision is increased by 5.3% on $\mathcal{C}^1$ (corresponding to a relative increase of 26%) and 9.5% on $\mathcal{C}^2$ (corresponding to a relative increase of 51%), even though the low-frequency words, which dominate the overall precision, account for a higher proportion in $\mathcal{C}^1$ (61.3%) and $\mathcal{C}^2$ (61.3%) than in $\mathcal{C}^0$ (56.2%). For the medium and high frequency words, the precision is increased by over 11% on $\mathcal{C}^1$ and 16% on $\mathcal{C}^2$. As pointed out in other studies, the performance for the low-frequency words is usually bad due to the lack of context information. This explains the relatively small improvement obtained here (only 2.2% on $\mathcal{C}^1$ and 6.7% on $\mathcal{C}^2$). It should also be noticed that the performance of the standard approach is better on $\mathcal{C}^2$ than on $\mathcal{C}^1$, which may be due to the fact that $\mathcal{C}^2$ is slightly larger than $\mathcal{C}^1$ and thus provides more information or to the actual content of these corpora. Lastly, if we consider the results on the corpus $\mathcal{C}_H$ which is produced by only choosing the highly comparable part from $\mathcal{C}^0$, the overall precision is increased by only 1.9%, which might come from the fact that the size of $\mathcal{C}_H$ is less than half the size of $\mathcal{C}^0$. We also notice the better results on $\mathcal{C}_H$ than on $\mathcal{C}'_H$ of the same size which consists of randomly choosing documents from $\mathcal{C}^0$.

The results obtained with the refined approach making use of the comparable document pairs found in the improved corpus (equation 2) are also displayed in Table 4 (columns "$\mathcal{C}^1$ new" and "$\mathcal{C}^2$ new"). From these results, one can see that the overall precision is further improved by 2.0% on $\mathcal{C}^1$ and 2.3% on $\mathcal{C}^2$, compared with the standard approach. For all the low, medium and high-frequency words, the precision has been improved, which demonstrates that the information obtained through the corpus enrichment process contributes to improve the quality of the extracted bilingual lexicons. Compared with the original base corpus $\mathcal{C}^0$, the overall improvement of the precision on both $\mathcal{C}^1$ and $\mathcal{C}^2$ with the refined approach is significant and important (respectively corresponding to a relative improvement of 35% and 62%), which also demonstrates that the efficiency of the refined approach is not confined to a specific external corpus.

6 Discussion

It is in a way useless to deploy bilingual lexicon extraction techniques if translation equivalents are not present in the corpus. This simple fact is at the basis of our approach which consists in constructing comparable corpora close to the original corpus and which are more likely to contain translation equivalents as they have a guaranteed degree of comparability. The pseudo-alignments identified in the construction process are then used to leverage state-of-the-art bilingual lexicon extraction methods. This approach to bilingual lexicon extraction from comparable corpora radically differs, to our knowledge, from previous approaches

	$\mathcal{C}^0$	$\mathcal{C}_H$	$\mathcal{C}'_H$	$\mathcal{C}^1$	$\mathcal{C}^2$	$\mathcal{C}^1$ new	$> \mathcal{C}^1, > \mathcal{C}^0$	$\mathcal{C}^2$ new	$> \mathcal{C}^2, > \mathcal{C}^0$
W_L	0.114	0.144	0.125	0.136	0.181	0.156	2.0%, 4.2%	0.205	2.4%, 9.1%
W_M	0.233	0.313	0.270	0.345	0.401	0.369	2.4%, 3.6%	0.433	3.2%, 20.0%
W_H	0.417	0.456	0.377	0.568	0.633	0.581	1.3%, 16.4%	0.643	1.0%, 22.6%
All	**0.205**	**0.224**	**0.189**	**0.258**	**0.310**	**0.278**	**2.0%, 7.3%**	**0.333**	**2.3%, 12.8%**

Table 4: Precision of the different approaches on different corpora

which are mainly variants of the standard method proposed in (Fung and Yee, 1998) and (Rapp, 1999). For example, the method developed in (Déjean et al., 2002) and (Chiao and Zweigenbaum, 2002) involves a representation of dictionary entries with context vectors onto which new words are mapped. Pekar et al. (2006) smooth the context vectors used in the standard approach in order to better deal with low frequency words. A nice geometric interpretation of these processes is proposed in (Gaussier et al., 2004), which furthermore introduces variants based on Fisher kernels, Canonical Correlation Analysis and a combination of them, leading to an improvement of the F1-score of 2% (from 0.14 to 0.16) when considering the top 20 candidates. In contrast, the approach we have developed yields an improvement of 7% (from 0.13 to 0.20) of the F-1 score on $\mathcal{C}^2$, again considering the top 20 candidates. More important, however, is the fact that the approach we have developed can be used in conjunction with any existing bilingual extraction method, as the strategies for improving the corpus quality and the re-weighting formula (equation 2) are general. We will assess in the future whether substantial gains are also attained with other methods.

Some studies have tried to extract subparts of comparable corpora to complement existing parallel corpora. Munteanu (2004) thus developed a maximum entropy classifier aiming at extracting those sentence pairs which can be deemed parallel. The step for choosing similar document pairs in this work resembles some of our steps. However their work focuses on high quality and specific documents pairs, as opposed to the entire corpus of guaranteed quality we want to build. In this latter case, the cross-interaction between documents impacts the overall comparability score, and new methods, as the one we have introduced, need to be proposed. Similarly, Munteanu and Marcu (2006) propose a method to extract subsentential fragments from non-parallel corpora. Again, the targeted elements are very specific (parallel sentences or sub-sentences) and limited, and the focus is put on a few sentences which can be considered parallel. As already mentioned, we rather focus here on building a new corpus which preserves most of the information in the original corpus. The construction process we have presented is theoretically justified and allows one to preserve ca. 95% of the original vocabulary.

7 Conclusion

We have first introduced in this paper a comparability measure based on the expectation of finding translation word pairs in the corpus. We have then designed a strategy to construct an improved comparable corpus by (a) extracting a subpart of the original corpus with a guaranteed comparability level, and (b) by completing the remaining subpart with external resources, in our case other existing bilingual corpora. We have then shown how the information obtained during the construction process could be used to improve state-of-the-art bilingual lexicon extraction methods. We have furthermore assessed the various steps of our approach and shown: (a) that the comparability measure we introduced captures variations in the degree of comparability between corpora, (b) that the construction process we introduced leads to an improved corpus preserving most of the original vocabulary, and (c) that the use of pseudo-alignments through simple re-weighting yields bilingual lexicons of higher quality.

Acknowledgements

This work was supported by the French National Research Agency grant ANR-08-CORD-009.

References

Ballesteros, Lisa and W. Bruce Croft. 1997. Phrasal translation and query expansion techniques for cross-language information retrieval. In *Proceedings of the 20th ACM SIGIR*, pages 84–91, Philadelphia, Pennsylvania, USA.

Chen, Stanley F. 1993. Aligning sentences in bilingual corpora using lexical information. In *Proceedings of the 31st Annual Conference of the Association for Computational Linguistics*, pages 9–16, Columbus, Ohio, USA.

Chiao, Yun-Chuang and Pierre Zweigenbaum. 2002. Looking for candidate translational equivalents in specialized, comparable corpora. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–7, Taipei, Taiwan.

Déjean, Hervé, Eric Gaussier, and Fatia Sadat. 2002. An approach based on multilingual thesauri and model combination for bilingual lexicon extraction. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–7, Taipei, Taiwan.

Fung, Pascale and Kathleen McKeown. 1997. Finding terminology translations from non-parallel corpora. In *Proceedings of the 5th Annual Workshop on Very Large Corpora*, pages 192–202, Hong Kong.

Fung, Pascale and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *Proceedings of the 17th international conference on Computational linguistics*, pages 414–420, Montreal, Quebec, Canada.

Gaussier, E., J.-M. Renders, I. Matveeva, C. Goutte, and H. Déjean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 526–533, Barcelona, Spain.

Kay, Martin and Martin Röscheisen. 1993. Text-translation alignment. *Computational Linguistics*, 19(1):121–142.

Koehn, Philipp. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit 2005*.

Melamed, I. Dan. 1997a. A portable algorithm for mapping bitext correspondence. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 305–312, Madrid, Spain.

Melamed, I. Dan. 1997b. A word-to-word model of translational equivalence. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 490–497, Madrid, Spain.

Morin, Emmanuel, Béatrice Daille, Koichi Takeuchi, and Kyo Kageura. 2007. Bilingual terminology mining - using brain, not brawn comparable corpora. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 664–671, Prague, Czech Republic.

Munteanu, Dragos Stefan and Daniel Marcu. 2006. Extracting parallel sub-sentential fragments from non-parallel corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 81–88, Sydney, Australia.

Munteanu, Dragos Stefan, Alexander Fraser, and Daniel Marcu. 2004. Improved machine translation performance via parallel sentence extraction from comparable corpora. In *Proceedings of the HLT-NAACL 2004*, pages 265–272, Boston, MA., USA.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Pekar, Viktor, Ruslan Mitkov, Dimitar Blagoev, and Andrea Mulloni. 2006. Finding translations for low-frequency words in comparable corpora. *Machine Translation*, 20(4):247–266.

Rapp, Reinhard. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 519–526, College Park, Maryland, USA.

Robitaille, Xavier, Yasuhiro Sasaki, Masatsugu Tonoike, Satoshi Sato, and Takehito Utsuro. 2006. Compiling French-Japanese terminologies from the web. In *Proceedings of the 11st Conference of the European Chapter of the Association for Computational Linguistics*, pages 225–232, Trento, Italy.

Yu, Kun and Junichi Tsujii. 2009. Extracting bilingual dictionary from comparable corpora with dependency heterogeneity. In *Proceedings of HLT-NAACL 2009*, pages 121–124, Boulder, Colorado, USA.

Structure-Aware Review Mining and Summarization

Fangtao Li[1], Chao Han[1], Minlie Huang[1], Xiaoyan Zhu[1],
Ying-Ju Xia[2], Shu Zhang[2] and Hao Yu[2]
[1]State Key Laboratory of Intelligent Technology and Systems,
[1]Tsinghua National Laboratory for Information Science and Technology,
[1]Department of Computer Science and Technology, Tsinghua University
[2]Fujitsu Research and Development Center
fangtao06@gmail.com; zxy_dcs@tsinghua.edu.cn

Abstract

In this paper, we focus on object feature[1] based review summarization. Different from most of previous work with linguistic rules or statistical methods, we formulate the review mining task as a joint structure tagging problem. We propose a new machine learning framework based on Conditional Random Fields (CRFs). It can employ rich features to jointly extract positive opinions, negative opinions and object features for review sentences. The linguistic structure can be naturally integrated into model representation. Besides linear-chain structure, we also investigate conjunction structure and syntactic tree structure in this framework. Through extensive experiments on movie review and product review data sets, we show that structure-aware models outperform many state-of-the-art approaches to review mining.

1 Introduction

With the rapid expansion of e-commerce, people are more likely to express their opinions and hands-on experiences on products or services they have purchased. These reviews are important for both business organizations and personal costumers. Companies can decide on their strategies for marketing and products improvement. Customers can make a better decision when purchasing products or services. Unfortunately, reading through all customer reviews is difficult, especially for popular items, the number of reviews can be up to hundreds or even thousands. Therefore, it is necessary to provide coherent and concise summaries for these reviews.

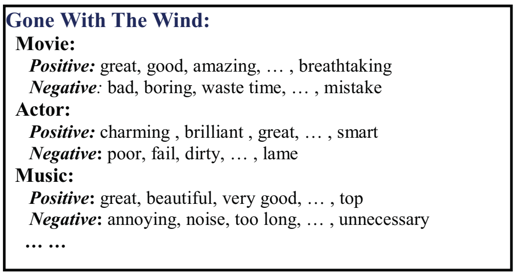

Gone With The Wind:
Movie:
Positive: great, good, amazing, … , breathtaking
Negative: bad, boring, waste time, … , mistake
Actor:
Positive: charming , brilliant , great, … , smart
Negative: poor, fail, dirty, … , lame
Music:
Positive: great, beautiful, very good, … , top
Negative: annoying, noise, too long, … , unnecessary
… …

Figure 1. Feature based Review Summarization

Inspired by previous work (Hu and Liu, 2004; Jin and Ho, 2009), we aim to provide object feature based review summarization. Figure 1 shows a summary example for movie "Gone with the wind". The object (movie) features, such as "movie", "actor", with their corresponding positive opinions and negative opinions, are listed in a structured way. The opinions are ranked by their frequencies. This provides a concise view for reviews. To accomplish this goal, we need to do three tasks: 1), extract all the object features and opinions; 2), determine the sentiment polarities for opinions; 3), for each object feature, determine the relevant opinions, i.e. object feature-opinion pairs.

For the first two tasks, most previous studies employ linguistic rules or statistical methods (Hu and Liu, 2004; Popescu and Etzioni 2005). They mainly use unsupervised learning methods, which lack an effective way to address infrequent object features and opinions. They are also hard to incorporate rich overlapping features.

[1] Note that there are two meanings for word "feature". We use "object feature" to represent the target entity, which the opinion expressed on, and use "feature" as the input for machine learning methods.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 653–661,
Beijing, August 2010

Actually, there are many useful features, which have not been fully exploited for review mining. Meanwhile, most of previous methods extract object features, opinions, and determine the polarities for opinions separately. In fact, the object features, positive opinions and negative opinions correlate with each other.

In this paper, we formulate the first two tasks, i.e. object feature, opinion extraction and opinion polarity detection, as a joint structure tagging problem, and propose a new machine learning framework based on Conditional Random Fields (CRFs). For each sentence in reviews, we employ CRFs to jointly extract object features, positive opinions and negative opinions, which appear in the review sentence. This framework can naturally encode the linguistic structure. Besides the neighbor context with linear-chain CRFs, we propose to use Skip-chain CRFs and Tree CRFs to utilize the conjunction structure and syntactic tree structure. We also propose a new unified model, Skip-Tree CRFs to integrate these structures. Here, "structure-aware" refers to the output structure, which model the relationship among output labels. This is significantly different from the previous input structure methods, which consider the linguistic structure as heuristic rules (Ding and Liu, 2007) or input features for classification (Wilson et al. 2009). Our proposed framework has the following advantages: First, it can employ rich features for review mining. We will analyze the effect of features for review mining in this framework. Second, the framework can utilize the relationship among object features, positive opinions and negative opinions. It jointly extracts these three types of expressions in a unified way. Third, the linguistic structure information can be naturally integrated into model representation, which provides more semantic dependency for output labels. Through extensive experiments on movie review and product review, we show our proposed framework is effective for review mining.

The rest of this paper is organized as follows: In Section 2, we review related work. We describe our structure aware review mining methods in Section 3. Section 4 demonstrates the process of summary generation. In Section 5, we present and discuss the experiment results. Section 6 is the conclusion and future work.

2 Related Work

Object feature based review summary has been studied in several papers. Zhuang et al. (2006) summarized movie reviews by extracting object feature keywords and opinion keywords. Object feature-opinion pairs were identified by using a dependency grammar graph. However, it used a manually annotated list of keywords to recognize movie features and opinions, and thus the system capability is limited. Hu and Liu (2004) proposed a statistical approach to capture object features using association rules. They only considered adjective as opinions, and the polarities of opinions are recognized with WordNet expansion to manually selected opinion seeds. Popescu and Etzioni (2005) proposed a relaxation labeling approach to utilize linguistic rules for opinion polarity detection. However, most of these studies focus on unsupervised methods, which are hard to integrate various features. Some studies (Breck et al. 2007; Wilson et al, 2009; Kobayashi et al. 2007) have used classification based methods to integrate various features. But these methods separately extract object features and opinions, which ignore the correlation among output labels, i.e. object features and opinions. Qiu et al. (2009) exploit the relations of opinions and object features by adding some linguistic rules. However, they didn't care the opinion polarity. Our framework can not only employ various features, but also exploit the correlations among the three types of expressions, i.e. object features, positive opinions, and negative opinions, in a unified framework. Recently, Jin and Ho (2009) propose to use Lexicalized HMM for review mining. Lexicalized HMM is a variant of HMM. It is a generative model, which is hard to integrate rich, overlapping features. It may encounter sparse data problem, especially when simultaneously integrating multiple features. Our framework is based on Conditional Random Fields (CRFs). CRFs is a discriminative model, which can easily integrate various features.

These are some studies on opinion mining with Conditional Random Fields. For example, with CRFs, Zhao et al (2008) and McDonald et al. (2007) performed sentiment classification in sentence and document level; Breck et al (2007) identified opinion expressions from newswire documents; Choi et al. (2005) determined opinion holders to opinions also from newswire data. None of previous work focuses on jointly extracting object features, positive opinions and negative opinions simultaneously from review data. More importantly, we also show how to encode the linguistic structure, such as conjunction structure and syntactic tree structure, into model representation in our framework. This is significantly different from most of previous studies, which consider the structure information as heuristic rules (Hu and Liu, 2004) or input features (Wilson et al. 2009).

Recently, there are some studies on joint sentiment/topic extraction (Mei et al. 2007; Titov and McDonald, 2008; Snyder and Barzilay, 2007). These methods represent reviews as several coarse-grained topics, which can be considered as clusters of object features. They are hard to indentify the low-frequency object features and opinions. While in this paper, we will extract all the present object features and corresponding opinions with their polarities. Besides, the joint sentiment/topic methods are mainly based on review document for topic extraction. In our framework, we focus on sentence-level review extraction.

3 Structure Aware Review Mining

3.1 Problem Definition

To produce review summaries, we need to first finish two tasks: identifying object features, opinions, and determining the polarities for opinions. In this paper, we formulate these two tasks as a joint structure tagging problem. We first describe some related definitions:

Definition (Object Feature): is defined as whole target expression that the subjective expressions have been commented on. Object features can be products, services or their elements and properties, such as "character", "movie", "director" for movie review, and "battery", "battery life", "memory card" for product review.

Definition (Review Opinion): is defined as the whole subjective expression on object features. For example, in sentence "The camera is easy to use", "easy to use" is a review opinion. "opinion" is used for short.

Definition (Opinion Polarity): is defined as the sentiment category for review opinion. In this paper, we consider two types of polarities: positive opinion and negative opinion. For example, "easy to use" belongs to positive opinion.

For our review mining task, we need to represent three types of expressions: object features, positive opinions, and negative opinions. These expressions may be words, or whole phrases. We use BIO encoding for tag representation, where the non-opinion and neutral opinion words are represented as "O". With Negation (N), which is only one word, such as "not", "don't", as an independent tag, there are totally 8 tags, as shown in Table 1. The following is an example to denote the tags:

The*/O* camera*/FB* comes*/O* with*/O* a*/O* pitiful*/CB* 32mb*/FB* compact*/FI* flash*/FI* card*/FI ./O*

FB	Feature Beginning	CB	Negative Beginning
FI	Feature Inside	CI	Negative Inside
PB	Positive Beginning	N	Negation Word
PI	Positive Inside	O	Other

Table 1. Basic Tag Set for Review Mining

3.2 Structure Aware Model

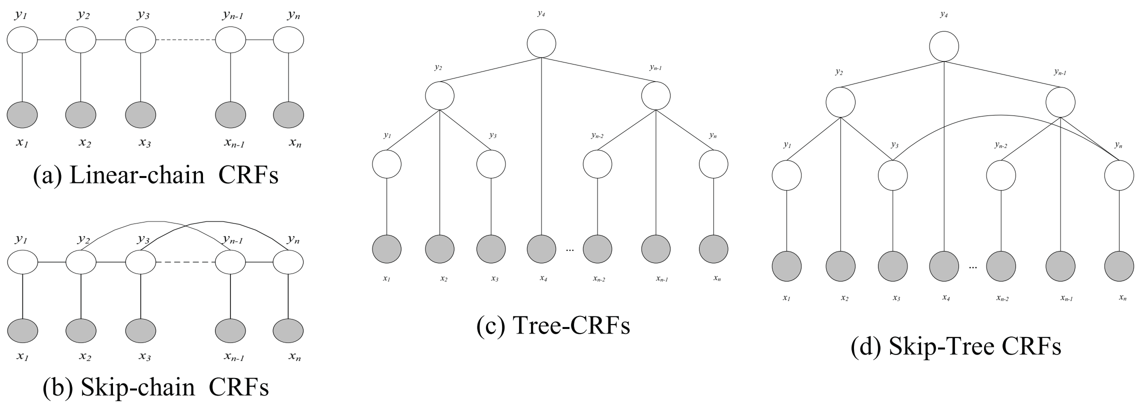
In this section, we describe how to encode different linguistic structure into model representation based on our CRFs framework.

3.2.1 Using Linear CRFs.

For each sentence in a review, our task is to extract all the object features, positive opinions and negative opinions. This task can be modeled as a classification problem. Traditional classification tools, e.g. Maximum Entropy model (Berger et al, 1996), can be employed, where each word or phrase will be treated as an instance. However, they independently consider each word or phrase, and ignore the dependency relationship among them.

Actually, the context information plays an important role for review mining. For example, given two continuous words with same part of speech, if the previous word is a positive opinion, the next word is more likely a positive opinion. Another example is that if the previous word is an adjective, and it is an opinion, the next noun word is more likely an object feature.

To this end, we formulate the review mining task as a joint structure tagging problem, and propose a general framework based on Conditional Random Fields (CRFs) (Lafferty et al., 2001) which are able to model the dependencies


y_1 y_2 y_3 y_{n-1} y_n
x_1 x_2 x_3 x_{n-1} x_n
y_4 y_2 y_{n-1} y_1 y_3 y_{n-2} y_n
x_1 x_2 x_3 x_4 x_{n-2} x_{n-1} x_n
...

(a) Linear-chain CRFs

(b) Skip-chain CRFs

(c) Tree-CRFs

(d) Skip-Tree CRFs

Figure 2 CRFs models

between nodes. (See Section 3.2.5 for more about CRFs)

In this section, we propose to use linear-chain CRFs to model the sequential dependencies between continuous words, as discussed above. It views each word in the sentence as a node, and adjacent nodes are connected by an edge. The graphical representation is shown in Figure 2(a). Linear CRFs can make use of dependency relationship among adjacent words.

3.2.2 Leveraging Conjunction Structure

We observe that the conjunctions play important roles on review mining: If the words or phrases are connected by conjunction "and", they mostly belong to the same opinion polarity. If the words or phrases are connected by conjunction "but", they mostly belong to different opinion polarity, as reported in (Hatzivassiloglou and McKeown, 1997; Ding and Liu, 2007). For example, "This phone has a very cool and useful feature – the speakerphone", if we only detect "cool", it is hard to determine its opinion polarity. But if we see "cool" is connected with "useful" by conjunction "and", we can easily acquire the polarity of "cool" as positive. This conjunction structure not only helps to determine the opinions, but also helps to recognize object features. For example, "I like the special effects and music in this movie", with word "music" and conjunction "and", we can easily detect that "special effects" as an object feature.

To model the long distance dependency with conjunctions, we use Skip-chain CRFs model to detect object features and opinions. The graphical representation of a Skip-chain CRFs, given in Figure 2(b), consists of two types of edges: linear-edge (y_{t-1} to y_t) and skip-edge (y_i to y_j). The linear-edge is described as linear CRFs. The skip-edge is imported as follows:

We first identify the conjunctions in the review sentence, with a collected conjunction set, including "and", "but", "or", "however", "although" etc. For each conjunction, we extract its connected two text sequences. The nearest two words with same part of speech from the two text sequences are connected with the skip-edge. Here, we just consider the noun, adjective, and adverb. For example, in "good pictures and beautiful music", there are two skip-edges: one connects two adjective words "good" and "beautiful"; the other connects two nouns "pictures" and "music". We also employ the general sentiment lexicons, SentiWordNet (Esuli and Sebastiani, 2006), to connect opinions. Two nearest opinion words, detected by sentiment lexicon, from two sequences, will also be connected by skip-edge. If the nearest distance exceeds the threshold, this skip edge will be discarded. Here, we consider the threshold as nine.

Skip-chain CRFs improve the performance of review mining, because it naturally encodes the conjunction structure into model representation with skip-edges.

3.2.3 Leveraging Syntactic Tree Structure

Besides the conjunction structure, the syntactic tree structure also helps for review mining. The tree denotes the syntactic relationship among words. In a syntactic dependency representation, each node is a surface word. For example, the corresponding dependency tree (Klein and Manning, 2003) for the sentence, "I really like this long movie", is shown in Figure 3.

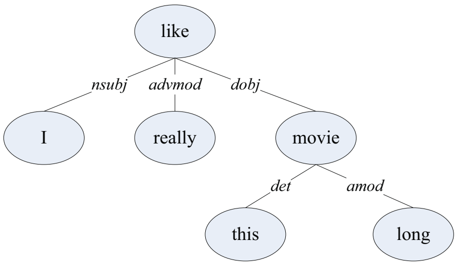


Figure 3. Syntactic Dependency Tree Representation

In linear-chain structure and skip-chain structure, “like” and “movie” have no direct edge, but in syntactic tree, “movie” is directly connected with “like”, and their relationship “dobj” is also included, which shows “movie” is an objective of “like”. It can provide deeper syntactic dependencies for object features, positive opinions and negative opinions. Therefore, it is important to consider the syntactic structure in the review mining task.

In this section, we propose to use Tree CRFs to model the syntactic tree structure for review mining. The representation of a Tree CRFs is shown in Figure 2(c). The syntactic tree structure is encoded into our model representation. Each node is corresponding to a word in the dependency tree. The edge is corresponding to dependency tree edge. Tree CRFs can make use of dependency relationship in syntactic tree structure to boost the performance.

3.2.4 Integrating Conjunction Structure and Syntactic Tree Structure

Conjunction structure provides the semantic relations correlated with conjunctions. Syntactic tree structure provides dependency relation in the syntactic tree. They represent different semantic dependencies. It is interesting to consider these two dependencies in a unified model. We propose Skip-Tree CRFs, to combine these two structure information. The graphical representation of a Skip-Tree CRFs, given in Figure 2(d), consists of two types of edges: tree edges and conjunction skip-edges. We hope to simultaneously model the dependency in conjunction structure and syntactic tree structure.

We also notice that there is a relationship “conj” in syntactic dependency tree. However, we find that it only connects two head words for a few coordinating conjunction, such as “and", “or", “but”. Our designed conjunction skip-edge provides more information for joint structure tagging. We analyze more conjunctions to connect not only two head words, but also the words with same part of speech. We also connect the words with sentiment lexicon. We will show that the skip-tree CRFs, which combine the two structures, is effective in the experiment section.

3.2.5 Conditional Random Fields

A CRFs is an undirected graphical model G of the conditional distribution $P(Y|X)$. Y are the random variables over the labels of the nodes that are globally conditioned on *X*, which are the random variables of the observations. The conditional probability is defined as:

$$\mathrm{P}(Y\,|X) = \frac{1}{Z(X)}\,exp\left(\sum_{e\in E,i} \gamma_i t_i(e, Y|e, X) + \sum_{v\in V,i} \mu_i s_i(v, Y|v, X)\right)$$

where *Z(x)* is the normalization factor, s_i is the state function on node, t_i is the transition functions on edge, and $\,_{,}r_i$ and μ_i are parameters to estimate (Sutton and McCallum, 2006).

Inference and Parameter Estimation. For Linear CRFs, dynamic programming is used to compute the maximum a posteriori (MAP) of *Y* given *X*. For more complicated graphs with cycles, we employ Tree Re-Parameterization (TRP) algorithm (Wainwright et al. 2001) for approximate inference.

Given the training Data $D = \{x^{(i)}, y^{(i)}\}_{i=1}^{n}$, the parameter estimation is to determine the parameters based on maximizing the log-likelihood $L_\theta = \sum_{i=1}^{n} log\, p(y^{(i)}|x^{(i)})$. In Linear CRFs model, dynamic programming and L-BFGS algorithm can be used to optimize objective function L_θ, while for complicated CRFs, TRP is used instead to calculate the marginal probability.

3.3 Feature Space

In this section, we describe the features used in the learning methods. All the features are listed in Figure 4. Word features include the word’s token, lemma, and part of speech. The adjacent words’ information is considered. We detect whether the negation words appear in the previous four words as a binary feature. We also detect whether this word is the superlative form, such as “best”, and comparative form, such as “better”, as binary features. Two types of dictionaries are employed. We use WordNet to acquire the synonyms and antonyms for each word. SentiWordNet (Esuli and Sebastiani, 2006) is used to acquire the prior polarity for each word. We use the words with positive or negative score

Word Feature:
- Word token
- Word lemma
- Word part of speech
- Previous word token, lemma, part of speech
- Next word token, lemma, part of speech
- Negation word appears in previous 4 words
- Is superlative degree
- Is comparative degree

Dictionary Feature
- WordNet Synonym
- WordNet Antonym
- SentiWordNet Prior Polarity

Sentence Feature
- Num of positive words in SentiWordNet
- Num of negative words in SentiWordNet
- Num of Negation word

Syntactic Features:
- Parent word
- Parent SentiWordnet Prior Polarity
- In subject
- In copular
- In object

Edge Feature
- Conjunction word
- Syntactic relationship

Figure 4. Features for learning Methods

above a threshold (0.6). Sentence Feature provides sentence level information. It includes the count of positive words and negative words, which are detected by SentiWordNet. We also incorporate the count of negation words as a feature. There are some syntactic features from dependency tree. Parent word and its polarity are considered. We also detect if the word is subject, object or copular. For edge features, the conjunction words are incorporated as corresponding skip-edge features. The syntactic relationship is considered as a feature for corresponding tree-edge. For classification and linear CRFs models, we just add this edge features as general features.

4 Review Summary Generation

After extracting the object features and opinions, we need to extract the relevant opinions for each feature. In this paper, we identify the nearest opinion word/phrase for each object feature as object feature-opinion pair, which is widely used in previous work (Hu and Liu, 2004; Jin and Ho, 2009). The review summary is generated as a list of structured object feature-opinion pairs, as shown in Figure 1.

5 Experiment

5.1 Experiment setup

Data Set: For our structure tagging task, we need to know the labels for all the words in reviews. In this paper, we manually annotate two types of these review data sets. One is movie review, which contains five movies with totally 500 reviews. The other is product review, which contains four products with totally 601 reviews. We need to label all object features, positive opinions, negative opinions, and the object feature-opinion pairs for all sentences. Each sentence is labeled by two annotators. The conflict is checked by the third person. Finally, we acquire 2207 sentences for movie review and 2533 sentences for product review. For each type, including movie and product, the data set is divided into five parts. We select four parts as training data, and the fifth part as testing data.

Evaluation Metric:

Precision, Recall and F measure are used to test our results, as Jin and Ho (2009).

5.2 Baselines

First word	Second Word	Third Word
JJ	NN or NNS	Anything
RB, RBR or RBS	JJ	NN or NNS
JJ	JJ	NN or NNS
NN or NNS	JJ	Not NN or NNS

Table 2. Rules in rule based method

Rule based Method:

The rule based method is used in Jin and Ho (2009), which is motivated by (Hu and Liu, 2004; Turney, 2002). The employed rules are shown in Table 2. The matching adjective is identified as opinion, and matching nouns are extracted as object features. To determine the polarities of the opinions, 25 positive adjectives and 25 negative adjectives are used as seeds, and then expanded by searching synonyms and antonyms in WordNet. The polarity of a word is detected by checking the collected lists.

Lexicon based Method:

The object features and opinions extraction is same as rule based method. The general sentiment lexicon SentiWordNet is employed to detect the polarity for each word.

Lexicalized HMM:

The object features and opinions are identified by Lexicalized HMM (L-HMM), as Jin and Ho (2009). L-HMM is a variant of HMM. It has two observations. The current tag is not only related

	Methods	Object Features			Positive Opinions			Negative Opinions			Overall		
		P(%)	*R(%)*	*F(%)*	*P(%)*	*R(%)*	*F(%)*	*P(%)*	*R(%)*	*F(%)*	*P(%)*	*R(%)*	*F(%)*
Movie Review	Rule	41.2	32.3	36.2	**82.9**	31.1	45.3	23.5	13.7	17.3	49.2	25.7	33.8
	Lexicon	41.2	32.3	36.2	64.0	38.1	47.8	19.6	6.8	10.2	41.6	25.8	31.8
	L-HMM	**88.0**	52.6	65.9	82.1	49.6	61.9	65.9	**41.1**	**50.6**	78.7	47.8	59.5
	MaxEnt	83.4	75.1	79.1	82.2	**65.0**	**72.6**	74.1	29.5	42.2	**79.9**	56.5	66.2
	Linear CRFs	81.8	**78.4**	**80.1**	79.1	63.9	70.7	**75.8**	32.2	45.2	79.0	**58.2**	**67.0**
Product Review	Rule	53.5	35.6	42.8	74.4	22.5	34.6	17.1	8.9	11.7	48.3	22.3	30.6
	Lexicon	53.5	35.6	42.8	48.9	29.7	40.0	14.7	3.7	5.9	39.1	23.0	29.0
	L-HMM	83.9	48.7	61.6	**90.3**	56.8	69.8	47.2	25.2	32.9	73.8	43.6	54.8
	MaxEnt	83.4	55.1	66.4	82.2	65.0	72.6	64.1	30.0	40.4	76.6	49.9	60.4
	Linear CRFs	**91.1**	**56.3**	**69.6**	88.7	**70.4**	**78.5**	**67.7**	**32.6**	**44.0**	**82.5**	**53.1**	**64.6**

Table 3. Comparison Results with Baselines

(the learning methods only employ word token and part of speech as features).

	Methods	Object Features			Positive Opinions			Negative Opinions			Overall		
		P(%)	*R(%)*	*F(%)*	*P(%)*	*R(%)*	*F(%)*	*P(%)*	*R(%)*	*F(%)*	*P(%)*	*R(%)*	*F(%)*
Movie Review	MaxEnt	82.8	76.6	79.6	80.3	67.8	73.5	**82.8**	36.3	50.5	81.9	60.2	69.4
	Linear CRFs	83.5	75.4	79.2	77.8	71.4	74.5	70.9	53.4	60.9	77.4	66.8	71.7
	Skip CRFs	83.9	78.7	81.2	81.8	73.4	77.4	75.2	62.3	68.2	80.3	71.5	75.7
	Tree CRFs	84.1	79.0	81.5	**82.7**	75.4	78.9	76.7	61.0	67.9	81.2	72.2	76.2
	SkipTreeCRFs	**85.5**	**82.0**	**83.7**	82.3	**80.0**	**81.1**	80.2	**66.4**	**72.7**	**82.6**	**76.2**	**79.3**
Product Review	MaxEnt	80.0	70.8	75.1	85.6	65.7	74.3	65.1	37.8	47.8	76.9	58.1	66.2
	Linear CRFs	84.0	72.9	78.1	86.7	72.0	78.6	60.4	49.6	54.5	77.0	64.8	70.4
	Skip CRFs	84.8	73.5	78.7	87.8	74.5	80.6	73.1	50.4	59.6	81.2	66.1	73.2
	Tree CRFs	83.0	72.7	77.5	86.6	73.4	79.4	64.3	54.8	59.2	78.0	67.0	72.1
	SkipTreeCRFs	**87.1**	**74.1**	**80.1**	**91.8**	**76.7**	**83.6**	**81.1**	**57.0**	**67.0**	**86.6**	**69.3**	**77.0**

Table 4. Comparative experiments with all features

with the previous tag, but also correlates with previous observations. They use word token and part of speech as two features.

Classification based Method:

We also formulate the review mining as a classification task. Each word is considered as an instance. Maximum Entropy (MaxEnt) is used in this paper.

5.3 Experiment results

Since Lexicalized HMM employ word token and part of speech as features (Jin and Ho, 2009), we first conduct comparative experiments with these two features for learning methods. Table 3 shows the results. The rule based method is a little better than lexicon based method. SentiWordNet is designed for general opinion mining, which may be not suitable for domain specific review mining task. For rule based method, the seeds are selected in the review domain, which is more suitable for domain specific task. However, both methods achieve low performance. This because that they only employ simple linguistic rules to extract object features and opinions, which is not effective for infrequent cases and phrase cases. Lexicalized HMM is an extension of HMM. It uses word token and part of speech as two observations. The current tag is not only related with the previous tag, but also correlates with previous two observations. Lexicalized HMM can employ dependency relationship among adjacent words. However, it doesn't achieve the expected result. This is because that Lexicalized HMM is a generative model, which is hard to incorporate rich overlapping features. Even Lexicalized HMM uses linear interpolation smoothing technique. The data sparsity problem seriously hurt the performance. There are many sentences with zero probability. MaxEnt classifier is a discrimitive model, which can incorporate various features. However, it independently classifies each word, and ignores the dependency among successive words. The linear CRFs model achieves best performances for movie review, and product review in overall F-score. This is because that, in our joint structure tagging framework, linear CRFs can employs the global structure to make use of the adjacent dependency relation, and easily incorporate various features to boost the performance.

We also conduct the comparative experiments with all features. From Table 4, we can see that linear CRFs, which consider the chain structure,

	Object Features			Positive Opinions			Negative Opinions			Overall		
	P(%)	*R(%)*	*F(%)*	*P(%)*	*R(%)*	*F(%)*	*P(%)*	*R(%)*	*F(%)*	*P(%)*	*R(%)*	*F(%)*
Basic	83.8	79.2	81.4	79.5	71.0	75.0	76.1	37.0	49.8	79.8	62.4	70.0
Basic +Word Feature	84.0	81.4	82.7	79.2	75.6	77.4	78.9	48.6	60.2	80.7	68.6	74.1
Basic +Dictionary	80.5	76.6	78.5	**82.7**	76.3	79.4	76.5	60.3	67.4	80.0	71.0	75.2
Basic +Sentence	82.5	75.6	78.9	80.4	75.4	77.8	**84.0**	46.7	60.0	82.3	65.9	73.2
Basic +Syntactic	84.5	70.8	77.0	79.6	73.9	76.7	79.5	47.9	59.8	81.2	64.2	71.7
Basic + Edge	84.1	80.1	82.1	79.5	75.4	77.4	82.4	47.9	60.6	82.0	67.8	74.2
All Features	**85.5**	**82.0**	**83.7**	82.3	**80.0**	**81.1**	80.2	**66.4**	**72.7**	**82.6**	**76.2**	**79.3**

Table 5. Feature Evaluations with Skip Tree CRFs (movie)

still achieve better results than MaxEnt classifier method. Skip-chain CRFs model the conjunction structure in the sentence. We can see that the Skip-chain CRFs achieve better results than linear CRFs. This shows that conjunction structure is really important for review mining. For example "although this camera takes great pictures, it is extremely fragile.", "fragile" is not correctly classified by MaxEnt and Linear CRFs. But the Skip-chain CRFs can correctly classify "fragile" as negative opinion, with conjunction "although", and the skip edge between "great" and "fragile". Tree CRFs encode the syntactic tree structure into model representation. Compared with linear-CRFs, the performances are improved for most of expression identification tasks, except for a little decline for product object feature, which may be because that the tags "FB" and "FI" are out of order when transferring to tree structure. These are no significant difference between Skip-Chain CRFs and Tree CRFs. Conjunction structure and syntactic structure represent the semantic dependency from different views. When integrating these two types of dependencies, the Skip-Tree CRFs achieve better overall results than both Skip-Chain CRFs and Tree CRFs.

Table 5 shows the movie review result for Skip Tree model for different types of features. The basic feature only employs word token as feature set. Other features are defined as shown in Figure 4. By adding different features, we find that they all achieve overall improvements than basic feature. The dictionary features are the most important features, especially for positive opinion and negative opinion identification, which shows the importance of prior word's sentiment. Word features also play important roles: Part of speech is reported useful in several papers (such as Jin and Ho, 2009); the superlative and comparative forms are good indicators for opinion words. Syntactic features acquire limited improvement in this experiment. They may overlap with CRF based structure model. We also find that sentence level features contribute to the review mining task. Edge feature is also important. It makes the skip edge and tree edge with the semantic representation. When combing all the features, the result is significantly improved compared with any single feature set, which shows that it is crucial to integrate various features for review mining.

A review summary example, generated by our methods, is shown in Figure 1.

6 Conclusion

In this paper, we formulate the review mining task as a joint structure tagging problem. A new framework based on Conditional Random Fields is proposed. The framework can employ rich features to simultaneously extract object features, positive opinions and negative opinions. With this framework, we investigate the chain structure, conjunction structure and syntactic tree structure for review mining. A new unified model, called skip tree CRFs, is proposed for review mining. Through extensive experiments, we show that our proposed framework is effective. It outperforms many state-of-the-art methods.

In future work, we will improve the object feature-opinion pair detection with other learning methods. We also want to cluster the related object features to provide more concise review summary.

Acknowledgement

This work was partly supported by Chinese NSF grant No.60973104 and No. 60803075, a grand from Fujitsu Research Center, and a grant from the International Development Research Center, Ottawa, Canada. We also thank the anonymous reviewers, Qiang Yang, Lei Zhang and Qiu Long for their valuable comments.

References

A. Berger and Vincent Della Pietra and Stephen A. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. Computational Linguistics.

E. Breck, Y. Choi, and C. Cardie. 2007. Identifying expressions of opinion in context. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI).

Y. Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying Sources of Opinions with Conditional Random Fields and Extraction Patterns. In Proceedings of HLT-EMNLP.

X. Ding and Bing Liu. 2007. The Utility of Linguistic Rules in Opinion Mining. In Proceedings of SIGIR.

A. Esuli and Fabrizio Sebastiani. 2006. SENTIWORDNET: A Publicly Available Lexical Resource for Opinion Mining. In Proceedings of LREC.

V. Hatzivassiloglou and K. McKeown. 1997. Predicting the semantic orientation of adjectives. Proceedings of the Joint ACL/EACL Conference.

M. Hu and B. Liu. 2004. Mining and Summarizing Customer Reviews. Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04).

W. Jin, H.H. Ho. 2009. A novel lexicalized HMM-based learning framework for web opinion mining. Proceedings of the 26th Annual International Conference on Machine Learning (ICML 2009).

J. Lafferty, A. McCallum, F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. 18th International Conf. on Machine Learning (ICML).

D. Klein and Christopher D. Manning. 2003. Fast Exact Inference with a Factored Model for Natural Language Parsing. In Advances in Neural Information Processing Systems 15 (NIPS 2002),

N. Kobayashi, K. Inui, and Y. Matsumoto. 2007. Opinion Mining from Web documents: Extraction and Structurization. Journal of Japanese society for artificial intelligence.

R. McDonald, K. Hannan, T. Neylon, M. Wells, and J. Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. Proceedings of the Association for Computational Linguistics (ACL).

Q. Mei, X. Ling, M. Wondra, H. Su, and C. Zhai. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In Proceedings of the 16th international conference on World Wide Web.

A. Popescu and O. Etzioni. 2005. Extracting Product Features and Opinions from Reviews. Proceedings of 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP'05), 339-346.

G. Qiu, B. Liu, J. Bu and C. Chen. 2009. Expanding Domain Sentiment Lexicon through Double Propagation, International Joint Conference on Artificial Intelligence (IJCAI-09).

B. Snyder and R. Barzilay. 2007. Multiple Aspect Ranking using the Good Grief Algorithm", In Proc. of NAACL

C. Sutton, A. McCallum. 2006. An Introduction to Conditional Random Fields for Relational Learning. In "Introduction to Statistical Relational Learning". Edited by Lise Getoor and Ben Taskar. MIT Press.

I. Titov and R. McDonald. 2008. A Joint Model of Text and Aspect Ratings for Sentiment Summarization.In Proceeding of the Association for Computational Linguistics (ACL).

P. D. Turney. 2002. Thumbs up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. Proceedings of Association for Computational Linguistics (ACL'02).

M. Wainwright, T. Jaakkola, and A. Willsky. 2001. Tree-based reparameterization for approximate estimation on graphs with cycles. In Proceedings of Advances in Neural Information Processing Systems (NIPS'2001). pp. 1001-1008.

T. Wilson, Janyce Wiebe, and Paul Hoffmann 2009. Recognizing Contextual Polarity: an exploration of features for phrase-level sentiment analysis. Computational Linguistics 35(3).

J. Zhao, Kang Liu, Gen Wang. 2008. Adding Redundant Features for CRFs-based Sentence Sentiment Classification. In Proceedings of EMNLP.

L. Zhuang, Feng Jing and Xiaoyan Zhu. 2006. Movie Review Mining and Summarization. In Proceedings of CIKM.

Adaptive Development Data Selection for Log-linear Model in Statistical Machine Translation

Mu Li
Microsoft Research Asia
muli@microsoft.com

Yinggong Zhao[*]
Nanjing University
zhaoyg@nlp.nju.edu.cn

Dongdong Zhang
Microsoft Research Asia
dozhang@microsoft.com

Ming Zhou
Microsoft Research Asia
mingzhou@microsoft.com

Abstract

This paper addresses the problem of dynamic model parameter selection for log-linear model based statistical machine translation (SMT) systems. In this work, we propose a principled method for this task by transforming it to a test data dependent development set selection problem. We present two algorithms for automatic development set construction, and evaluated our method on several NIST data sets for the Chinese-English translation task. Experimental results show that our method can effectively adapt log-linear model parameters to different test data, and consistently achieves good translation performance compared with conventional methods that use a fixed model parameter setting across different data sets.

1 Introduction

In recent years, log-linear model (Och and Ney, 2002) has been a mainstream method to formulate statistical models for machine translation. Using this formulation, various kinds of relevant properties and data statistics used in the translation process, either on the monolingual-side or on the bilingual-side, are encoded and used as real-valued *feature functions*, thus it provides an effective mathematical framework to accommodate a large variety of SMT formalisms with different computational linguistic motivations.

Formally, in a log-linear SMT model, given a source sentence f, we are to find a translation e^* with largest posterior probability among all possible translations:

$$e^* = \operatorname*{argmax}_e \Pr(e|f)$$

and the posterior probability distribution $\Pr(e|f)$ is directly approximated by a log-linear formulation:

$$\Pr(e|f) = p_{\boldsymbol{\lambda}}(e|f) = \frac{\exp(\sum_{m=1}^{M} \lambda_m h_m(e,f))}{\sum_{e'} \exp(\sum_{m=1}^{M} \lambda_m h_m(e',f))} \quad (1)$$

in which h_m's are feature functions and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_M)$ are model parameters (*feature weights*).

For a successful practical log-linear SMT model, it is usually a combined result of the several efforts:

- Construction of well-motivated SMT models
- Accurate estimation of feature functions
- Appropriate scaling of log-linear model features (feature weight tuning).

In this paper, we focus on the last mentioned issue – parameter tuning for log-linear model. In general, log-linear model parameters are optimized on a held-out development data set. Using this method, similarly to many machine learning tasks, the model parameters are solely tuned based on the development data, and the optimality of obtained model on unseen test data relies on the assumption that both development and test data observe identical probabilistic distribution,

[*]This work was done while the author was visiting Microsoft Research Asia.

which often does not hold for real-world data. The goal of this paper is to investigate novel methods for test data dependent model parameter selection. We begin with discussing the principle of parameter learning for log-linear SMT models, and explain the rationale of task transformation from parameter selection to development data selection. We describe two algorithms for automatic development set construction, and evaluated our method on several NIST MT evaluation data sets. Experimental results show that our method can effectively adapt log-linear model parameters to different test data and achieves consistent good translation performance compared with conventional methods that use a group of fixed model parameters across different data sets.

2 Model Learning for SMT with Log-linear Models

Model learning refers to the task to estimate a group of suitable log-linear model parameters $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_M)$ for use in Equation 1, which is often formulated as an optimization problem that finds the parameters maximizing certain *goodness* of the translations generated by the learnt model on a development corpus D. The goodness can be measured with either the translations' likelihood or specific machine translation evaluation metrics such as TER or BLEU.

More specifically, let e^* be the most probable translation of D with respect to model parameters $\boldsymbol{\lambda}$, and $E(e^*, \boldsymbol{\lambda}, D)$ be a score function indicating the goodness of translation e^*, then a parameter estimation algorithm will try to find the $\boldsymbol{\lambda}$ which satisfies:

$$\boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}}{\operatorname{argmax}}\, E(e^*, \boldsymbol{\lambda}, D) \qquad (2)$$

Note when the goodness scoring function $E(\cdot)$ is specified, the parameter learning criterion in Equation 2 indicates that the derivation of model parameters $\boldsymbol{\lambda}^*$ only depends on development data D, and does not require any knowledge of test data T. The underlying rationale for this rule is that if the test data T observes the same distribution as D, $\boldsymbol{\lambda}^*$ will be optimal for both of them.

On the other side, however, when there are mismatches between development and test data, the translation performance on test data will be suboptimal, which is very common for real-world data. Due to the difference between data sets, generally there is no such $\boldsymbol{\lambda}^*$ that is optimal for multiple data sets at the same time. Table 1 shows some empirical evidences when two data sets are mutually used as development and test data. In this setting, we used a hierarchical phrase based decoder and 2 years' evaluation data of NIST Chinese-to-English machine translation task (for the year 2008 only the newswire subset was used because we want to limit both data sets within the same domain to show that data mismatch also exists even if there is no domain difference), and report results using BLEU scores. Model parameters were tuned using the MERT algorithm (Och, 2003) optimized for BLEU metric.

Dev data	MT05	MT08-nw
MT05	**0.402**	0.306
MT08-nw	0.372	**0.343**

Table 1: Translation performance of cross development/test on two NIST evaluation data sets.

In our work, we present a solution to this problem by using test data dependent model parameters for test data translation. As discussed above, since model parameters are solely determined by development data D, selection of log-linear model parameters is basically equivalent to selecting a set of development data D.

However, automatic development data selection in current SMT research remains a relatively open issue. Manual selection based on human experience and observation is still a common practice.

3 Adaptive Model Parameter Selection

An important heuristic behind manual development data selection is to use the dataset which is as similar to test set as possible in order to work around the data mismatch problem to maximal extent. There are also empirical evidences supporting this heuristics. For instance, it is generally perceived that data set MT03 is more similar to MT05, while MT06-nw is closer to MT08-nw. Table 2 shows experimental results using model parameters induced from MT03 and MT06-nw as development sets with the same settings as in Table 1. As expected, MT06-nw is far more suitable than MT03 as the development data for MT08-nw; yet for test set MT05, the situation is just the opposite.

Dev data	MT05	MT08-nw
MT03	**0.397**	0.306
MT06-nw	0.381	**0.337**

Table 2: Translation performance on different test sets of using different development sets.

In this work, this heuristic is further exploited for automatic development data selection when there is no prior knowledge of the test data available. In the following discussion, we assume the availability of a set of candidate source sentences together with translation references that are qualified for the log-linear model parameter learning task. Let D_F be the full candidate set, given a test set T, the task of selecting a set of development data which can optimize the translation quality on T can be transformed to searching for a suitable subset of D_F which is most similar to T:

$$D^* = \operatorname*{argmax}_{D \subseteq D_F} \mathrm{Sim}(D, T)$$

To achieve this goal, we need to address the following key issues:

- How to define and compute $\mathrm{Sim}(D, T)$, the similarity between different data sets;
- How to extract development data sets from a full candidate set for unseen test data.

3.1 Dataset Similarity

Computing document similarity is a classical task in many research areas such as information retrieval and document classification. However, typical methods for computing document similarity may not be suitable for our purpose. The reasons are two-fold:

1. The sizes of both development and test data are small in usual circumstances, and using similarity measures such as cosine or dice coefficient based on term vectors will suffer from severe data sparseness problems. As a result, the obtained similarity measure will not be statistically reliable.
2. More importantly, what we care about here is not the surface string similarity. Instead, we need a method to measure how similar two data sets are from the view of a log-linear SMT model.

Next we start with discussing the similarity between sentences. Given a source sentence f, we denote its possible translation space with $\mathcal{H}(f)$. In a log-linear SMT model, every translation $e \in \mathcal{H}(f)$ is essentially a feature vector $\boldsymbol{h}(e) = (h_1, \ldots, h_M)$. Accordingly, the similarity between two sentences f_1 and f_2 should be defined on the feature space of the model in use. Let $\boldsymbol{V}(f) = \{\boldsymbol{h}(e) : e \in \mathcal{H}(f)\}$ be the set of feature vectors for all translations in $\mathcal{H}(f)$, we have

$$\mathrm{Sim}(f_1, f_2) = \mathrm{Sim}\Big(\boldsymbol{V}(f_1), \boldsymbol{V}(f_2)\Big) \quad (3)$$

Because it is not practical to compute Equation 3 directly by enumerating all translations in $\mathcal{H}(f_1)$ and $\mathcal{H}(f_2)$ due to the huge search space in SMT tasks, we need to resort to some approximations. A viable solution to this is that if we can use a single feature vector $\tilde{\boldsymbol{h}}(f)$ to represent $\boldsymbol{V}(f)$, then Equation 3 can be simply computed using existing vector similarity measures.

One reasonable method to derive $\tilde{\boldsymbol{h}}(f)$ is to use a feature vector based on the *average* principle – each dimension of the vector is set to the expectation of its corresponding feature value over all translations:

$$\tilde{\boldsymbol{h}}(f) = \sum_{e \in \mathcal{H}(f)} P(e|f)\boldsymbol{h}(e) \quad (4)$$

An alternative and much simpler way to compute $\tilde{\boldsymbol{h}}(f)$ is to employ the *max* principle in which we just use the feature vector of the best translation in $\mathcal{H}(f)$:

$$\tilde{\boldsymbol{h}}(f) = \boldsymbol{h}(e^*) \quad (5)$$

where $e^* = \operatorname{argmax}_e P(e|f)$.

Note that in both Equation 4 and Equation 5 we make use of e's posterior probability $P(e|f)$.

Since the true distribution is unknown, a pre-learnt model $\mathcal{M}$ has to be used to assign approximate probabilities to translations, which indicates that the obtained similarity depends on a specific model. As a convention, we use $\mathrm{Sim}_{\mathcal{M}}(f_1, f_2)$ to denote the similarity between f_1 and f_2 based on $\mathcal{M}$, and call $\mathcal{M}$ the reference model of the computed similarity. To avoid unexpected bias caused by a single reference model, multiple reference models can be simultaneously used, and the similarity is defined to be the maximum of all model-dependent similarity values:

$$\mathrm{Sim}(f_1, f_2) = \max_{\mathcal{M}} \mathrm{Sim}_{\mathcal{M}}(f_1, f_2) \qquad (6)$$

where $\mathcal{M}$ belongs to $\{\mathcal{M}_1, \ldots, \mathcal{M}_n\}$, which is the set of reference models under consideration.

To generalize this method to data set level, we compute the vector $\tilde{\boldsymbol{h}}(S)$ for a data set $S = (f_1, \ldots, f_{|S|})$ as follows:

$$\tilde{\boldsymbol{h}}(S) = \sum_{i=1}^{|S|} \tilde{\boldsymbol{h}}(f_i) \qquad (7)$$

3.2 Development Sets Pre-construction

In the following, we sketch a method for automatically building a set of development data based on the full candidate set D_F before seeing any test data.

Theoretically, a subset of D_F containing randomly sampled sentences from D_F will not meet our requirement well because it is very probable that it will observe a distribution similar to D_F. What we expect is that the pre-built development sets can approximate as many as possible typical data distributions that can be estimated from subsets of D_F. Our solution is based on the assumption that D_F can be depicted by some mixture models, hence we can use classical clustering methods such as k-means to partition D_F into subsets with different distributions.

Let S_F be the set of extracted development data from D_F. The construction of S_{D_F} proceeds as following:

1. Train a log-linear model $\mathcal{M}_F$ using D_F as development data;
2. Compute a feature vector $\tilde{\boldsymbol{h}}(d)$[1] for each sentence $d \in D_F$ using $\mathcal{M}_F$ as reference model;
3. Cluster sentences in D_F using $\tilde{\boldsymbol{h}}(d)/|d|$ as feature vectors;
4. Add obtained sentence clusters to S_{D_F} as candidate development sets.

In the third step, since the feature vector $\tilde{\boldsymbol{h}}(d)$ is defined at sentence level, it is averaged by the number of words in d so that it is irrelevant to the length of a sentence. Considering the outputs of unsupervised data clustering methods are usually sensitive to initial conditions, we include in S_{D_F} sentence clusters based on different initialization configurations to remove related random effects. An initialization configuration for sentence clustering in our work includes starting point for each cluster and total number of clusters. In fact, the inclusion of more sentence clusters increases the diversity of the resulted S_{D_F} as well.

At decoding time, when a test set T is presented, we compute the similarity between T and each development set $D \in S_{D_F}$, and choose the one with largest similarity score as the development set for T:

$$D^* = \operatorname*{argmax}_{D \in S_{D_F}} \mathrm{Sim}(T, D) \qquad (8)$$

When a single reference model is used to compute $\mathrm{Sim}(T, D)$, $\mathcal{M}_F$ is a natural choice. In the multi-model setting as shown in Equation 6, models learnt from the development sets in S_{D_F} can serve this purpose.

Note in this method model learning is not required for every new test set because the model parameters for each development set in S_{D_F} can also be pre-learnt and ready to be used for decoding.

3.3 Dynamic Development Set Construction

In the previous method, test data T is only involved in the process of choosing a development set from a list of candidates but not in process of development set construction. Next we present a

[1] Throughout this paper, a development sentence d generally refers to the source part of it if there is no extra explanation.

method for building a development set on demand based on test data T.

Let $D_F = (d_1, \ldots, d_n)$ be the data set containing all candidate sentences for development data selection. The method is iterative process in which development data and learnt model are alternatively updated. Detailed steps are illustrated as follows:

1. Let $i = 0$, $D_0 = D_F$;
2. Train a model $\mathcal{M}_i$ based on D_i;
3. For each $d_k \in D_F$, compute the similarity score $\mathrm{Sim}_{\mathcal{M}_i}(T, d_k)$ between T and d_k based on model $\mathcal{M}_i$;
4. Select top n candidate sentences with highest similarity scores from D_F to form D_{i+1};
5. Repeat step 2 to step 4 until the similarity between T and latest selected development data converges (the increase in similarity measure is less than a specified threshold compared to last round) or the specified iteration limit is reached.

In step 4, D_{i+1} is greedily extracted from D_F, and there is no guarantee that $\mathrm{Sim}_{\mathcal{M}_i}(T, D_{i+1})$ will increase or decrease after a new sentence is added to D_{i+1}. Thereby the number of selected sentences n needs to be empirically determined. If n is too small, neither the selected data nor the learnt model parameters will be statistically reliable; while if n is too large, we may have to include some sentences that are not suitable for test data in the development data, and miss the opportunity to extract the most desirable development set.

One drawback of this method is the relatively high computational cost because it requires multiple parameter training passes when any test set is presented to the system for translation.

4 Experiments

4.1 Data

Experiments were conducted on the data sets used for NIST Chinese-English machine translation evaluation tasks. MT03 and MT06 data sets, which contain 919 and 1,664 sentences respectively, were used for development data in various settings. MT04, MT05 and MT08 data sets were used for test purpose. In some settings, we also used a test set MT0x, which containing 1,000 sentences randomly sampled from the above 3 data sets. All the translation performance results were measured in terms of case-insensitive BLEU scores.

For all experiments, all parallel corpora available to the constrained track of NIST 2008 Chinese-English MT evaluation task were used for translation model training, which consist of around 5.1M bilingual sentence pairs. GIZA++ was used for word alignment in both directions, which was further refined with the intersec-diag-grow heuristics.

We used a 5-gram language model which was trained from the Xinhua portion of English Gigaword corpus version 3.0 from LDC and the English part of parallel corpora.

4.2 Machine Translation System

We used an in-house implementation of the hierarchical phrase-based decoder as described in Chiang (2005). In addtion to the standard features used in Chiang (2005), we also used a lexicon feature indicating how many word paris in the translation found in a conventional Chinese-English lexicon. Phrasal rules were extracted from all the parallel data, but hierarchical rules were only extracted from the FBIS part of the parallel data which contains around 128,000 sentence pairs. For all the development data, feature weights of the decoder were tuned using the MERT algorithm (Och, 2003).

4.3 Results of Development Data Pre-construction

In the following we first present some overall results using the method of development data pre-construction, then dive into more detailed settings of the experiments.

Table 3 shows the results using 3 different data sets for log-linear model parameter tuning. Elements in the first column indicate the data sets used for parameter tuning, and other columns contain evaluation results on different test sets. In the

Tuning set	MT04	MT05	MT08	MT0x
MT03	0.399 / 0.392	0.395 / 0.390	0.241 / 0.258	0.319 / 0.322
MT06	0.381 / 0.388	0.382 / 0.391	0.275 / 0.283	0.343 / 0.342
MT03+MT06	0.391 / 0.401	0.392 / 0.397	0.265 / 0.281	0.336 / 0.345
Oracle cluster	0.401	0.398	0.293	0.345
Self-training	0.406	0.402	0.298	0.351

Table 3: Translation performance using different methods and data sets for parameter tuning.

third row of the table, MT03+MT06 means combining the data sets of MT03 and MT06 together to form a larger tuning set. The first number in each cell denotes the BLEU score using the tuning set as standard development set D, and the second for using the tuning set as a candidate set D_F.

For all experiment settings in the table, we used cosine value between feature vectors to measure similarity between data sets, and feature vectors were computed according to Equation 5 and Equation 7 using a reference model which is trained on the corresponding candidate set D_F as development set.[2] We adopted the k-means algorithm for data clustering with the number of clusters iterating from 2 to 5. In each iteration, we ran 4 passes of clustering using different initial values. Therefore, in total there are 56 sentence clusters generated in each S_{D_F}.[3]

From the table it can be seen that given the same set of sentences (MT03, MT06 and MT03+MT06), when they are used as the candidate set D_F for the development set preconstruction method, the translation performance is generally better than when they are just used as development sets as a whole. Using MT03 data set as D_F is an exception: there is slight performance drop on test sets MT04 and MT05, but it also helps reduce the performance *see-saw* problem on different test sets as shown in Table 1. Meanwhile, in the other two settings of D_F, we observed significant BLEU score increase on all test sets but MT0x (on which the performance almost kept unchanged). In addition, the fact that using MT03+MT06 as D_F achieves best (or almost best) performance on all test sets implies that it should be a better choice to include as diverse data as possible in D_F.

We also appended two oracle BLEU numbers for each test set in Table 3 for reference. One is denoted with *oracle cluster*, which is the highest possible BLEU that can be achieved on the test set when the development set must be chosen from the sentence clusters in $S_{MT03+MT06}$. The other is labeled as *self-training*, which is the BLEU score that can be obtained when the test data itself is used as development data. This number can serve as actual performance upper bound on the test set.

Next we investigated the impact of using different ways to compute feature vectors presented in Section 3.1. We re-ran some previous experiments on test sets MT04, MT05 and MT08 using MT03+MT06 as D_F. Most settings were kept unchanged except that the feature vector of each sentence was computed according to Equation 4. A 20-best translation list was used to approximate $\mathcal{H}(f)$. The results are shown in Table 4.

Test set	average	max
MT04	0.397	0.401
MT05	0.393	0.397
MT08	0.286	0.281

Table 4: Translation performance when using averaged feature values for similarity computation.

The numbers in the second column are based on Equation 4. Numbers based on Equation 5 are also listed in the third column for comparison. In all the experiment settings we did not observe consistent or significant advantage when using Equation 4 over using Equation 5. Since Equation 5

[2] For example, in all the experiments in the row of MT03 as D_F, we use the same reference model trained with MT03 as development set.

[3] Sometimes some clusters are empty or contain too few sentences, so the actual number may be smaller.

is much simpler, it is a good decision to use it in practice. So did we conduct all following experiments based on Equation 5.

We are also interested in the correlation between two measures: the similarity between development and test data and the actual translation performance on test data.

First we would like to echo the motivating experiment presented in Section 3. Table 5 shows the similarity between the data sets used in the experiment with $\mathcal{M}_{\mathrm{MT03+MT06}}$ as reference model. Obviously the results in Table 2 and Table 5 fit each other very well.

Dev data	MT05	MT08-nw
MT03	**0.99988**	0.99012
MT06-nw	0.99004	**0.99728**

Table 5: Similarity between NIST data sets.

Figure 1 shows the results of a set of more comprehensive experiments on MT05 data set concerning the similarity between development and test sets.

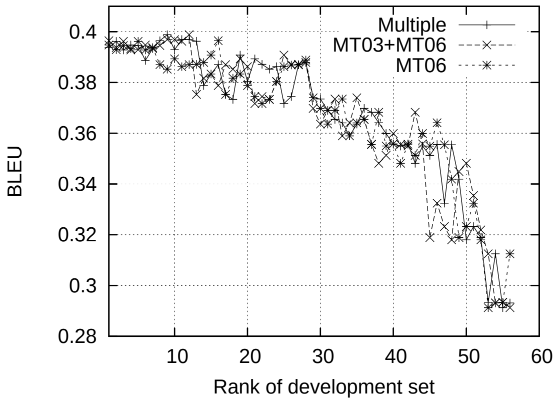


Figure 1: Correlation between similarity and BLEU on MT05 data set

In the figure, every data line shows how BLEU score changes when different pre-built development set in $S_{MT03+MT06}$ is used for model learning. The data points in each line are sorted by the rank of similarity between the development set in use and the MT05 data set. We also compared results based on 3 reference model settings. In the first one (multiple), the similarity was computed using Equation 6, and the reference model set contains all models learnt from the development sets in $S_{MT03+MT06}$. The other two settings use reference models learnt from MT06 and MT03+MT06 data sets respectively.

We can observe from the figure that the correlation between BLEU scores and data set similarity can only be identified on macro scales for all the three similarity settings. Although using data similarity may not be able to select the perfect development data set from S_{D_F}, by picking a development set with highest similarity score, we can usually (almost always) get good enough BLEU scores in our experiments.

4.4 Results of Development Data Dynamic Generation

We ran two sets of experiments for the method of development data dynamic construction.

The first one was designed to investigate how the size of extracted development data affects the translation performance. Using MT05 and MT08 as test sets and MT03+MT06 as D_F, we ran experiments for the algorithm presented in Section 3.3 with $n = 200$ to $n = 1,000$. In this experiment we did not observe significant enough changes in BLEU scores – the difference between the highest and lowest numbers is generally less than 0.005.

The second one aimed at examining how BLEU numbers changes when the extracted development data were iteratively updated. Figure 2 shows one set of results on test sets MT05 and MT08 using MT03+MT06 data set as D_F and n set to 400.

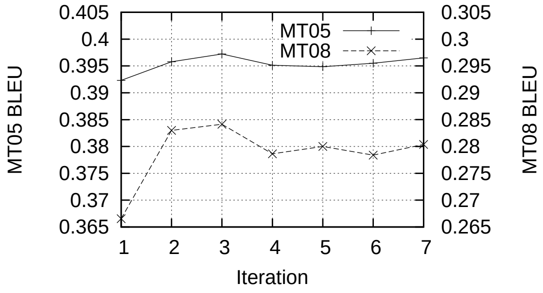


Figure 2: BLEU score as function of iteration in dynamic development data extraction.

The similarity usually converged after 2 to 3 iterations, which is consistent with trend of BLEU scores on test sets. However, in all our experimental settings, we did not observe any results significantly better than using the development set preconstruction method.

5 Discussions

Some of the previous work related to building adaptive SMT systems were discussed in the domain adaptation context, in which one fundamental idea is to estimate a more suitable domain-specific translation model or language model. When the target domain is already known, adding a small amount of domain data (both monolingual and bilingual) to the existing training corpora has been shown to be very effective in practice. But model adaptation is required in more scenarios other than explicitly defined domains. As shown by the results in Table 2, even for the data from the same domain, distribution mismatch can also be a problem.

There are also considerable efforts made to deal with the unknown distribution of text to be translated, and the research topics were still focused on translation and language model adaptation. Typical methods used in this direction include dynamic data selection (Lü et al., 2007; Zhao et al., 2004; Hildebrand et al., 1995) and data weighting (Foster and Kuhn, 2007; Matsoukas et al., 2009). All the mentioned methods use information retrieval techniques to identify relevant training data from the entire training corpora.

Our work presented here also makes no assumption about the distribution of test data, but it differs from the previous methods significantly from a log-linear model's perspective. Adjusting translation and language models based on test data can be viewed as *adaptation of feature values*, while our method is essentially *adaptation of feature weights*. This difference makes these two kinds of methods complementary to each other — it is possible to make further improvement by using both of them in one task.

To our knowledge, there is no dedicated discussion on principled methods to perform development data selection in previous research. In Lü et al. (2007), log-linear model parameters can also be adjusted at decoding time. But in their approach, the adjustment was based on heuristic rules and re-weighted training data distribution. In addition, compared with training data selection, the computational cost of development data selection is much smaller.

From machine learning perspective, both proposed methods can be viewed as certain form of transductive learning applied to the SMT task (Ueffing et al., 2007). But our methods do not rely on surface similarities between training and training/development sentences, and development/test sentences are not used to re-train SMT sub-models.

6 Conclusions and Future Work

In this paper, we addressed the data mismatch issue between training and decoding time of log-linear SMT models, and presented principled methods for dynamically inferring test data dependent model parameters with development set selection. We describe two algorithms for this task, development set pre-construction and dynamic construction, and evaluated our method on the NIST data sets for the Chinese-English translation task. Experimental results show that our methods are capable of consistently achieving good translation performance on multiple test sets with different data distributions without manual tweaking of log-linear model parameters. Though theoretically using the dynamic construction method could bring better results, the pre-construction method performs comparably well in our experimental settings. Considering the fact that the pre-consruction method is computationally cheaper, it should be a better choice in practice.

In the future, we are interested in two directions. One is to explore the possibility to perform data clustering on test set as well and choosing suitable model parameters for each cluster separately. The other involves dynamic SMT model selection – for example, some parts of the test data fit the phrase-based model better while other parts can be better translated using a syntax-based model.

References

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of the 43th Annual Meeting of the Association for Computational Linguistic (ACL)*. Ann Arbor, Michigan.

George Foster and Roland Kuhn. 2007. Mixture-Model Adaptation for SMT. In *Proc. of the Second ACL Workshop on Statistical Machine Translation.*. Prague, Czech Republic.

Michel Galley, Mark Hopkins, Kevin Knight and Daniel Marcu. 2004. What's in a translation rule? In *Proc. of the Human Language Technology Conf. (HLT-NAACL)*. Boston, Massachusetts.

Almut Hildebrand, Matthias Eck, Stephan Vogel, and Alex Waibel. 1995. Adaptation of the Translation Model for Statistical Machine translation Based on Information Retrieval. In *Proc. of EAMT*. Budapest, Hungary.

Philipp Koehn, Franz Och and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proc. of the Human Language Technology Conf. (HLT-NAACL)*. Edmonton, Canada.

Yang Liu, Qun Liu, and Shouxun Lin. 2007. Tree-to-string alignment template for statistical machine translation. In *Proc. of the 45th Annual Meeting of the Association for Computational Linguistic (ACL)*. Prague, Czech Republic.

Yajuan Lü, Jin Huang and Qun Liu. 2007. Improving Statistical Machine Translation Performance by Training Data Selection and Optimization. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*. Prague, Czech Republic.

Spyros Matsoukas, Antti-Veikko I. Rosti and Bing Zhang. 2009. Discriminative Corpus Weight Estimation for Machine Translation. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*. Singapore.

Franz Och and Hermann Ney. 2002. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistic (ACL)*. Philadelphia, PA.

Franz Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of the 41th Annual Meeting of the Association for Computational Linguistic (ACL)*. Sapporo, Japan.

Nicola Ueffing, Gholamreza Haffari and Anoop Sarkar. 2007. Transductive learning for statistical machine translation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*. Prague, Czech Republic.

Bing Zhao, Matthias Eck, and Stephan Vogel. 2004. Language Model Adaptation for Statistical Machine Translation with Structured Query Models. In *Proc. of COLING*. Geneva, Switzerland.

Learning the Scope of Negation via Shallow Semantic Parsing

Junhui Li   Guodong Zhou[*]   Hongling Wang   Qiaoming Zhu
School of Computer Science and Technology
Soochow University at Suzhou
{lijunhui, gdzhou, redleaf, qmzhu}@suda.edu.cn

Abstract

In this paper we present a simplified shallow semantic parsing approach to learning the scope of negation (SoN). This is done by formulating it as a shallow semantic parsing problem with the negation signal as the predicate and the negation scope as its arguments. Our parsing approach to SoN learning differs from the state-of-the-art chunking ones in two aspects. First, we extend SoN learning from the chunking level to the parse tree level, where structured syntactic information is available. Second, we focus on determining whether a constituent, rather than a word, is negated or not, via a simplified shallow semantic parsing framework. Evaluation on the BioScope corpus shows that structured syntactic information is effective in capturing the domination relationship between a negation signal and its dominated arguments. It also shows that our parsing approach much outperforms the state-of-the-art chunking ones.

1 Introduction

Whereas negation in predicate logic is well-defined and syntactically simple, negation in natural language is much complex. Generally, learning the scope of negation involves two subtasks: negation signal finding and negation scope finding. The former decides whether the words in a sentence are negation signals (i.e., words indicating negation, e.g., *no*, *not*, *fail*, *rather than*), where the semantic information of the words, rather than the syntactic information, plays a critical role. The latter determines the sequences of words in the sentence which are negated by the given negation signal. Compared with negation scope finding, negation signal finding is much simpler and has been well resolved in the literature, e.g. with the accuracy of 95.8%-98.7% on the three subcorpora of the Bioscope corpus (Morante and Daelemans, 2009). In this paper, we focus on negation scope finding instead. That is, we assume golden negation signal finding.

Finding negative assertions is essential in information extraction (IE), where in general, the aim is to derive factual knowledge from free text. For example, Vincze et al. (2008) pointed out that the extracted information within the scopes of negation signals should either be discarded or presented separately from factual information. This is especially important in the biomedical domain, where various linguistic forms are used extensively to express impressions, hypothesized explanations of experimental results or negative findings. Szarvas et al. (2008) reported that 13.45% of the sentences in the abstracts subcorpus of the BioScope corpus and 12.70% of the sentences in the full papers subcorpus of the Bioscope corpus contain negative assertions. In addition to the IE tasks in the biomedical domain, SoN learning has attracted more and more attention in some natural language processing (NLP) tasks, such as sentiment classification (Turney, 2002). For example, in the sentence "*The chair is not comfortable but cheap*", although both the polarities of the words "*comfortable*" and "*cheap*" are positive, the polarity of "the chair" regarding the attribute "*cheap*" keeps positive while the polarity of "the chair" regarding the attribute "*comfortable*" is reversed due to the negation signal "*not*".

Most of the initial research on SoN learning focused on negated terms finding, using either some heuristic rules (e.g., regular expression), or machine learning methods (Chapman et al., 2001; Huang and Lowe, 2007; Goldin and Chapman, 2003). Negation scope finding has been largely ignored until the recent release of

[*] Corresponding author

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 671–679,
Beijing, August 2010

the BioScope corpus (Szarvas et al., 2008; Vincze et al., 2008). Morante et al. (2008) and Morante and Daelemans (2009) pioneered the research on negation scope finding by formulating it as a chunking problem, which classifies the words of a sentence as being inside or outside the scope of a negation signal. However, this chunking approach suffers from low performance, in particular on long sentences, due to ignoring structured syntactic information. For example, given golden negation signals on the Bioscope corpus, Morante and Daelemans (2009) only got the performance of 50.26% in PCS (percentage of correct scope) measure on the full papers subcorpus (22.8 words per sentence on average), compared to 87.27% in PCS measure on the clinical reports subcorpus (6.6 words per sentence on average).

This paper explores negation scope finding from a parse tree perspective and formulates it as a shallow semantic parsing problem, which has been extensively studied in the past few years (Carreras and Màrquez, 2005). In particular, the negation signal is recast as the predicate and the negation scope is recast as its arguments. The motivation behind is that structured syntactic information plays a critical role in negation scope finding and should be paid much more attention, as indicated by previous studies in shallow semantic parsing (Gildea and Palmer, 2002; Punyakanok et al., 2005). Our parsing approach to negation scope finding differs from the state-of-the-art chunking ones in two aspects. First, we extend negation scope finding from the chunking level into the parse tree level, where structured syntactic information is available. Second, we focus on determining whether a constituent, rather than a word, is negated or not. Evaluation on the BioScope corpus shows that our parsing approach much outperforms the state-of-the-art chunking ones.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces the Bioscope corpus on which our approach is evaluated. Section 4 describes our parsing approach by formulating negation scope finding as a simplified shallow semantic parsing problem. Section 5 presents the experimental results. Finally, Section 6 concludes the work.

2 Related Work

While there is a certain amount of literature within the NLP community on negated terms finding (Chapman et al., 2001; Huang and Lowe, 2007; Goldin and Chapman, 2003), there are only a few studies on negation scope finding (Morante et al., 2008; Morante and Daelemans, 2009).

Negated terms finding

Rule-based methods dominated the initial research on negated terms finding. As a representative, Chapman et al. (2001) developed a simple regular expression-based algorithm to detect negation signals and identify medical terms which fall within the negation scope. They found that their simple regular expression-based algorithm can effectively identify a large portion of the pertinent negative statements from discharge summaries on determining whether a finding or disease is absent. Besides, Huang and Lowe (2007) first proposed some heuristic rules from a parse tree perspective to identify negation signals, taking advantage of syntactic parsing, and then located negated terms in the parse tree using a corresponding negation grammar.

As an alternative to the rule-based methods, various machine learning methods have been proposed for finding negated terms. As a representative, Goldin and Chapman (2003) adopted both Naïve Bayes and decision trees to distinguish whether an observation is negated by the negation signal "*not*" in hospital reports.

Negation scope finding

Morante et al. (2008) pioneered the research on negation scope finding, largely due to the availability of a large-scale annotated corpus, the Bioscope corpus. They approached the negation scope finding task as a chunking problem which predicts whether a word in the sentence is inside or outside of the negation scope, with proper post-processing to ensure consecutiveness of the negation scope. Morante and Daelemans (2009) further improved the performance by combing several classifiers.

Similar to SoN learning, there are some efforts in the NLP community on learning the scope of speculation. As a representative, Özgür and Radev (2009) divided speculation learning into two subtasks: speculation signal finding and speculation scope finding. In particular, they formulated speculation signal finding as a classification problem while employing some heuristic rules from the parse tree perspective on speculation scope finding.

3 Negation in the BioScope Corpus

This paper employs the BioScope corpus (Szarvas et al., 2008; Vincze et al., 2008)[1], a freely downloadable negation resource from the biomedical domain, as the benchmark corpus. In this corpus, every sentence is annotated with negation signals and speculation signals (if it has), as well as their linguistic scopes. Figure 1 shows a self-explainable example. In this paper, we only consider negation signals, rather than speculation ones. Our statistics shows that 96.57%, 3.23% and 0.20% of negation signals are represented by one word, two words and three or more words, respectively. Additional, adverbs (e.g., *not*, *never*) and determiners (e.g., *no*, *neither*) occupy 45.66% and 30.99% of negation signals, respectively.

<sentence id="S26.8">These findings <xcope id="X26.8.2"><cue type="**speculation**" ref="X26.8.2">indicate that</cue> <xcope id="X26.8.1">corticosteroid resistance in bronchial asthma <cue type="**negation**" ref="X26.8.1">can not</cue> be explained by abnormalities in corticosteroid receptor characteristics</xcope></xcope>.</sentence>

Figure 1: An annotated sentence in the BioScope corpus.

The Bioscope corpus consists of three subcorpora: the full papers and the abstracts from the GENIA corpus (Collier et al., 1999), and clinical (radiology) reports. Among them, the full papers subcorpus and the abstracts subcorpus come from the same genre, and thus share some common characteristics in statistics, such as the number of words in the negation scope to the right (or left) of the negation signal and the average scope length. In comparison, the clinical reports subcorpus consists of clinical radiology reports with short sentences. For detailed statistics about the three subcorpora, please see Morante and Daelemans (2009).

For preprocessing, all the sentences in the Bioscope corpus are tokenized and then parsed using the Berkeley parser[2] (Petrov and Klein, 2007) trained on the GENIA TreeBank (GTB) 1.0 (Tateisi et al., 2005)[3], which is a bracketed corpus in (almost) PTB style. 10-fold cross-validation on GTB1.0 shows that the parser achieves the performance of 86.57 in F1-measure. It is worth noting that the GTB1.0 corpus includes all the sentences in the abstracts subcorpus of the Bioscope corpus.

4 Negation Scope Finding via Shallow Semantic Parsing

In this section, we first formulate the negation scope finding task as a shallow semantic parsing problem. Then, we deal with it using a simplified shallow semantic parsing framework.

4.1 Formulating Negation Scope Finding as a Shallow Semantic Parsing Problem

Given a parse tree and a predicate in it, shallow semantic parsing recognizes and maps all the constituents in the sentence into their corresponding semantic arguments (roles) of the predicate. As far as negation scope finding considered, the negation signal can be regarded as the predicate[4], while the scope of the negation signal can be mapped into several constituents which are negated and thus can be regarded as the arguments of the negation signal. In particular, given a negation signal and its negation scope which covers $word_m$, …, $word_n$, we adopt the following two heuristic rules to map the negation scope of the negation signal into several constituents which can be deemed as its arguments in the given parse tree.

1) The negation signal itself and all of its ancestral constituents are non-arguments.
2) If constituent *X* is an argument of the given negation signal, then *X* should be the highest constituent dominated by the scope of $word_m$, …, $word_n$. That is to say, *X*'s parent constituent must cross-bracket or include the scope of $word_m$, …, $word_n$.

[1] http://www.inf.u-szeged.hu/rgai/bioscope

[2] http://code.google.com/p/berkeleyparser/

[3] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA

[4] If a negation signal consists of multiply words (e.g., rather than), the last word (e.g., than) is chosen to represent the negation signal.

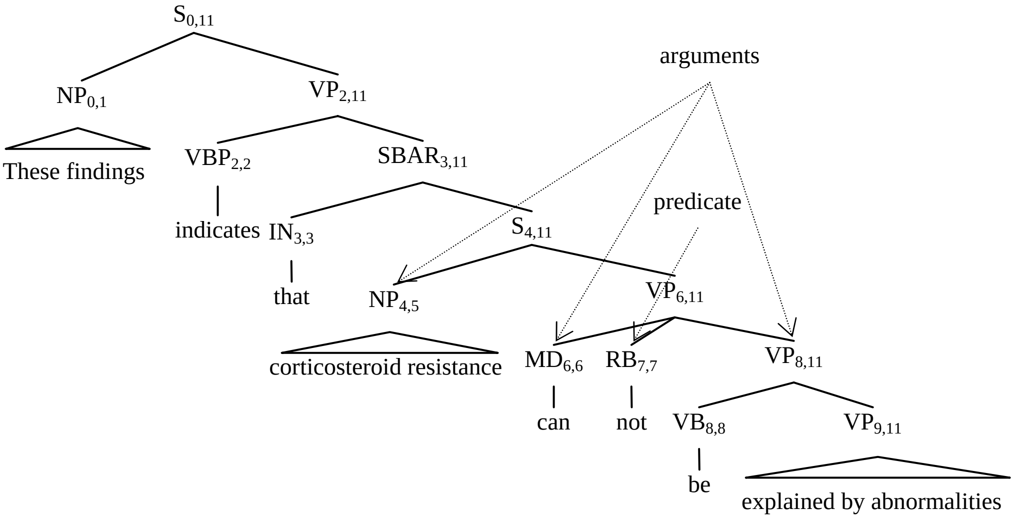


Figure 2: An illustration of a negation signal and its arguments in a parse tree.

The first rule ensures that no argument covers the negation signal while the second rule ensures no overlap between any two arguments. For example, in the sentence "*These findings indicate that corticosteroid resistance can not be explained by abnormalities*", the negation signal "*can not*" has the negation scope "*corticosteroid resistance can not be explained by abnormalities*". As shown in Figure 2, the node "$RB_{7,7}$" (i.e., *not*) represents the negation signal "*can not*" while its arguments include three constituents {$NP_{4,5}$, $MD_{6,6}$, and $VP_{8,11}$}. It is worth noting that according to the above rules, negation scope finding via shallow semantic parsing, i.e. determining the arguments of a given negation signal, is robust to some variations in parse trees. This is also empirically justified by our later experiments. For example, if the $VP_{6,11}$ in Figure 2 is incorrectly expanded by the rule $VP_{6,11} \rightarrow MD_{6,6}+RB_{7,7}+VB_{8,8}+VP_{9,11}$, the negation scope of the negation signal "*can not*" can still be correctly detected as long as {$NP_{4,5}$, $MD_{6,6}$, $VB_{8,8}$, and $VP_{9,11}$} are predicted as the arguments of the negation signal "*can not*".

Compared with common shallow semantic parsing which needs to assign an argument with a semantic label, negation scope finding does not involve semantic label classification and thus could be divided into three consequent phases: argument pruning, argument identification and post-processing.

4.2 Argument Pruning

Similar to the predicate-argument structures in common shallow semantic parsing, the negation signal-scope structures in negation scope finding can be also classified into several certain types and argument pruning can be done by employing several heuristic rules to filter out constituents, which are most likely non-arguments of a negation signal. Similar to the heuristic algorithm as proposed in Xue and Palmer (2004) for argument pruning in common shallow semantic parsing, the argument pruning algorithm adopted here starts from designating the negation signal as the current node and collects its siblings. It then iteratively moves one level up to the parent of the current node and collects its siblings. The algorithm ends when it reaches the root of the parse tree. To sum up, except the negation signal and its ancestral constituents, any constituent in the parse tree whose parent covers the given negation signal will be collected as argument candidates. Taking the negation signal node "$RB_{7,7}$" in Figure 2 as an example, constituents {$MD_{6,6}$, $VP_{8,11}$, $NP_{4,5}$, $IN_{3,3}$, $VBP_{2,2}$, and $NP_{0,1}$} are collected as its argument candidates consequently.

4.3 Argument Identification

Here, a binary classifier is applied to determine the argument candidates as either valid arguments or non-arguments. Similar to argument identification in common shallow semantic parsing, the structured syntactic information plays a critical role in negation scope finding.

Basic Features

Table 1 lists the basic features for argument identification. These features are also widely used in common shallow semantic parsing for both verbal and nominal predicates (Xue, 2008; Li et al., 2009).

Feature	Remarks
b1	Negation: the stem of the negation signal, e.g., not, rather_than. (*can_not*)
b2	Phrase Type: the syntactic category of the argument candidate. (*NP*)
b3	Path: the syntactic path from the argument candidate to the negation signal. (*NP<S>VP>RB*)
b4	Position: the positional relationship of the argument candidate with the negation signal. "left" or "right". (*left*)

Table 1: Basic features and their instantiations for argument identification in negation scope finding, with $NP_{4,5}$ as the focus constituent (i.e., the argument candidate) and "*can not*" as the given negation signal, regarding Figure 2.

Additional Features

To capture more useful information in the negation signal-scope structures, we also explore various kinds of additional features. Table 2 shows the features in better capturing the details regarding the argument candidate and the negation signal. In particular, we categorize the additional features into three groups according to their relationship with the argument candidate (AC, in short) and the given negation signal (NS, in short).

Some features proposed above may not be effective in argument identification. Therefore, we adopt the greedy feature selection algorithm as described in Jiang and Ng (2006) to pick up positive features incrementally according to their contributions on the development data. The algorithm repeatedly selects one feature each time which contributes most, and stops when adding any of the remaining features fails to improve the performance. As far as the negation scope finding task concerned, the whole feature selection process could be done by first running the selection algorithm with the basic features (b1-b4) and then incrementally picking up effective features from (ac1-ac6, AC1-AC2, ns1-ns4, NS1-NS2, nsac1-nsac2, and NSAC1 -NSAC7).

Feature	Remarks
argument candidate (AC) related	
ac1	the headword (ac1H) and its POS (ac1P). (*resistance*, *NN*)
ac2	the left word (ac2W) and its POS (ac2P). (*that*, *IN*)
ac3	the right word (ac3W) and its POS (ac3P). (*can*, *MD*)
ac4	the phrase type of its left sibling (ac4L) and its right sibling (ac4R). (*NULL*, *VP*)
ac5	the phrase type of its parent node. (*S*)
ac6	the subcategory. (*S:NP+VP*)
combined features (AC1-AC2)	
b2&fc1H, b2&fc1P	
negation signal (NS) related	
ns1	its POS. (*RB*)
ns2	its left word (ns2L) and right word (ns2R). (*can*, *be*)
ns3	the subcategory. (*VP:MD+RB+VP*)
ns4	the phrase type of its parent node. (*VP*)
combined features (NS1-NS2)	
b1&ns2L, b1&ns2R	
NS-AC-related	
nsac1	the compressed path of b3: compressing sequences of identical labels into one. (*NP<S>VP>RB*)
nsac2	whether AC and NS are adjacent in position. "yes" or "no". (*no*)
combined features (NSAC1-NSAC7)	
b1&b2, b1&b3, b1&nsac1, b3&NS1, b3&NS2, b4&NS1, b4&NS2	

Table 2: Additional features and their instantiations for argument identification in negation scope finding, with $NP_{4,5}$ as the focus constituent (i.e., the argument candidate) and "*can not*" as the given negation signal, regarding Figure 2.

4.4 Post-Processing

Although a negation signal in the BioScope corpus always has only one continuous block as its negation scope (including the negation signal itself), the negation scope finder may result in discontinuous negation scope due to independent prediction in the argument identification phase. Given the golden negation signals, we observed that 6.2% of the negation scopes predicted by our negation scope finder are discontinuous.

Figure 3 demonstrates the projection of all the argument candidates into the word level. According to our argument pruning algorithm in Section 4.2, except the words presented by the negation signal, the projection covers the whole sentence and each constituent (*LAC_i* or *RAC_j* in Figure 3) receives a probability distribution of being an argument of the given negation signal in the argument identification phase.

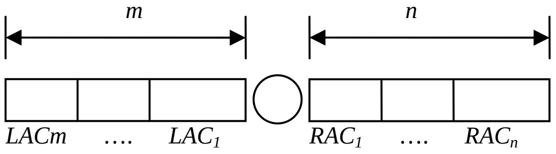


Figure 3: Projecting the left and the right argument candidates into the word level.

Since a negation signal is deemed inside of its negation scope in the BioScope corpus, our post-processing algorithm first includes the negation signal in its scope and then starts to identify the left and the right scope boundaries, respectively.

As shown in Figure 3, the left boundary has *m+1* possibilities, namely the negation signal itself, the leftmost word of constituent *LAC_i* (*1<=i<=m*). Supposing *LAC_i* receives probability of *P_i* being an argument, we use the following formula to determine *LAC_{k*}* whose leftmost word represents the boundary of the left scope. If *k*=0*, then the negation signal itself represents its left boundary.

$$k^* = \arg\max_k \prod_{i=1}^{k} P_i * \prod_{i=k+1}^{m} (1-P_i)$$

Similarly, the right boundary of the given negation signal can be decided.

5 Experimentation

We have evaluated our shallow semantic parsing approach to negation scope finding on the BioScope corpus.

5.1 Experimental Settings

Following the experimental setting in Morante and Daelemans (2009), the abstracts subcorpus is randomly divided into 10 folds so as to perform 10-fold cross validation, while the performance on both the papers and clinical reports subcorpora is evaluated using the system trained on the whole abstracts subcorpus. In addition, SVMLight[5] is selected as our classifier. In particular, we adopt the linear kernel and the training parameter C is fine-tuned to 0.2.

The evaluation is made using the accuracy. We report the accuracy using three measures: *PCLB* and *PCRB*, which indicate the percentages of correct left boundary and right boundary respectively, *PCS*, which indicates the percentage of correct scope as a whole.

5.2 Experimental Results on Golden Parse Trees

In order to select beneficial features from the additional features proposed in Section 4.3, we randomly split the abstracts subcorpus into training and development datasets with proportion of 4:1. After performing the greedy feature selection algorithm on the development data, features {NSAC5, ns2R, NS1, ac1P, ns3, NSAC7, ac4R} are selected consecutively for argument identification. Table 3 presents the effect of selected features in an incremental way on the development data. It shows that the additional features significantly improve the performance by 11.66% in PCS measure from 74.93% to 86.59% ($\chi^2; p < 0.01$).

Feature	PCLB	PCRB	PCS
Baseline	84.26	88.92	74.93
+NSAC5	90.96	88.92	81.34
+ns2R	91.55	88.92	81.92
+NS1	92.42	89.50	83.09
+ac1P	93.59	89.50	84.26
+ns3	93.88	90.09	84.84
+NSAC7	94.75	89.80	85.42
+ac4R	95.04	90.67	86.59

Table 3: Performance improvement (%) of including the additional features in an incremental way on the development data (of the abstracts subcorpus).

However, Table 3 shows that the additional features behave quite differently in terms of PCLB and PCRB measures. For example, PCLB measure benefits more from features NSAC5, ns2R, NS1, ac1P, and NSAC7 while PCRB measure benefits more from features NS1 and ac4R. It also shows that the features (e.g., NSAC5, ns2R, NS1, NSAC7) related to neighboring words of the negation signal play a critical role in recognizing both left and right boundaries. This may be due to the fact that neighboring words usually imply sentential information. For example, "*can not be*" indicates a passive clause while "*did not*" indicates an active clause. Table 3 also shows that the recognition of left boundaries is much easier than that of right boundaries. This may be due

[5] http://svmlight.joachims.org/

to the fact that 83.6% of negation signals have themselves as the left boundaries in the abstracts subcorpus.

Table 4 presents the performance on the abstracts subcorpus by performing 10-fold cross-validation. It shows that the additional features significantly improve the performance over the three measures ($\chi^2; p < 0.01$).

Feature	PCLB	PCRB	PCS
Baseline	84.29	87.82	74.05
+selected features	93.06	88.96	83.10

Table 4: Performance (%) of negation scope finding on the abstracts subcorpus using 10-fold cross-validation.

5.3 Experimental Results on Automatic Parse Trees

The GTB1.0 corpus contains 18,541 sentences in which 11,850 of them (63.91%) overlap with the sentences in the abstracts subcorpus[6]. In order to get automatic parse trees for the sentences in the abstracts subcorpus, we train the Berkeley parser with the remaining 6,691 sentences in GTB1.0. The Berkeley parser trained on 6,691 sentences achieves the performance of 85.22 in F1-measure on the other sentences in GTB1.0. For both the full papers and clinical reports subcorpora, we get their automatic parse trees by using two Berkeley parsers: one trained on 6,691 sentences in GBT1.0, and the other trained on all the sentences in GTB1.0.

To test the performance on automatic parse trees, we employ two different configurations. First, we train the argument identification classifier on the abstracts subcorpus using *automatic parse trees* produced by Berkeley parser trained on 6,691 sentences. The experimental results are presented in the rows of *autoparse(t&t)* in Table 5 and Table 6. Then, we train the argument identification classifier on the abstracts subcorpus using *golden parse trees*. The experimental results are presented in the rows of *autoparse(test)* in Table 5 and Table 6.

We also report an oracle performance to explore the best possible performance of our system by assuming that our negation scope finder can always correctly determine whether a candidate is an argument or not. That is, if an argument candidate is outside or cross-brackets with the golden negation scope, then it is a non-argument. The oracle performance is presented in the rows of *oracle* in Table 5 and Table 6.

Table 5 and Table 6 show that:

1) Automatic syntactic parsing lowers the performance of negation scope finding on the abstracts subcorpus in all three measures (e.g. from 83.10 to 81.84 in PCS). As expected, the parser trained on the whole GTB1.0 corpus works better than that trained on 6,691 sentences (e.g. 64.02 Vs. 62.70, and 89.79 Vs. 85.21 in PCS measure on the full papers and the clinical reports subcorpora, respectively). However, the performance decrease shows that negation scope finding is not as sensitive to automatic syntactic parsing as common shallow semantic parsing, whose performance might decrease by about ~10 in F1-measure (Toutanova et al., 2005). This indicates that negation scope finding via shallow semantic parsing is robust to some variations in the parse trees.
2) *autoparse(test)* consistently outperforms *autoparse(t&t)* on both the abstracts and the full papers subcorpora. However, it is surprising to find that *autoparse(t&t)* achieves better performance on the clinical reports subcorpus than *autoparse(test)*. This may be due to the special characteristics of the clinical reports subcorpus, which mainly consists of much shorter sentences with 6.6 words per sentence on average, and better adaptation of the argument identification classifier to the variations in the automatic parse trees.
3) The performance on all three subcorpora indicates that the recognition of right boundary is much harder than that of left boundary. This may be due to the longer right boundary on an average. Our statistics shows that the average left/right boundaries are 1.1/6.9, 0.1/3.7, and 1.2/6.5 words on the abstracts, the full papers and the clinical reports subcorpora, respectively.
4) The oracle performance is less sensitive to automatic syntactic parsing. In addition, given the performance gap between the performance of our negation scope finder and the oracle performance, there is still much room for further performance improvement.

[6] There are a few cases where two sentences in the abstracts subcorpus map into one sentence in GTB.

	Abstracts			Papers			Clinical		
	PCLB	PCRB	PCS	PCLB	PCRB	PCS	PCLB	PCRB	PCS
autoparse(t&t)	91.97	87.82	80.88	85.45	67.20	59.26	97.48	88.30	85.89
autoparse(test)	92.71	88.33	81.84	87.57	68.78	62.70	97.48	87.73	85.21
oracle	99.72	94.59	94.37	98.94	84.13	83.33	99.89	98.39	98.39

Table 5: Performance (%) of negation scope finding on the three subcorpora by using automatic parser trained with 6,691 sentences in GTB1.0.

	Papers			Clinical		
	PCLB	PCRB	PCS	PCLB	PCRB	PCS
autoparse(t&t)	85.98	67.99	60.32	97.48	92.66	90.48
autoparse(test)	87.83	70.11	64.02	97.36	92.20	89.79
oracle	98.94	83.86	83.07	99.77	97.94	97.82

Table 6: Performance (%) of negation scope finding on the two subcorpora by using automatic parser trained with all the sentences in GTB1.0.

Method	Abstracts	Papers	Clinical
M et al. (2008)	57.33	n/a	n/a
M & D (2009)	73.36	50.26	87.27
Our baseline	73.42	53.70	88.42
Our final system	81.84	64.02	89.79

Table 7: Performance comparison over the PCS measure (%) of our system with other state-of-the-art ones.

Table 7 compares our performance in PCS measure with related work. It shows that even our baseline system with four basic features as presented in Table 1 performs better than Morante et al. (2008) and Morante and Daelemans(2009). This indicates the appropriateness of our simplified shallow semantic parsing approach and the effectiveness of structured syntactic information on negation scope finding. It also shows that our final system significantly outperforms the state-of-the-art ones using a chunking approach, especially on the abstracts and full papers subcorpora. However, the improvement on the clinical reports subcorpus is less apparent, partly due to the fact that the sentences in this subcorpus are much simpler (with average length of 6.6 words per sentence) and thus a chunking approach can achieve high performance. Following are two typical sentences from the clinical reports subcorpus, where the negation scope covers the whole sentence (except the period punctuation). Such sentences account for 57% of negation sentences in the clinical reports subcorpus.

(1) No evidence of focal pneumonia .

(2) No findings to account for symptoms .

6 Conclusion

In this paper we have presented a simplified shallow semantic parsing approach to negation scope finding by formulating it as a shallow semantic parsing problem, which has been extensively studied in the past few years. In particular, we regard the negation signal as the predicate while mapping the negation scope into several constituents which are deemed as arguments of the negation signal. Evaluation on the Bioscope corpus shows the appropriateness of our shallow semantic parsing approach and that structured syntactic information plays a critical role in capturing the domination relationship between a negation signal and its negation scope. It also shows that our parsing approach much outperforms the state-of-the-art chunking ones. To our best knowledge, this is the first research on exploring negation scope finding via shallow semantic parsing.

Future research will focus on joint learning of negation signal and its negation scope findings. Although Morante and Daelemans (2009) reported the performance of 95.8%-98.7% on negation signal finding, it lowers the performance of negation scope finding by about 7.29%-16.52% in PCS measure.

Acknowledgments

This research was supported by Projects 60683150, 60970056, and 90920004 under the National Natural Science Foundation of China, Project 20093201110006 under the Specialized Research Fund for the Doctoral Program of Higher Education of China.

References

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In *Proceedings of CoNLL 2005*.

Wendy W. Chapman, Will Bridewell, Paul Hanbury, Gregory F. Cooper, and Bruce G. Buchanan. 2001. A Simple Algorithm for Identifying Negated Findings and Diseases in Discharge Summaries. *Journal of Biomedical Informatics*, 34: 301-310.

Nigel Collier, Hyun Seok Park, Norihiro Ogata, et al. 1999. The GENIA project: corpus-based knowledge acquisition and information extraction from genome research papers. In *Proceedings of EACL 1999*.

Daniel Gildea and Martha Palmer. 2002. The Necessity of Parsing for Predicate Argument Recognition. In *Proceedings of ACL 2002*.

Ilya M. Goldin and Wendy W. Chapman. 2003. Learning to Detect Negation with 'Not' in Medical Texts. In *Proceedings of SIGIR 2003*.

Yang Huang and Henry Lowe. 2007. A Novel Hybrid Approach to Automated Negation Detection in Clinical Radiology Reports. *Journal of the American Medical Informatics Association*, 14(3): 304-311.

Zheng Ping Jiang and Hwee Tou Ng. 2006. Semantic Role Labeling of NomBank: A Maximum Entropy Approach. In *Proceedings of EMNLP 2006*.

Junhui Li, Guodong Zhou, Hai Zhao, Qiaoming Zhu, and Peide Qian. Improving Nominal SRL in Chinese Language with Verbal SRL Information and Automatic Predicate Recognition. In *Proceedings of EMNLP 2009*.

Roser Morante, Anthony Liekens, and Walter Daelemans. 2008. Learning the Scope of Negation in Biomedical Texts. In *Proceedings of EMNLP 2008*.

Roser Morante and Walter Daelemans. 2009. A Metalearning Approach to Processing the Scope of Negation. In *Proceedings of CoNLL 2009*.

Arzucan Özgür; Dragomir R. Radev. 2009. Detecting Speculations and their Scopes in Scientific Text. In *Proceedings of EMNLP 2009*.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proceedings of NAACL 2007*.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. The Necessity of Syntactic Parsing for Semantic Role Labeling. In *Proceedings of IJCAI 2005*.

György Szarvas, Veronika Vincze, Richárd Farkas, and János Csirik. 2008. The BioScope corpus: annotation for negation, uncertainty and their scope in biomedical texts. In *Proceedings of BioNLP 2008*.

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. 2005. Syntax Annotation for the GENIA Corpus. In *Proceedings of IJCNLP 2005, Companion volume*.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2005. Joint Learning Improves Semantic Role Labeling. In *Proceedings of ACL 2005*.

Peter D. Turney. 2002. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In *Proceedings of ACL 2002*.

Veronika Vincze, György Szarvas, Richárd Farkas, György Móra, and János Csirik. 2008. The BioScope corpus: biomedical texts annotated for uncertainty, negation and their scopes. BMC Bioinformatics, 9(Suppl 11):S9.

Nianwen Xue and Martha Palmer. 2004. Calibrating Features for Semantic Role Labeling. In *Proceedings of EMNLP 2004*.

Nianwen Xue. 2008. Labeling Chinese Predicates with Semantic Roles. *Computational Linguistics*, 34(2):225-255.

Filtered Ranking for Bootstrapping in Event Extraction

Shasha Liao
Dept. of Computer Science
New York University
liaoss@cs.nyu.edu

Ralph Grishman
Dept. of Computer Science
New York University
grishman@cs.nyu.edu

Abstract

Several researchers have proposed semi-supervised learning methods for adapting event extraction systems to new event types. This paper investigates two kinds of bootstrapping methods used for event extraction: the document-centric and similarity-centric approaches, and proposes a filtered ranking method that combines the advantages of the two. We use a range of extraction tasks to compare the generality of this method to previous work. We analyze the results using two evaluation metrics and observe the effect of different training corpora. Experiments show that our new ranking method not only achieves higher performance on different evaluation metrics, but also is more stable across different bootstrapping corpora.

1 Introduction

The goal of event extraction is to identify instances of a class of events in text, along with the arguments of the event (the participants, place, and time). In this paper we shall focus on the sub-problem of identifying the events themselves.

Event extraction systems from the early and mid 90s relied primarily on hand-coded rules, which must be written anew for every task. Since then, supervised and semi-supervised methods have been developed in order to build systems for new scenarios more easily. Supervised methods can perform quite well with enough training data, but annotating sufficient data may require months of labor. Semi-supervised methods aim to reduce the annotated data required, ideally to a small set of seeds.

Most semi-supervised event extractors seek to learn sets of *patterns* consisting of a predicate and some lexical or semantic constraints on its arguments. The semi-supervised learning was based primarily on one of two assumptions: the document-centric approach, which assumes that relevant patterns should appear more frequently in relevant documents (Riloff 1996; Yangarber et al. 2000; Yangarber 2003; Surdeanu et al 2006); and the similarity-centric approach, which assumes that relevant patterns should have lexically related terms (Stevenson and Greenwood 2005, Greenwood and Stevenson 2006).

An effective semi-supervised extractor will have good performance over a range of extraction tasks and corpora. However, many of the learning procedures just cited have been tested on only one or two extraction tasks, so their generality is uncertain. To remedy this, we have tested learners based on both assumptions, targeting both a MUC (Message Understanding Conference) scenario and several ACE (Automatic Content Extraction) event types. We identify shortcomings of the prior bootstrapping methods, propose a more effective and stable ranking method, and consider the effect of different corpora and evaluation metrics.

2 Related Work

The basic assumption of the document-centric approach is that documents containing a large number of patterns already identified as relevant to a particular IE scenario are likely to contain further relevant patterns. Riloff (1996) initiated

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 680–688,
Beijing, August 2010

this approach and claimed that if a corpus can be divided into documents involving a certain event type and those not involving that type, patterns can be evaluated based on their frequency in relevant and irrelevant documents. Yangarber et al. (2000) incorporated Riloff's metric into a bootstrapping procedure, which started with several seed patterns but required no manual document classification or corpus annotation. The seed patterns were used to identify some relevant documents, and the top-ranked patterns (based on their distribution in relevant and irrelevant documents) were added to the seed set. This process was repeated, assigning a relevance score to each document based on the relevance of the patterns it contains and gradually growing the set of relevant patterns. This approach was further refined by Surdeanu et al. (2006), who used a co-training strategy in which two classifiers seek to classify documents as relevant to a particular scenario. Patwardhan and Riloff (2007) presented an information extraction system that find relevant regions of text and applies extraction patterns within those regions. They created a self-trained relevant sentence classifier to identify relevant regions, and use a semantic affinity measure to automatically learn domain-relevant extraction patterns. They also distinguish primary patterns from secondary patterns and apply the patterns selectively in the relevant regions.

Stevenson and Greenwood (2005) (henceforth 'S&G') suggested an alternative method for ranking the candidate patterns. Their approach relied on the assumption that useful patterns will have similar lexical items to the patterns that have already been accepted. They used WordNet to calculate word similarity. They chose to represent each pattern as a vector consisting of the lexical items and used a version of the cosine metric to determine the similarity between pairs of patterns. Later, Greenwood and Stevenson (2006) introduced a structural similarity measure that could be applied to extraction patterns consisting of linked dependency chains.

3 Ranking Methods in Bootstrapping

Most semi-supervised event extraction systems are based on patterns with variables which have semantic type constraints. A simple example is "*organization* appoints *person* as *position*"; if this pattern matches a passage in a test document, a hiring event will be instantiated with the items matching the variables being the arguments of the event. So training an event extractor becomes primarily a task of acquiring these patterns. In a semi-supervised setting, this involves ranking candidate patterns and accepting the top-ranked patterns at each iteration. Our goal was to create a more robust learner through improved pattern ranking.

3.1 Problems of Document-centric Bootstrapping

Document-centric bootstrapping tries to find patterns with high frequency in relevant documents and low frequency in irrelevant documents. The assumption is that descriptions of the same event or the same type of event may occur multiple times in a document, and so a document containing a relevant pattern is more likely to contain more such patterns. This approach may end up extracting patterns for related events; for example, *start-position* often comes with *end-position* events. This effect may be salutary if the extraction scenario includes these related events (as in MUC-6), but will pose a problem if the goal is to extract individual event types. Also, because an extra corpus for bootstrapping is needed, different corpora might perform quite differently (see Figure 2).

3.2 Problems of Similarity-centric Bootstrapping

Similarity-centric bootstrapping tries to find patterns with high lexical similarities. The most crucial issue is how to evaluate the similarity of two patterns, which is based on the similarity of two words. In this strategy, no extra corpus is needed, which eliminates the effort to find a good bootstrapping corpus, but a semantic dictionary that can provide word similarity is required. S&G used WordNet[1] to provide word similarity information. However, in the similarity-centric approach, lexical polysemy can lead the bootstrapping down false paths. For example, for *start-position (hire)* events, "*name*" and "*charge*" are in the same *Synset* as *appoint*, but including these words is quite dangerous because they contain other common senses

[1] http://wordnet.princeton.edu/

unrelated to *start-position* events. For *die* events, we might have words like "*go*" and "*pass*", which are also used in very specific contexts when they refer to "*die*". If similarity-centric ranking extracts patterns including these words, performance will deteriorate very quickly, because most of the time, these words do not predicate the proper event, and more and more wrong patterns will be extracted.

3.3 Our Approach

We propose a new ranking method, which constrains the document-centric and similarity-centric assumptions, and makes a more restricted assumption: patterns that appear in relevant documents *and* are lexically similar are most likely to be relevant. This method limits the effect of ambiguous patterns by narrowing the search to relevant documents, and limits irrelevant patterns in relevant documents by word similarity restriction. For example, although "charge" has high word similarity to "appoint", its document relevance score is very low, and we will not include this word in bootstrapping starting from "appoint".

Many different combinations are possible; we propose one that uses the word similarity as a filter. The document relevance score is first applied to rank the patterns in relevant documents, then the patterns with lexical similarity scores below a similarity threshold will be removed from the ranking; only patterns above threshold will be added to the seeds. However, if in the current iteration, no pattern meets the threshold, the threshold will be lowered until new patterns can be found. We call this ranking method *filtered ranking*[2]:

$$Filter(p) = \begin{cases} Yangarber(p) & Stevenson(p) >= t \\ 0 & otherwise \end{cases}$$

where t is the threshold, which is initialized to 0.9 in our experiments.

4 System Description

Our approach is similar to that for document-centric bootstrapping, but the ranking function is changed to incorporate lexical similarity information. For our experiments bootstrapping was terminated after a fixed number of iterations; in practice, we would monitor performance on a held-out (dev-test) sample and stop when it declines for k iterations.

4.1 Pre-processing

Instead of limiting ourselves to surface syntactic relations, we want to get more general and meaningful patterns. To this end, we used semantic role labeling (Gildea and Jurafsky, 2002) to generate the logical grammatical and predicate-argument representation automatically from a parse tree (Meyers et al. 2009). The output of the semantic labeling is the dependency representation of the text, where each sentence is a graph consisting of nodes (corresponding to words) and arcs. Each arc captures up to three relations between two words: (1) a SURFACE relation, the relation between a predicate and an argument in the parse tree of a sentence; (2) a LOGIC1 (grammatical logical) relation which regularizes for lexical and syntactic phenomena like passive, relative clauses, and deleted subjects; and (3) a LOGIC2 (predicate-argument) relation corresponding to relations in PropBank (Palmer et al. 2005) and NomBank

In constructing extraction patterns from this graph, we take each dependency link along with its predicate-argument role; if that role is null, we use its logical grammatical role, and finally, its surface role. For example, for the sentence:

```
John is hit by Tom's brother.
```

we generate the patterns:

<Arg1 hit John>
<Arg0 hit brother>
<T-pos brother Tom>

where the first two represent LOGIC2 relations and the third a SURFACE relation. To reduce data sparseness, all inflected words are changed to their root form (e.g. "attackers"→"attacker"), and all names are replaced by their ACE type (*person, organization, location, etc.*), so the first pattern would become

<Arg1 hit PERSON>

4.2 Document-based Ranking

The document-centric method employs a

[2] We also tried using the product of the document relevance score and word similarity score, and found the results to be quite similar. Due to space limitations, we do not report these results here.

re-implementation of the procedure described in (Yangarber et al. 2000), using the disjunctive voting scheme for document relevance. At each iteration *i* we compute a precision score *Prec^i(p)* for each pattern *p* and a relevance score *Rel^i(d)* for each document *d*. Initially the seed patterns have precision 1 and all other patterns precision 0. These are updated by

$$\mathrm{Re}\,l^i(d) = 1 - \prod_{p \in K(d)} (1 - \mathrm{Pr}\,ec^i(p))$$

where *K(d)* is the set of accepted patterns that match document *d*, and

$$\mathrm{Pr}\,ec^{i+1}(p) = \frac{1}{|H(p)|} \bullet \sum_{d \in H(p)} \mathrm{Re}\,l^i(d)$$

where *H(p)* is the set of documents matching pattern *p*. Patterns are then ranked by

$$RankFun_{Yangarber}(p) = \frac{Sup(p)}{|H(p)|} * \log Sup(p)$$

where

$$Sup(p) = \sum_{d \in H(p)} \mathrm{Re}\,l(d)$$

(a generalization of Yangarber's metric), and the top-ranked candidates are added to the set of accepted patterns.

4.3 Pattern Similarity

For two words, there are several ways to measure their similarity using WordNet, which can be roughly divided into two categories: distance-based, including Leacock and Chodorow (1998), Wu and Palmer (1994); and information content based, including Resnik (1995), Lin (1998), and Jiang and Conrath (1997). We follow S&G (2005)'s method and use the semantic similarity of concepts based on Information Content (IC).

Every pattern consists of a predicate and a constraint ("argument") on its local syntactic context, and so the similarity of two patterns depends on the similarity of the predicates and the similarity of the arguments. We modified S&G's structural similarity measure to reflect some differences in pattern structure: first, S&G only focus on patterns headed by verbs, while we include verbs, nouns and adjectives; second, they only record the subject and object to a verb, while we record all argument relations; third, our patterns only contain a predicate and a single constraint (argument), while their pattern might contain two arguments, subject and object. With two arguments, many more patterns are possible and the vector similarity calculation over all patterns in a large corpus becomes very time consuming.

We do not limit ourselves to verb patterns because nouns and (occasionally) adjectives can also represent an event. For example, "Stevenson's promotion is a signal …" expresses a *start-position* event. Moreover, in our pattern, we assume that the predicate is more important than constraint, because it is the root (head) of the pattern in the semantic graph structure, and place different weights on predicate and constraint. Finally, the similarity of two patterns p_1 and p_2 is computed as follows:

$$\begin{aligned} &Sim(p_1,p_2) \\ &\quad = \alpha * Sim(f_1,f_2) + \beta * Sim(r_1,r_2) * Sim(a_1,a_2) \end{aligned}$$

where $\alpha+\beta=1$, *f* represents a predicate, *r* represent a role, and *a* represent an argument. In our experiment, α is set to 0.6 and β is set to 0.4. The role similarity is 1 for identical roles and for roles which generally correspond at the syntactic and predicate-argument level (arg0 ↔ subj; arg1 ↔ obj); selected other role pairs are assigned a small positive similarity (0.1 or 0.2), and others 0.

As with the document-centric method, bootstrapping begins by accepting a set of seed patterns. At each iteration, the procedure computes the similarity between all patterns in the training corpus and the currently accepted patterns and accepts the most similar pattern(s). In S&G's experiments the evaluation corpus also served as the training corpus.

5 Experiments

There have been two types of event extraction tasks. One involved several 'elementary' event types, such as "attack", "die", "injure" etc.; for example, the ACE 2005 evaluation[3] used a set of 33 event types and subtypes. The other type involved a *scenario* – a set of related events, like "attacks and the damage, injury, and death they cause", or "arrest, trial, sentencing etc.". The

[3] See http://projects.ldc.upenn.edu/ace/docs/English-Events-Guidelines_v5.4.3.pdf for a description of this task.

MUC evaluations included two scenarios that have been the subject of considerable research on learning methods: *terrorist incidents* (MUC-3/4) and *executive succession* (MUC-6).

We conducted experiments on the MUC-6 task to make a comparison to previous work. We also did experiments on ACE 2005 data, because it provides many distinct event types; we conducted experiments on three disparate event types: *attack*, *die*, and *start-position*. Note that MUC-6 identifies a scenario while ACE identifies specific event types, and types which are in the same MUC scenario might represent different ACE events. For example, the *executive succession* scenario (MUC-6) includes the *start-position* and *end-position* events in ACE.

5.1 Data Description

There are four corpora used in the experiments:

MUC-6 corpora

- **Bootstrapping:** pre-selected data from the Reuters corpus (Rose et al. 2002) from 1996 and 1997, including 3000 related documents and 3000 randomly chosen unrelated documents
- **Evaluation:** MUC-6 annotated data, including 200 documents (official training and test). We were guided by the MUC-6 key file in annotating every document and sentence as relevant or irrelevant.

ACE corpora

- **Bootstrapping:** untagged data from the Gigaword corpus from January 2006, including 14,171 English newswire articles from Agence France-Presse (AFP).
- **Evaluation:** ACE 2005 annotated (training) data, including 589 documents

5.2 Parameters used in Experiments

In our bootstrapping process, we only extract patterns appearing more than 2 times in the corpus, and the similarity filter threshold is originally set to 0.9. If no patterns are found, it is reduced by 0.1 until new patterns are found.

In each iteration, the top 3 patterns in the ranking function will be added to the seeds.

For the similarity-centric method, only patterns appearing more than 2 times and in less than 30% of the documents will be extracted, which is the same as S&G's approach.

5.3 MUC-6 Experiments

Our overall goal was to demonstrate that filtered ranking was in all cases competitive with and in at least some cases clearly superior to the earlier methods, over a range of extraction tasks and bootstrapping corpora. We began with the MUC-6 task, where the efficacy of the earlier methods had already been demonstrated.

< Arg0 resign Person >
< Arg1 appoint Person >
< Arg0 appoint Org_commercial>
<Arg1 succeed Person >

Table 1. Seeds for MUC-6 evaluation

For MUC-6 evaluation, we follow S&G's approach and assess extraction patterns by their ability to identify event-relevant sentences.[4] The system treats a sentence as relevant if it matches an extraction pattern. Bootstrapping starts from four seeds which yield 80% precision and 24% recall for sentence filtering.

To compare with previous work, we tested the filtered ranking method on two corpora: the first is the Reuters corpus used in S&G's recreation of Yangarber's experiment (Filter1), to compare with their results for the document-centric method; the second uses the test corpus as S&G did (Filter2), to compare with their results for the similarity-centric method. We compare methods based on peak F score; in practice, this would mean controlling the bootstrapping using a held-out test sample.

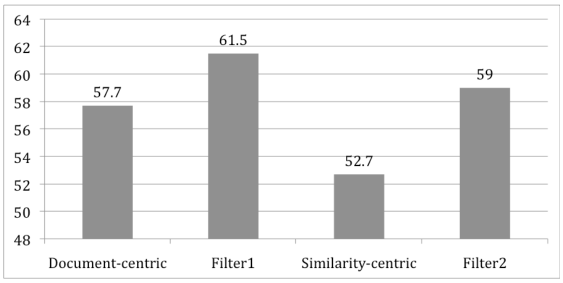


Figure 1. F score for different ranking methods on MUC-6 evaluation

Figure 1 showed that the filtered ranking

[4] We also tried the document filtering evaluation introduced by Yangarber but, as S&G observed, this metric is too insensitive because over 50% of the documents in the MUC-6 test set are relevant.

methods edge out both document and similarity-centric methods. Our scores are comparable to S&G's, although they report somewhat better performance for similarity-centric than for document-centric (55 vs. 51) whereas document-centric did better for us. This difference may reflect differences in pattern generation (discussed above) and possibly differences in the specific corpora used.

However, document-centric bootstrapping needs an extra corpus for bootstrapping; S&G used a pre-selected corpus that contains approximately same number of relevant and irrelevant documents[5]. We wanted to check if such a corpus is essential for the document-centric method, and if the need for pre-selection can be reduced through filtered ranking. Thus, we set up another experiment to see if the document-centric method is stable or sensitive to different corpora. We used two additional corpora for MUC-6 evaluation: one is a subset of the Wall Street Journal (WSJ) 1991 corpus, which contains 18,734 untagged documents; the other is the Gigaword AFP corpus described in section 5.1. Both corpora are much larger than the Reuters corpus, and while we do not have precise information about relevant document density, the WSJ contains quite a few *start-position* events because it is primarily business news; the Gigaword corpus (AFP newswire) has fewer *start-position* events because it contains a wider variety of news.

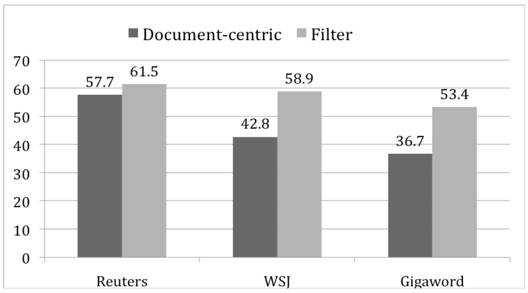


Figure 2. Document-centric and Filtered ranking results on different corpora for MUC-6

Figure 2 showed that the document-centric method performs quite differently on different corpora, which indicates that a pre-selected corpus plays an important role in document-centric ranking. It suggests that the percentage of relevant documents may be more important than the overall corpus size. The figure also shows that filtered ranking is much more stable across different corpora. Richer corpora still have better peak performance, but the difference is not quite as great; also, peak performance on a given corpus is consistently better than the document-centric method.

From the above experiments, we conclude that our filtering method is better in two aspects: first, bootstrapping on the same corpus performs better than either document or similarity-centric methods; second, if we can not get a corpus with an assured high density of relevant documents, it is safer to use filtered ranking because it is more stable across different corpora.

5.4 ACE2005 Experiments

The ACE2005 corpus includes annotations for 33 different event types and subtypes, offering us an opportunity to assess the generality of our methods across disparate event types. We selected 3 event types to report on here:

- **Die:** "occurs whenever the life of a PERSON Entity ends. It can be accidental, intentional or self-inflicted." This event appears 535 times in the corpus.
- **Attack:** "is defined as a violent physical act causing harm or damage." Attack events include a variety of sub-events like "person attack person", "country invade country", and "weapons attack locations". This event type appears 1120 times.
- **Start-Position:** "occurs whenever a PERSON Entity begins working for (or changes offices within) an ORGANIZATION or GPE. This includes government officials starting their terms, whether elected or appointed". It appears 116 times in the corpus.

We choose these three event types because they reflect the diversity of events ACE annotated: *die* events appear frequently in the ACE corpus and its definition is very clear; *attack* events also appear frequently, but its definition is rather complicated and contains several different sub-events; *start-position*'s definition is clear, but it is relatively infrequent in the corpus.

Based on the observations from the MUC-6 corpus, we eschewed corpus pre-selection for

[5] The pre-selection of relevant and irrelevant documents is based on document meta-data provided as part of the Reuters Corpus Volume I (Rose et al., 2002).

two reasons: first, building a different corpus for training each event type is an extra burden in developing a system for handling multiple events; second, we want to demonstrate that filtered ranking would work without pre-selection, while the document-centric method does not. As a result, we used the Gigaword AFP corpus for all event types.

In the ACE 2005 corpus, for every event, the annotators recorded a trigger, which is the main word that most clearly expresses an event occurrence. This added information allowed us to conduct dual evaluations: one based on sentence relevance - following S&G - presented in section 5.4.2, and one based on trigger identification, presented in section 5.4.3.

5.4.1 ACE2005 Supervised Model

To provide a benchmark for our semi-supervised learners, we built a very simple pattern-based supervised learning model. For training, for every pattern, we count how many times it contains an event trigger and how many times it does not. If more than 50% of the time it contains an event trigger, we treat it as a positive pattern.

For sentence level evaluation, if there is a positive pattern in a sentence, we tag this sentence as relevant; otherwise not. For word level evaluation, if the word is the predicator of a positive pattern, we tag it as a trigger; otherwise not[6].

We did a 5-fold cross-validation on the ACE 2005 data, report the average results and compare it to the semi-supervised learning method (see figure 3 & 4).

5.4.2 Sentence level ACE Event Evaluation[7]

Different event types have quite different performance (see figure 3): for the *die* event, the peak performance of all methods is quite good, and quite close to the supervised result; for the *attack* event, filtered ranking performs much better than both document and similarity-centric methods, but still worse than the supervised method; for *start-position* events, the semi-supervised method beats the supervised method. The reason might be as follows:

Die events appear frequently in ACE 2005, and most instances correspond to a small number of forms, so it is easy to find the correct patterns both from WordNet or related documents. As a result, filtered ranking provides no apparent benefit.

Attack is a more complicated event including several sub-events, which also have a lot of related events like *die* and *injure*. As a result, the document-centric method's performance goes down much faster, because patterns for related event types get drawn in; while the similarity-centric method performs worse than filtered ranking because some ambiguous words are introduced. For example, "hit" is an *attack* trigger, but words in the same Synset, such as "reach", "make", "attain", "gain" are quite dangerous because most of the time, these words do not refer to an attack event.

Start-position events do not appear frequently in ACE 2005, and supervised learning cannot achieve good performance because it can't collect enough training samples. The similarity-centric and Filter2 methods, which also depend on the ACE 2005 corpus, do not perform well either. Filter1 performs quite well because the Gigaword AFP corpus is quite large and contains more relevant documents, although the percentage is very small. This confirms our assumption that filtered ranking can achieve reasonable performance on a large unselected corpus, which is especially useful when the event is rare in the evaluation corpus.

<Arg1 kill Person>
<Arg1 slay Person>
<Arg1 death Person>

Table 2. Seeds for Ace 2005 *Die* evaluation

<Arg0 hire ORG>
<Arg1 hire Person>
<Arg1 appoint Person>
<Arg0 appoint ORG>

Table 3. Seeds for Ace 2005 *Start-Position* evaluation

[6] For word-level evaluation, we only consider trigger words with at least one semantic argument such as subject, object or a preposition; for that reason the performance is quite different from sentence level evaluation. We did the same for the word-level evaluation of semi-supervised learning.

[7] We do not list *Attack* seed patterns here as there are 34 patterns used.

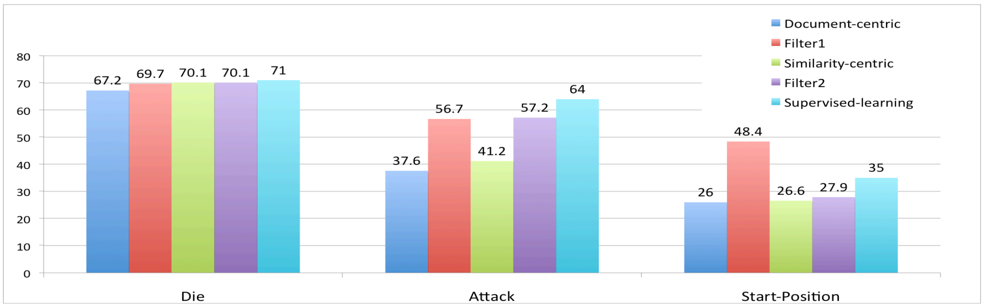


Figure 3. Performance on different ranking methods on ACE2005 sentence level evaluation

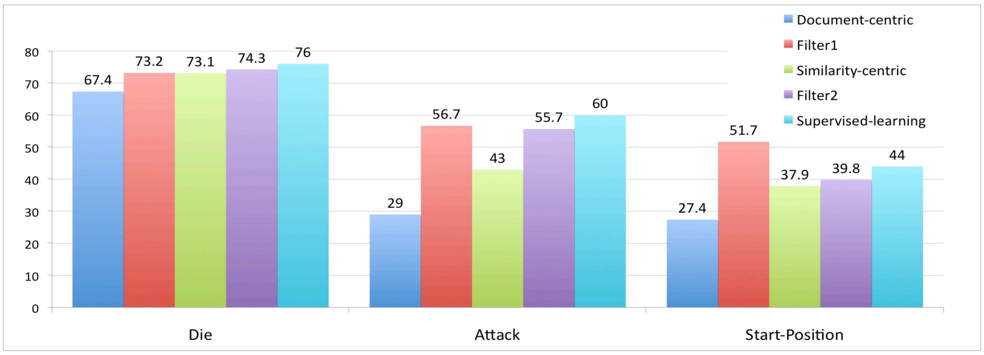


Figure 4. Performance on different ranking methods on ACE2005 word level evaluation

5.4.3 Word-level ACE Event Evaluation

Word-level evaluation is different from sentence-level evaluation because patterns which appear around an event but do not predicate an event are penalized in this evaluation. For example, the pattern <*Sbj chairman PERSON*>, which arises from a phrase like "*PERSON was the chairman of COMPANY*", appears much more in relevant *start-position* sentences than irrelevant sentences, and adding this pattern to the seeds will improve performance using the relevant-sentence metric. We would prefer a metric which discounted such patterns.

As noted above, ACE event annotations contain triggers, which are more specific event locators than a sentence, and we use this as the basis for a more specific evaluation. Extracted patterns are used to identify event triggers instead of identifying relevant sentences. For every word *w* in the ACE corpus, we extract all the patterns whose predicate is *w*. If the event extraction patterns include one of these patterns, we tag *w* as a trigger.

In word level evaluation, document-centric performs worse than the other methods. The reason is that some patterns appear often in the context of an event and are positive patterns for sentence level evaluation, but they do not actually predicate an event and are negative patterns in word level evaluation. In this situation, the document-centric method performs worse than the similarity-centric method, because it extracts many such patterns. For example, of the sentences which contain *die* events, 29% also contain *attack* events.

Thus in word level evaluation, filtered ranking continues to outperform either document- or similarity-centric methods, and its advantage over document-centric methods is accentuated.

6 Conclusions

In this paper, we propose a new ranking method in bootstrapping for event extraction and investigate the performance on different bootstrapping corpora with different ranking methods. This new method can block some irrelevant patterns coming from relevant documents, and, by preferring patterns from relevant documents, can eliminate some lexical ambiguity. Experiments show that this new ranking method performs better than previous ranking methods and is more stable across different corpora.

References

D. Gildea and D. Jurafsky. 2002. *Automatic Labeling of Semantic Roles.* Computational Linguistics, 28:245–288.

MA Greenwood, M. Stevenson. 2006. *Improving semi-supervised acquisition of relation extraction patterns.* Proceedings of the Workshop on Information Extraction Beyond the Document, pages 29–35.

Jay J. Jiang and David W. Conrath. 1997. *Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy.* In Proceedings of International Conference Research on Computational Linguistics (ROCLING X), Taiwan

C. Leacock and M. Chodorow. 1998. *Combining local context and WordNet similarity for word sense identification.* In C. Fellbaum, editor, WordNet: An electronic lexical database, pages 265–283. MIT Press.

D. Lin. 1998. *An information-theoretic definition of similarity.* In Proceedings of the International Conference on Machine Learning, Madison, August.

A. Meyers, M. Kosaka, N. Xue, H. Ji, A. Sun, S. Liao and W. Xu. 2009. Automatic Recognition of Logical Relations for English, Chinese and Japanese in the GLARF Framework. *In SEW-2009 (Semantic Evaluations Workshop) at NAACL HLT-2009*

MUC. 1995. Proceedings of the Sixth Message Understanding Conference (MUC-6), San Mateo, CA. Morgan Kaufmann.

Martha Palmer, Dan Gildea, and Paul Kingsbury, *The Proposition Bank: A Corpus Annotated with Semantic Roles,* Computational Linguistics, 31:1, 2005.

Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. 2003. *Using measures of semantic relatedness for word sense disambiguation.* In Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics, Mexico City, Mexico.

Patwardhan, S. and Riloff, E. 2007. *Effective Information Extraction with Semantic Affinity Patterns and Relevant Regions*. Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP-07)

T. Pedersen, S. Patwardhan, and J. Michelizzi. 2004. *WordNet::Similarity - Measuring the Relatedness of Concepts.* In Proceedings of the Nineteenth National Conference on Artificial Intelligence (Intelligent Systems Demonstrations), pages 1024-1025, San Jose, CA, July 2004.

P. Resnik. 1995. *Using information content to evaluate semantic similarity in a taxonomy.* In Proceedings of the 14th International Joint Conference on Artificial Intelligence, pages 448–453, Montreal, August.

Ellen Riloff. 1996. *Automatically Generating Extraction Patterns from Untagged Text.* In Proc. Thirteenth National Conference on Artificial Intelligence (AAAI-96), 1996, pp. 1044-1049.

T. Rose, M. Stevenson, and M. Whitehead. 2002. *The Reuters Corpus Volume 1 - from Yesterday's news to tomorrow's language resources.* In LREC-02, pages 827–832, La Palmas, Spain.

M. Stevenson and M. Greenwood. 2005. *A Semantic Approach to IE Pattern Induction.* Proceedings of ACL 2005.

Mihai Surdeanu, Jordi Turmo, and Alicia Ageno. 2006. *A Hybrid Approach for the Acquisition of Information Extraction Patterns.* Proceedings of the EACL 2006 Workshop on Adaptive Text Extraction and Mining (ATEM 2006)

Z. Wu and M. Palmer. 1994. *Verb semantics and lexical selection.* In 32nd Annual Meeting of the Association for Computational Linguistics, pages 133–138, Las Cruces, New Mexico.

Roman Yangarber; Ralph Grishman; Pasi Tapanainen; Silja Huttunen. 2000. *Automatic Acquisition of Domain Knowledge for Information Extraction.* Proc. COLING 2000.

Roman Yangarber. 2003. *Counter-Training in Discovery of Semantic Patterns.* Proceedings of ACL2003

Exploring variation across biomedical subdomains

Tom Lippincott and **Diarmuid Ó Séaghdha** and **Lin Sun** and **Anna Korhonen**
Computer Laboratory
University of Cambridge
{tl318,do242,ls418,alk23}@cam.ac.uk

Abstract

Previous research has demonstrated the importance of handling differences between domains such as "newswire" and "biomedicine" when porting NLP systems from one domain to another. In this paper we identify the related issue of *subdomain variation*, i.e., differences between subsets of a domain that might be expected to behave homogeneously. Using a large corpus of research articles, we explore how subdomains of biomedicine vary across a variety of linguistic dimensions and discover that there is rich variation. We conclude that an awareness of such variation is necessary when deploying NLP systems for use in single or multiple subdomains.

1 Introduction

One of the most noticeable trends in the past decade of Natural Language Processing (NLP) research has been the deployment of language processing technology to meet the information retrieval and extraction needs of scientists in other disciplines. This meeting of fields has proven mutually beneficial: scientists increasingly rely on automated tools to help them cope with the exponentially expanding body of publications in their field, while NLP researchers have been spurred to address new conceptual problems in theirs. Among the fundamental advances from the NLP perspective has been the realisation that tools which perform well on textual data from one source may fail to do so on another unless they are tailored to the new source in some way. This has led to significant interest in the idea of contrasting *domains* and the concomitant problem of *domain adaptation*, as well as the production of manually annotated domain-specific corpora.[1]

One definition of *domain variation* associates it with differences in the underlying probability distributions from which different sets of data are drawn (Daumé III and Marcu, 2006). The concept also mirrors the notion of variation across thematic subjects and the corpus-linguistic notions of *register* and *genre* (Biber, 1988). In addition to the differences in vocabulary that one would expect to observe, domains can vary in many linguistic variables that affect NLP systems. The scientific domain which has received the most attention (and is the focus of this paper) is the biomedical domain. Notable examples of corpus construction projects for the biomedical domain are PennBioIE (Kulick et al., 2004) and GENIA (Kim et al., 2003). These corpora have been used to develop systems for a range of processing tasks, from entity recognition (Jin et al., 2006) to parsing (Hara et al., 2005) to coreference resolution (Nguyen and Kim, 2008).

An implicit assumption in much previous work on biomedical NLP has been that particular subdomains of biomedical literature – typically molecular biology – can be used as a model of biomedical language in general. For example, GENIA consists of abstracts dealing with a specific set of subjects in molecular biology, while PennBioIE covers abstracts in two specialised domains, cancer genomics and the behaviour of a particular class of enzymes. This assumption of representativeness is understandable because linguistic annotation is labour-intensive and it may not be worthwhile to produce annotated corpora for multiple subdomains within a single discipline if there is lit-

[1] A workshop dedicated to domain adaptation is collocated with ACL 2010.

tle task-relevant variation across those subdomains. However, such conclusions should not be made before studying the actual degree of difference between the subdomains of interest.

One of the principal goals of this paper is to map how the concept of "biomedical language", often construed as a monolithic entity, is composed of diverse patterns of behaviour at more fine-grained topical levels. Hence we study linguistic variation in a broad biomedical corpus of abstracts and full papers, the PMC Open Access Subset.[2] We select a range of lexical and structural phenomena for quantitative investigation. The results indicate that common subdomains for resource development are not representative of biomedical text in general and furthermore that different linguistic features often partition the subdomains in quite different ways.

2 Related Work

A number of researchers have explored the differences between non-technical and scientific language. Biber and Gray (2010) describe two distinctive syntactic characteristics of academic writing which set it apart from general English. Firstly, in academic writing additional information is most commonly integrated by pre- and post-modification of phrases rather than by the addition of extra clauses. Secondly, academic writing places greater demands on the reader by omitting non-essential information, through the frequent use of passivisation, nominalisation and noun compounding. Biber and Gray also show that these tendencies towards "less elaborate and less explicit" language have become more pronounced in recent history.

We now turn to corpus studies that focus on biomedical writing. Verspoor et al. (2009) use measurements of lexical and structural variation to demonstrate that Open Access and subscription-based journal articles in a specific domain (mouse genomics) are sufficiently similar that research on the former can be taken as representative of the latter. While their primary goal is different from ours and they do not consider variation across multiple domains, they do compare their mouse genomics corpus with small reference corpora drawn from newswire and general biomedical sources. This analysis unsurprisingly finds differences between the domain and newswire corpora across many linguistic dimensions; more interestingly for our purposes, the comparison of domain text to the broader biomedical superdomain shows a more complex picture with similarities in some aspects (e.g., passivisation and negation) and dissimilarities in others (e.g., sentence length, semantic features).

Friedman et al. (2002) document the "sublanguages" associated with two biomedical domains: clinical reports and molecular biology articles. They set out restricted ontologies and frequent co-occurrence templates for the two domains and discuss the similarities and differences between them, but they do not perform any quantitative analysis.

Other researchers have focused on specific phenomena, rather than cataloguing a broad scope of variation. Cohen et al. (2008) carry out a detailed analysis of argument realisation with respect to verbs and nominalisations, using the GENIA and PennBioIE corpora. Nguyen and Kim (2008) compare the behaviour of anaphoric pronouns in newswire and biomedical corpora; they improve the performance of a pronoun resolver by incorporating their observations, thus demonstrating the importance of capturing domain-specific phenomena. Nguyen and Kim's findings are discussed in more detail in Section 5.4 below.

3 Subdomains in the OpenPMC Corpus

The Open Access Subset of PubMed (OpenPMC) is the largest publicly available corpus of full-text articles in the biomedical domain. OpenPMC is comprised of 169,338 articles drawn from 1233 medical journals, totalling approximately 400 million words. The NIH maintains a one-to-many mapping from journals to 122 subject areas (NIH, 2009b). This covers about 400 of the OpenPMC journals, but these account for over 70% of the database by byte size and word count. Journals are assigned up to five subject areas with the majority assigned one (69%) or two (26%) subjects. In this paper we adopt the OpenPMC subject areas (e.g. "Pulmonary Medicine", "Genetics", "Psychiatry") as the basis for subdomain comparison.

[2] http://www.ncbi.nlm.nih.gov/pmc/about/openftlist.html

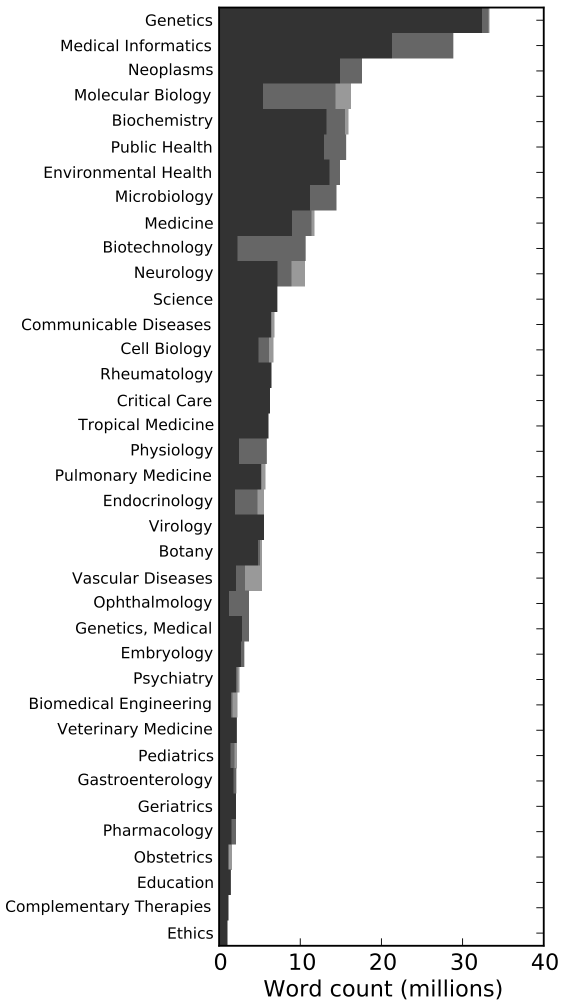


Figure 1: OpenPMC word count by subdomain, dark colouring indicates data assigned single subdomain, each lighter shade indicates an additional overlapping subdomain

4 Methodology

4.1 Data selection and preprocessing

An important initial question was how to treat data with multiple classifications: we only consider journals assigned a single subdomain, to avoid the added complexity of interactions in data from overlapping subdomains. To ensure sufficient data for comparing a variety of linguistic features, we discard the subdomains with less than one million words meeting the single-subdomain criterion. After review, we also drop the "Biology" subdomain, which appears to function as a catch-all for many loosely related areas. Figure 1 shows the distribution of data across the subjects we use, by word-count, with lighter-coloured areas representing data that is assigned multiple subjects. These subjects provide a convenient starting point for dividing the corpus into subdomains (hereafter, "subdomain" will be used rather than "subject"). We also add a reference subdomain, "Newswire", composed of a 6 million word random sample from the English Gigaword corpus (Graff et al., 2005). The final data set has a total of 39 subdomains.

Articles in the OpenPMC corpus are formatted according to a standard XML tag set (NIH, 2009a). We first convert each article to plain text, ignoring "non-content" elements such as tables and formulas, and split the result into sentences, aggregating the results by subdomain.

4.2 Feature extraction

We investigate subdomain variation in our corpus across a range of lexical, syntactic, sentential and discourse features. The corpus is lemmatised, tagged and parsed using the C&C pipeline (Curran et al., 2007) with the adapted part-of-speech and lexical category tagging models produced by Rimell and Clark (2009) for biomedical parsing.

From this output we count occurrences of noun, verb, adjective and adverb lemmas, part-of-speech (POS) tags, grammatical relations (GRs), chunks, and lexical categories. The lemma features are Zipfian-distributed items from an open class, so we have experimented with filtering low-frequency items at various thresholds to reduce noise and improve processing speed. The other feature sets can be viewed as closed classes, where filtering is unnecessary.

Since verbs are central to the meaning and structure of sentences, we consider their special behavior by constructing features for each verb's distribution over other grammatical properties. Several grammatical properties are captured by pairing each verb with its POS (indicating e.g. tense, such as present, past, and present participle). Voice is determined from additional annotation output by the C&C parser. Table 1 shows the POS-distribution for the verb "restrict", in two subdomains from the corpus. Finally, we record distributions over verb subcategorization frames (SCFs) taken by each verb, and over the GRs it participates in.

Subdomain	VB	VBG	VBN	VBP	VBZ
Medical Informatics	.35	.29	.06	.09	.21
Cell Biology	.14	.43	.05	.10	.29

Table 1: Distribution over POS tags for verb "restrict", in two subdomains

SCFs were extracted using a system of Preiss et al. (2007).

To facilitate a more robust and interpretable analysis of vocabulary differences, we estimate a "topic model" of the corpus with Latent Dirichlet Analysis (Blei et al., 2003) using the MALLET toolkit.[3] As preprocessing we divide the corpus into articles, removing stopwords and words shorter than 3 characters. The Gibbs sampling procedure is parameterised to induce 100 topics, each giving a coherent cluster of related words learned from the data, and to run for 1000 iterations. We collate the predicted distribution over topics for each article in a subdomain, weighted by article wordcount, to produce a topic distribution for the subdomain.

4.3 Measurements of divergence

Our goal is to illustrate the presence or absence of differences between the feature sets, and to do so we calculated the Jensen-Shannon divergence and the Pearson correlation. Jensen-Shannon divergence is a finite symmetric measurement of the divergence between probability distributions, while Pearson correlation quantifies the linear relationship between two real-valued samples.

The count-features are weighted, for a given subdomain, by the feature's log-likelihood between the subdomain's data and the rest of the corpus. Log-likelihood has been shown to perform well when comparing counts of potentially low-frequency features (Rayson and Garside, 2000) such as found in Zipfian-distributed data. This serves to place more weight in the comparison on items that are distinctive of the subdomain with respect to the entire corpus.

While the count-features are treated as a single distribution for the purposes of JSD, the verbwise-features are composed of many distributions, one for each verb lemma. Our approach is to combine the JSD of the verbs, weighted by the log-likelihood of the verb lemma between the two subdomains in question, and normalize the distances to the interval [0, 1]. Using the lemma's log-likelihood assumes that, when a verb's distribution behaves differently in a subdomain, its frequency changes as well.

We present the results as dendrograms and heat maps. Dendrograms are tree structures that illustrate the results of hierarchical clustering. We perform hierarchical clustering on the inter-subdomain divergences for each set of features. The algorithm begins with each instance (in our case, subdomains) as a singleton cluster, and repeatedly joins the two most similar clusters until all the data is clustered together. The order of these merges is recorded as a tree structure that can be visualized as a dendrogram in which the length of a branch represents the distance between its child nodes. Similarity between clusters is calculated using average distance between all members, known as "average linking".

Heat maps show the pairwise calculation of a metric in a grid of squares, where square (x, y) is shaded according to the value of $metric(sub_x, sub_y)$. For our measurements of JSD, black represents 0 (i.e. identical distributions) and white represents the metric's theoretical maximum of 1. We also inscribe the actual value inside each square. Dendrograms are tree structures that illustrate the hierarchical clustering procedure described above. The dendrograms present all 39 subdomains, while for readability the heatmaps present 12 subdomains selected for representativeness.

5 Results

Different thresholds for filtering low-frequency terms had little effect on the divergence measures, and served mainly to improve processing time. We therefore report results using a cutoff of 150 occurrences (over the entire 234 million word data set) and log-likelihood weights. The results of Pearson correlation and JSD show similar trends, and due to its specific design for comparing distributions we only report the latter.

[3] http://mallet.cs.umass.edu

5.1 Vocabulary and lexical features

Differences in vocabulary are what first comes to mind when describing subdomains. Word features are fundamental components for systems such as POS taggers and lexicalised parsers; one therefore expects that these systems will be affected by variation in lexical distributions. Figure 2a uses JSD calculated on each subdomain's distribution over 100 LDA-induced topics to compare vocabulary distributions. Subdomains related to molecular biology (Genetics, Molecular Biology) show the smallest divergences, an interesting fact since these are heavily used in building resources for BioNLP. The dendrogram shows a rough division into "public policy", "patient-centric", "applied" and "microscopic" subdomains, with the distance between unrelated subdomains such as Biochemistry and Pediatrics almost as large as their respective differences from Newswire.

We omit figures for variation over noun, verb and adjective lemmas due to space restrictions; in general, these correlate with the variation in LDA topics though there are some differences. Figure 2b shows JSD calculated on distributions over adverb lemmas. Part of the variation is due to characteristic markers of scientific argument ("therefore", "significantly", "statistically"). A more interesting factor is the coining of domain-specific adverbs, an example of the tendency in scientific text to use complex lexical items and premodifiers rather than additional clauses. This also has the effect of moving subdomain-specific objects and processes from verbs and nouns to adverbs. This behavior seems non-continuous, in that subdomains either make heavy, or almost no, use of it: for example, Pediatrics has no subdomain-specific items among the its ten top adverbs by log-likelihood, while Neoplasms has "histologically", "immunohistochemically" and "subcutaneously". These information-dense terms could prove useful for tasks like automatic curation of subdomain vocabularies, where they imply relationships between their components, the items they modify, etc.

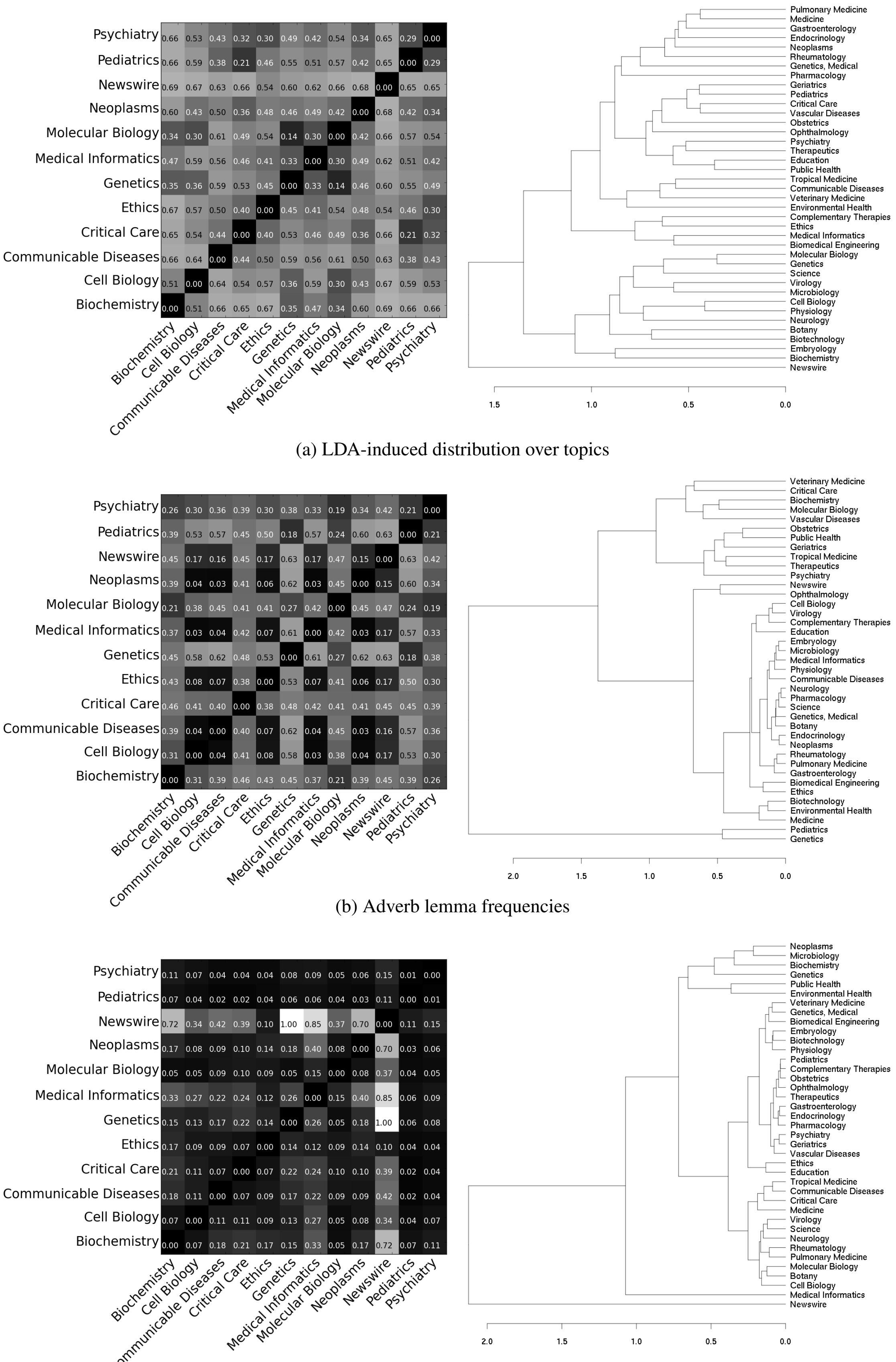
5.2 Verb distributional behavior

Modelling verb behavior is important for both syntactic (Collins, 2003) and semantic (Korhonen et al., 2008) processing, and subdomains are known to conscript verbs into specific roles that change the distributions of their syntactic properties (Roland and Jurafsky, 1998). The four properties we considered verbs' distributions over (SCF, POS, GR and voice) produced similar inter-subdomain JSD values. Figure 2c demonstrates how verbs differ between subdomains with respect to SCFs. For example, while the Pediatrics subdomain uses the verb "govern" in a single SCF among its 12 possibilities, the Genetics subdomain distributes its usage over 7 of them. Two subdomains may both use "restrict" with high frequency (e.g. Molecular Biology and Ethics), but with different frequency distributions over SCFs.

5.3 Syntax

It is difficult to measure syntactic complexity accurately without access to a hand-annotated treebank, but it is well-known that sentence length correlates strongly with processing difficulty (Collins, 1996). The first column of Table 2 gives average sentence lengths (excluding punctuation and "sentences" of fewer than three words) for selected domains. All standard errors are < 0.1. It is clear that all biomedical subdomains typically use longer sentences than newswire, though there is also variation within biomedicine, from an average length of 27 words in Molecular Biology to 24.5 words in Pediatrics.

"Packaging" information in complex pre- and/or post-modified noun phrases is a characteristic feature of academic writing (Biber and Gray, 2010). This increases the information density of a sentence but brings with it syntactic and semantic ambiguities. For example, the difficulty of resolving the internal structure of noun-noun compounds and strings of prepositional phrases has been the focus of ongoing research in NLP; these phenomena have also been identified as significant challenges in biomedical language processing (Rosario and Hearst, 2001; Schuman and Bergler, 2006). The second and third columns of Table 2 present average lengths for full noun phrases, defined as every word dominated by a head noun in the grammatical relation graph for a sentence, and for base nominals, defined as nouns plus premodifying adjectives and nouns only. All standard errors are ≤ 0.01. Newswire text uses the simplest noun



(a) LDA-induced distribution over topics

(b) Adverb lemma frequencies

(c) Verb distributions over subcategorization frames

Figure 2: Subdomain variation plotted as heat maps and dendrograms

Sentence length		Full NP length		Base nominal length	
Mol. Biology	27.0	Biochemistry	4.03	Biochemistry	1.85
Genetics	26.6	Genetics	3.90	Neoplasms	1.85
Cell Biology	26.3	Critical Care	3.86	Mol. Biology	1.84
Ethics	26.2	Neoplasms	3.85	Genetics	1.83
PMC Average	25.9	PMC Average	3.85	PMC Average	1.80
Biochemistry	25.8	Pediatrics	3.84	Cell Biology	1.80
Neoplasms	25.5	Med. Informatics	3.84	Critical Care	1.80
Psychiatry	25.3	Comm. Diseases	3.81	Med. Informatics	1.78
Critical Care	25.0	Therapeutics	3.80	Comm. Diseases	1.78
Therapeutics	24.9	Mol. Biology	3.79	Therapeutics	1.75
Comm. Diseases	24.9	Psychiatry	3.77	Psychiatry	1.75
Med. Informatics	24.6	Ethics	3.69	Pediatrics	1.73
Pediatrics	24.6	Cell Biology	3.55	Ethics	1.65
Newswire	19.1	Newswire	3.18	Newswire	1.60

Table 2: Average sentence, NP and base nominal lengths across domains

phrase structures; there is notable variation across PMC domains. Full NP and base nominal lengths do not always correlate; for example, Cell Biology uses relatively long base NPs (nominalisations and multitoken names in particular) but relatively simple full NP structures.

5.4 Coreference

Resolving coreferential terms is a crucial and challenging task when extracting information from texts in any domain. Nguyen and Kim (2008) compare the use of pronouns in the newswire and biomedical domains, using the GENIA corpus as representative of the latter. Among the differences observed between the domains were the absence of any personal pronouns other than third-person neuter pronouns in the GENIA corpus, and a greater proportion of demonstrative pronouns in GENIA than in the ACE or MUC newswire corpora. Corroborating the importance of domain modelling, Nguyen and Kim demonstrate that tailoring a pronoun resolution system to specific properties of the biomedical domain improves performance.

As our corpus is not annotated for coreference we restrict our attention to types that are reliably coreferential: masculine/feminine personal pronouns (*he*, *she* and case variations), neuter personal pronouns (*they*, *it* and variations) and definite NPs with demonstrative determiners such as *this* and *that*. To filter out pleonastic pronouns we used a combination of the C+C parser's pleonasm tag and heuristics based on Lappin and Leass (1994). To filter out the most common class of non-anaphoric demonstrative NPs we simply discarded any matching the pattern *this*...*paper*|*study*|*article*.

Table 3 presents statistics for selected types of coreferential noun phrases in a number of domains. The results generally agree with the findings of Nguyen and Kim (2008): biomedical text is on average 200 times less likely than news text to use gendered pronouns and twice as likely to use anaphoric definite noun phrases. At the domain level, however, there is clear variation within the biomedical corpus. In contrast to Nguyen and Kim's observations about GENIA some domains do make non-negligible use of gendered pronouns, most notably Ethics (usually to refer to other scholars) and domains such as Psychiatry and Pediatrics where studies of actual patients are common. All biomedical domains use demonstrative NPs more frequently than newswire and only one (Ethics) matches newswire for frequent use of neuter 3rd-person pronouns.

6 Conclusion

In this paper we have explored the phenomenon of linguistic variation at a finer-grained level than previous NLP research, focusing on subdomains

Pronouns (neuter, 3rd)		Pronouns (non-neuter, 3rd)		Demonstrative NPs	
Ethics	0.0658	Newswire	0.0591	Genetics	0.0275
Newswire	0.0607	Ethics	0.0037	Med. Informatics	0.0263
Therapeutics	0.0354	Pediatrics	0.0015	Biochemistry	0.0263
Med. Informatics	0.0346	Psychiatry	0.0009	Ethics	0.0260
Psychiatry	0.0342	Comm. Diseases	0.0009	Mol. Biology	0.0251
Pediatrics	0.0308	Therapeutics	0.0005	PMC Average	0.0226
PMC Average	0.0284	PMC Average	0.0005	Cell Biology	0.0210
Genetics	0.0275	Critical Care	0.0004	Comm. Diseases	0.0207
Critical Care	0.0272	Neoplasms	0.0002	Neoplasms	0.0205
Mol. Biology	0.0258	Med. Informatics	0.0002	Psychiatry	0.0201
Biochemistry	0.0251	Genetics	0.0001	Critical Care	0.0201
Neoplasms	0.0227	Mol. Biology	2.5×10^{-5}	Therapeutics	0.0192
Cell Biology	0.0217	Biochemistry	2.0×10^{-5}	Pediatrics	0.0191
Comm. Diseases	0.0213	Cell Biology	1.5×10^{-5}	Newswire	0.0118

Table 3: Frequency of coreferential types (proportion of all NPs) across domains

rather than traditional domains such as “newswire” and “biomedicine”. We have identified patterns of variation across dimensions of vocabulary, syntax and discourse that are known to be of importance for NLP applications. While the magnitude of variation between subdomains is unsurprisingly less pronounced than between coarser domains, subdomain variation clearly does exist and should be taken into account when considering the generalisability of systems trained and evaluated on specific subdomains, for example molecular biology.

Future work includes directly evaluating the effect of subdomain variation on practical tasks, investigating further dimensions of variation such as nominalisation usage and learning alternative subdomain taxonomies directly from the corpus text. Ultimately, we expect that a more nuanced understanding of subdomain effects will have tangible benefits for many applications of scientific language processing.

Acknowledgements

This work was supported by EPSRC grant EP/G051070/1, the Royal Society (AK) and a Dorothy Hodgkin Postgraduate Award (LS).

References

Biber, Douglas and Bethany Gray. 2010. Challenging stereotypes about academic writing: Complexity, elaboration, explicitness. *Journal of English for Academic Purposes*, 9(1):2–20.

Biber, Douglas. 1988. *Variation Across Speech and Writing*. Cambridge University Press, Cambridge.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Cohen, K. Bretonnel, Martha Palmer, and Lawrence Hunter. 2008. Nominalization and alternations in biomedical language. *PLoS ONE*, 3(9):e3158.

Collins, Michael John. 1996. A new statistical parser based on bigram lexical dependencies. In *Proceedings of ACL-96*, Santa Cruz, CA.

Collins, Michael. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Curran, James, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and Boxer. In *Proceedings of the ACL-07 Demo and Poster Sessions*, Prague, Czech Republic.

Daumé III, Hal and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Friedman, Carol, Pauline Kraa, and Andrey Rzhetsky. 2002. Two biomedical sublanguages: a description based on the theories of Zellig Harris. *Journal of Biomedical Informatics*, 35(4):222–235.

Graff, David, Junbo Kong, Ke Chen, and Kazuaki Maeda, 2005. *English Gigaword Corpus, 2nd Edition*. Linguistic Data Consortium.

Hara, Tadayoshi, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Adapting a probabilistic disambiguation model of an HPSG parser to a new domain. In *Proceedings of IJCNLP-05*, Jeju Island, South Korea.

Jin, Yang, Ryan T. McDonald, Kevin Lerman, Mark A. Mandel, Steven Carroll, Mark Y. Liberman, Fernando C. Pereira, Raymond S. Winters, and Peter S. White. 2006. Automated recognition of malignancy mentions in biomedical literature. *BMC Bioinformatics*, 7:492.

Kim, J.-D., T. Ohta, Y. Tateisi, and J. Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(Suppl. 1):i180–i182.

Korhonen, Anna, Yuval Krymolowski, and Nigel Collier. 2008. The choice of features for classification of verbs in biomedical texts. In *Proceedings of COLING-08*, Manchester, UK.

Kulick, Seth, Ann Bies, Mark Liberman, Mark Mandel, Ryan McDonald, Martha Palmer, Andrew Schein, Lyle Ungar, Scott Winters, and Pete White. 2004. Integrated annotation for biomedical information extraction. In *Proceedings of the HLT-NAACL-04 Workshop on Linking Biological Literature, Ontologies and Databases*, Boston, MA.

Lappin, Shalom and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

Nguyen, Ngan L.T. and Jin-Dong Kim. 2008. Exploring domain differences for the design of a pronoun resolution system for biomedical text. In *Proceedings of COLING-08*, Manchester, UK.

NIH. 2009a. Journal publishing tag set. http://dtd.nlm.nih.gov/publishing/.

NIH. 2009b. National library of medicine: Journal subject terms. http://wwwcf.nlm.nih.gov/serials/journals/index.cfm.

Preiss, Judita, E.J. Briscoe, and Anna Korhonen. 2007. A system for large-scale acquisition of verbal, nominal and adjectival subcategorization frames from corpora. In *Proceedings of ACL-07*, Prague, Czech Republic.

Rayson, Paul and Roger Garside. 2000. Comparing corpora using frequency profiling. In *Proceedings of the ACL-00 Workshop on Comparing Corpora*, Hong Kong.

Rimell, Laura and Stephen Clark. 2009. Porting a lexicalized-grammar parser to the biomedical domain. *Journal of Biomedical Informatics*, 42(5):852–865.

Roland, Douglas and Daniel Jurafsky. 1998. How verb subcategorization frequencies are affected by corpus choice. In *Proceedings of COLING-ACL-98*, Montreal, Canada.

Rosario, Barbara and Marti Hearst. 2001. Classifying the semantic relations in noun compounds via a domain-specific lexical hierarchy. In *Proceedings of EMNLP-01*, Pittsburgh, PA.

Schuman, Jonathan and Sabine Bergler. 2006. Postnominal prepositional phrase attachment in proteomics. In *Proceedings of the HLT-NAACL-06 BioNLP Workshop on Linking Natural Language and Biology*, New York, NY.

Verspoor, Karin, K Bretonnel Cohen, and Lawrence Hunter. 2009. The textual characteristics of traditional and Open Access scientific journals are similar. *BMC Bioinformatics*, 10:183.

Semantic Role Labeling for News Tweets

[1,2]Xiaohua Liu, [3]Kuan Li*, [4]Bo Han*, [2]Ming Zhou,
[2]Long Jiang, [3]Zhongyang Xiong and [2]Changning Huang
[1]School of Computer Science and Technology
Harbin Institute of Technology
[2]Microsoft Research Asia
[3]College of Computer Science
Chongqing University
[4]School of Software
Dalian University of Technology
{xiaoliu, v-kuli, v-bohan, mingzhou, longj}
@microsoft.com
zyxiong@cqu.edu.cn
v-cnh@microsoft.com

Abstract

News tweets that report what is happening have become an important real-time information source. We raise the problem of Semantic Role Labeling (SRL) for news tweets, which is meaningful for fine grained information extraction and retrieval. We present a self-supervised learning approach to train a domain specific SRL system to resolve the problem. A large volume of training data is automatically labeled, by leveraging the existing SRL system on news domain and content similarity between news and news tweets. On a human annotated test set, our system achieves state-of-the-art performance, outperforming the SRL system trained on news.

1 Introduction

Tweets are text messages up to 140 characters. Every day, more than 50 million tweets are generated by millions of Twitter users. According to the investigation by Pear Analytics (2009), about 4% tweets are related to news[1].

We divide news related tweets into two categories: those excerpted from news articles and those not. The former kind of tweets, hereafter called news excerpt, is formally written while the latter, hereafter called news tweet, varies in style and often is not grammatically correct. To understand the proportion of news tweets, we randomly selected 1000 tweets related to news, and got 865 news tweets. Following is an example of anews tweet, containing *oh, yea,* which usually appear in spoken language, and *:-(,* an emoticon.

oh yea and Chile earthquake the earth off it's axis according to NASA and shorten the day by a wee second :-((S1)

News tweets arean important information source because they keep reporting what is happening in real time. For example, the earthquake near Los Angeles that happened on Tuesday, July 29, 2008 was first reported through news tweets only seconds later than the outbreak of the quake. Official news did not emerge about this event until four minutes later. By then, "Earthquake" was trending on Twitter Search with thousands of updates[2].

However, it is a daunting task for people to find out information they are interested in from such a huge number of news tweets, thus motivating us to conduct some kind of information

[*] This work has been done while the author was visiting Microsoft Research Asia.

[1] http://blog.twitter.com/2010/02/measuring-tweets.html

[2] http://blog.twitter.com/2008/07/twitter-as-news-wire.html

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 698–706,
Beijing, August 2010

extraction such as event mining, where SRL plays a crucial role (Surdeanu et al., 2003). Considering Sentence 1, suppose the agent *earthquake* and the patient *day* for the predicate *shorten* are identified. Then it is straightforward to output the event *Chile earthquake shorten the day*, which captures the essential information encoded in this tweet.

Following Màrquez (2009), we define SRL for news tweets as the task of identifying the arguments of a given verb as predicate in a news tweet and assigning them semantic labels describing the roles they play for the predicate. To make our method applicable to general information extraction tasks, rather than only to some special scenarios such as arresting event extraction, we adopt general semantic roles, i.e., Agent(*A0*), Patient(*A1*), Location(*AM-LOC*), Temporal(*AM-TMP*),etc., instead of situation-specific roles (Fillmore et al., 2004) such as Suspect, Authorities, and Offense in an arrest frame.

Our first attempt is to directly apply the state-of-art SRL system (Meza-Ruiz and Riedel, 2009) that trained on the CoNLL 08 shared task dataset(Surdeanu et al., 2008), hereafter called SRL-BS, to news tweets. Not surprisingly, we observe its F1 score drops sharply from 75.5% on news corpus to 43.3% on our human annotated news tweets, owing much to the informal written style of news tweets.

Therefore, we have to build a domain specific SRL system for news tweets. Given the diversified styles of news tweets, building such a system requires a larger number of annotated news tweets, which are not available, and are not affordable for human labeling. We propose a novel method to automatically annotate news tweets, which leverages the existing resources of SRL for news domain, and content similarity between news and news tweets. We argue that the same event is likely to be reported by both news and news tweets, which results in content similarity between the news and news tweet. Further, we argue that the news and news tweets reporting the same event tend to have similar predicate-argument structures. We tested our assumptions on the event *Chile earthquake* that happened on Match 2nd, 2010. We got 261 news and 722 news tweets published on the same day that described this event. Sentence 2 and 3 are two examples of the news excerpts and Sentence 1 is one example of news tweets for this event.

Chile Earthquake Shortened Earth Day (S2)

Chile Earthquake Shortened Day (S3)

Obviously Sentence 1, 2 and 3 all have predicate "*shortened*" with the same A0 and A1 arguments. Our manually checking showed that in average each news tweet in those 993 samples had 2.4 news excerpts that had the same predicate-argument structures.

Our news tweet annotation approach consists of four steps. First, we submit hot queries to Twitter and for each query we obtain a list of tweets. Second, for each list of tweets, we single out news excerpts using heuristic rules and remove them from the list, conduct SRL on news excerpts using SRL-BS, and cluster them in terms of the similarity in content and predicate-argument structures. Third, for each list of tweets, we try to merge every remaining tweet into one news excerpt cluster according to its content similarity to the cluster. Those that can be put into one news group are regarded as news tweet. Finally, semantic structures of news excerpts are passed to the news tweet in the same group through word alignment.

Our domain specific SRL system is then trained on automatically constructed training data using the Conditional Random Field (CRF: Lafferty et al., 2001) learning framework. Our system is evaluated on a human labeled dataset, and achieves state-of-the-art performance, outperforming the baseline SRL-BS.

Our contributions can be summarized as follows:

1) We propose to conduct SRL for news tweets for fine grained information extraction and retrieval;
2) We present a semi-supervised learning approach to train a domain specific SRL system for news tweets, which outperforms SRL-BS and achieves the state-of-the-art performance on a human labeled dataset.

The rest of this paper is organized as follows: In the next section, we review related work. In Section 3 we detail key components of our approach. In Section 4, we setup experiments and evaluate the effectiveness of our method. Finally, Section 5 concludes and presents the future work.

2 Related Work

Our related work falls into two categories: SRL on news and domain adaption.

As for SRL on news, most researchers used the pipelined approach, i.e., dividing the task into several phases such as argument identification, argument classification, global inference, etc., and conquering them individually (Xue and Palmer, 2004; Koomen et al., 2005; Cohn and Blunsom, 2005; Punyakanok et al., 2008; Toutanova et al., 2005; Toutanova et al., 2008). Exceptions to the pipelined approach exist. Màrquez et al. (2005) sequentially labeled the words according to their positions relative to an argument (i.e., inside, outside or at the beginning of it). Carreras et al. (2004) and Surdeanu et al. (2007) jointly labeled all the predicates. Vickrey and Koller(2008) simplified the input sentence by hand-written and machine learnt rules before conducting SRL. Some other approaches simultaneously resolved all the sub-tasks by integrating syntactic parsing and SRL into a single model (Musillo and Merlo, 2006; Merlo and Musillo, 2008), or by using Markov Logic Networks (MLN, Richardson and Domingos, 2005) as the learning framework (Riedel and Meza-Ruiz, 2008; Meza-Ruiz and Riedel, 2009).

All the above approaches focus on sentences from news articles or other formal documents, and depend on human annotated corpus for training. To our knowledge, little study has been carried out on SRL for news tweets.

As for domain adaption, some researchers regarded the out-of-domain data as "prior knowledge" and estimated the model parameters by maximizing the posterior under this prior distribution, and successfully applied their approach to language modeling (Bacchiani and Roark, 2003) and parsing (Roark and Bacchiani, 2003). Daumé III and Marcu (2006) presented a novel framework by defining a "general domain" between the "truly in-domain" and "truly out-of-domain".

Unlike existing domain adaption approaches, our method is about adapting SRL system on news domain to the news tweets domain, two domains that differ in writing style but are linked through content similarity.

3 Our Method

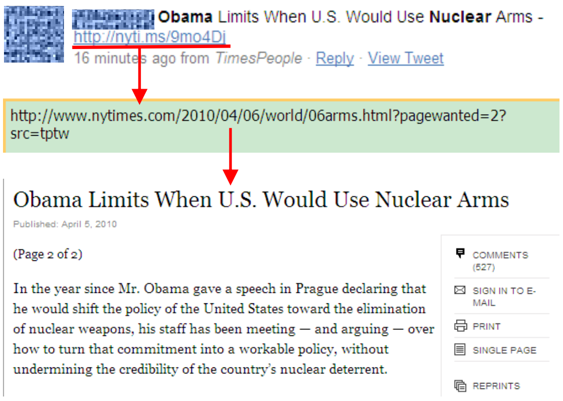
Our method of SRL for news tweets is to train a domain specific SRL on automatically annotated training data as briefed in Section 1.

In this section we present details of the five crucial components of our method, i.e., news excerpt identification, news excerpt clustering, news tweets identification, semantic structure mapping, and the domain specific SRL system constructing.

3.1 News Excerpt Identification

We use one heuristic rule to decide whether or not a tweet is news excerpt: if a tweet has a link to a news article and its text content is included by the news article, it is news excerpt, otherwise not.

Given a tweet, to apply this rule, we first extract the content link and expand it, if any, into the full link with the unshorten service[3]. This step is necessary because content link in tweet is usually shortened to reduce the total amount of characters. Next, we check if the full link points to any of the pre-defined news sites, which, in our experiments, are 57 English news websites. If yes, we download the web page and check if it exactly contains the text content of the input tweet. Figure 1 illustrates this process.



[redacted] **Obama** Limits When U.S. Would Use **Nuclear** Arms - http://nyti.ms/9mo4Dj

16 minutes ago from *TimesPeople* · Reply · View Tweet

http://www.nytimes.com/2010/04/06/world/06arms.html?pagewanted=2?src=tptw

Obama Limits When U.S. Would Use Nuclear Arms

Published: April 5, 2010

(Page 2 of 2)

In the year since Mr. Obama gave a speech in Prague declaring that he would shift the policy of the United States toward the elimination of nuclear weapons, his staff has been meeting — and arguing — over how to turn that commitment into a workable policy, without undermining the credibility of the country's nuclear deterrent.

Figure 1. An illustration of news excerpt identification.

To test the precision of this approach, while preparing for the training data for the experiments, we checked 100 tweets that were identified as news excerpt by this rule to find out they all are excerpted from news.

[3] http://unshort.me

3.2 News Excerpt Clustering

Given as input a list of news excerpts concerning the same query and published in the same time scope, this component uses the hierarchical agglomerative clustering algorithm (Manning et al., 2008) to divide news excerpts into groups in terms of the similarity in content and predicate-argument structures.

Before clustering, for every news excerpt, we remove the content link and other metadata such as author, retweet marks (starting with RT @), reply marks (starting with @ immediately after the author), hash tags (starting with #), etc., and keep only the text content; then it is further parsed into tokens, POS tags, chunks and syntactic tree using the OpenNLP toolkit[4]. After that, SRL is conducted with SRL-BS to get predicate-argument structures. Finally, every news excerpt is represented as frequency a vector of terms, including tokens, POS tagger, chunks, predicate-argument structures, etc. A news cluster is regarded as a "macro" news excerpt and is also represented as a term frequency vector, i.e., the sum of all the term vectors in the cluster. Noisy terms, such as numbers and predefined stop words are excluded from the frequency vector. To reduce data sparseness, words are stemmed by Porter stemmer (Martin F. Porter, 1980).

The cosine similarity is used to measure the relevance between two clusters, as defined in Formula 1.

$$CS\left(C,C^{'}\right)=\frac{CV\cdot CV^{'}}{\left\|CV\right\|\times\left\|CV^{'}\right\|} \tag{1}$$

Where *C, C'* denote two clusters, *CV, CV'* denote the term frequency vectors of *C* and *C'* respectively, and *CS(C, C')* stands for the cosine similarity between *C* and *C'*.

Initially, one news excerpt forms one cluster. Then the clustering process repeats merging the two most similar clusters into one till the similarity between any pair of clusters is below a threshold, which is experimentally set to 0.7 in our experiments.

During the training data preparation process, we randomly selected 100 clusters, each with 3.2 pieces of news in average. For every pair of news excerpts in the same cluster, we checked if they shared similar contents and semantic structures, and found out that 91.1% were the cases.

3.3 News Tweets Identification

After news excerpts are identified and removed from the list, every remaining tweet is checked if it is a news tweet. Here we group news excerpts and news tweets together in two steps because 1) news excerpts count for only a small proportion of all the tweets in the list, making our two-step clustering algorithm more efficient; and 2) one-step clustering tends to output meaningless clusters that include no news tweets.

Intuitively, news tweet, more often than not, have news counterparts that report similar contents. Thus we use the following rule to identify news tweets: if the content similarity between the tweet and any news excerpt cluster is greater than a threshold, which is experimentally set to 0.7 in our experiments, the tweet is a news tweet, otherwise it is not. Furthermore, each news tweet is merged into the cluster with most similar content. Finally, we re-label any news tweet as news excerpt, which is then process by SRL-BS, if its content similarity to the cluster exceeds a threshold, which is experimentally set to 0.9 in our experiments.

Again, the cosine similarity is used to measure the content similarity between tweet and news excerpt cluster. Each tweet is repressed as a term frequency vector. Before extracting terms from tweet, tweet metadata is removed and a rule-based normalization process is conducted to restore abnormal strings, say "&apos;", into their human friendly form, say ‘' '. Next, stemming tools and OpenNLP are applied to get lemmas, POS tags, chunks, etc., and noisy terms are filtered.

We evaluated the performance of this approach when preparing for the training data. We randomly sampled 500 tweets that were identified as news tweets, to find that 93.8% were true news tweets.

3.4 Semantic Structure Mapping

Semantic structure mapping is formed as the task of word alignment from news excerpt to news tweet. A HMM alignment model is trained with GIZA++ (Franz and Hermann, 2000) on all (news excerpt, news tweet) pairs in the same cluster. After word alignment is done, semantic

[4] http://opennlp.sourceforge.net/

information attached to a word in a news excerpt is passed to the corresponding word in the news tweet as illustrated in Figure 2.

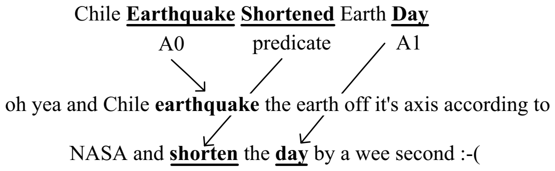

Chile **Earthquake** **Shortened** Earth **Day**
A0
predicate
A1
oh yea and Chile **earthquake** the earth off it's axis according to
NASA and **shorten** the **day** by a wee second :-(

Figure 2. An example of mapping semantic structures from news excerpts to news tweets.

In Figure 2, *shorten*, *earthquake* and *day* in two sentences are aligned, respectively; and two predicate-argument structures in the first sentence, i.e., *(shortened, earthquake, A0), (shortened, day, A1)*, are passed to the second.

News tweets may receive no semantic information from related news excerpts after mapping, because of word alignment errors or no news excerpt in the cluster with similar semantic structures. Such tweets are dropped.

Mapping may also introduce cases that violate the following two structural constraints in SRL (Meza-Ruiz and Riedel, 2009): 1) one (predicate, argument) pair has only one role label in one sentence; and 2) for each predicate, each of the proper arguments *(A0~A5)* can occur at most once. Those conflicts are largely owing to the noisy outputs of SRL trained on news and to the alignment errors. While preparing for the training data for our experiments, we found 38.9% of news tweets had such conflicts.

A majority voting schema and the structural constrains are used to resolve the conflicts as described below.

1) Step 1, for every cluster, each (*predicate*, *argument*, *role*) is weighted according to its frequency in the cluster;
2) Step 2, for every cluster, detect conflicts using the structural constrains; if no conflicts exist, stop; otherwise go to Step 3;
3) Step 3, for every cluster, keep the one with higher weight in each conflicting (*predicate*, *argument*, *role*) pair; if the weights are equal, drop both;

Here is an example to show the conflicting resolution process. Consider the cluster including Sentence 1, 2 and 3, where *(shorten, earthquake, A0), (shorten, earthquake, A1), (shorten, axis, A0),* and *(shorten, day, A1)* occur 6, 4, 1 and 3 times, respectively. This cluster includes three conflicting pairs:

1) *(shorten, earthquake, A0) vs. (shorten, earthquake, A1);*
2) *(shorten, earthquake, A1) vs. (shorten, day, A1);*
3) *(shorten, earthquake, A0) vs. (shorten, axis, A0);*

The first pair is first resolved, causing *(shorten, earthquake, A0)* to be kept and *(shorten, earthquake, A1)* removed, which leads to the second pair being resolved as well; then we process the third pair resulting in *(shorten, earthquake, A0)* being kept and *(shorten, axis, A0)* dropped; finally *(shorten, earthquake, A0)* and *(shorten, day, A1)* stay in the cluster.

The conflicting resolution algorithm is sensitive to the order of conflict resolution in Step 3. Still consider the three conflicting pairs listed above. If the second pair is first processed, only *(shorten, earthquake, A0)* will be left. Our strategy is to first handle the conflict resolving which leads to most conflicts resolved.

We tested the performance of this semantic structure mapping strategy while preparing for the training data. We randomly selected 56 news tweets with conflicts and manually annotated them with SRL. After the conflict resolution method was done, we observed that 38 news tweets were resolved correctly, 9 resolved but incorrectly, and 9 remain unresolved, suggesting the high precision of this method, which fits our task. We leave it to our future work to study more advanced approach for semantic structure mapping.

3.5 SRL System for News Tweets

Following Màrquez et al. (2005), we regard SRL for tweets as a sequential labeling task, because of its joint inference ability and its openness to support other languages.

We adopt conventional features for each token defined in Màrquez et al.(2005), such as the lemma/POS tag of the current/previous/next token, the lemma of predicate and its combination with the lemma/POS tag of the current token, the voice of the predicate (active/passive), the distance between the current token and the predicate, the relative position of the current token to the predicate, and so on. We do not use features related to syntactic parsing trees, to allow our system not to rely on any syntactic parser, whose performance depends on style and language of text, which limits the generality of our system.

Before extracting features, we perform a preprocessing step to remove tweet metadata and normalize tweet text content, as described in Section 3.3. The OpenNLP toolkit is used for feature extraction, and the CRF++ toolkit[5] is used to train the model.

4 Experiments

In this section, we evaluate our SRL system on a gold-standard dataset consisting of 1,110 human annotated news tweets and show that our system achieves the state-of-the-art performance compared with SRL-BS that is trained on news. Furthermore, we study the contribution of automatically generated training data.

4.1 Evaluation Metric

We adopt the widely used precision (Pre.), recall (Rec.) and F-score (F., the harmonic mean of precision and recall) as evaluation metrics.

4.2 Baseline System

We use SRL-BS as our baseline because of its state-of-art performance on news domain, and its readiness to use as well.

4.3 Data Preparation

We restrict to English news tweets to test our method. Our method can label news tweets of other languages, given that the related tools such as the SRL system on news domain, the word alignment tool, OpenNLP, etc., can support other languages.

We build two corpora for our experiments: one is the training dataset of 10,000 news tweets with semantic roles automatically labeled; the other is the gold-standard dataset of 1,110 news tweets with semantic roles manually labeled.

Training Dataset

We randomly sample 80 queries from 300 English queries extracted from the top stories of Bing news, Google news and Twitter trending topics from March 1, 2010 to March 4, 2010.

Submitting the 80 queries to Twitter search, we retrieve and download 512,000 tweets, from which we got 4,785 news excerpts and 11,427 news tweets, which were automatically annotated using the method described in Section 3.

Furthermore, 10,000 tweets are randomly selected from the automatically annotated news tweets, forming the training dataset, while the other 1,427 news tweets are used to construct the gold-standard dataset.

Gold-standard Dataset

We ask two people to annotate the 1,427 news tweets, following the Annotation guidelines for PropBank[6] with one exception: for phrasal arguments, only the head word is labeled as the argument, because our system and SRL-BS conduct word level SRL.

317 news tweets are dropped because of inconsistent annotation, and the remaining 1,110 news tweets form the gold-standard dataset.

Quality of Training dataset

Since the news tweets in the gold-standard dataset are randomly sampled from the automatically labeled corpus and are labeled by both human and machine, we use them to estimate the quality of training data, i.e., to which degree the automatically generated results are similar to humans'.

We find that our method achieves 75.6% F1 score, much higher than the baseline, suggesting the relatively high quality of the training data.

4.4 Result and Analysis

Table 1 reports the experimental results of our system (SRL-TS) and the baseline on the gold-standard dataset.

	Precision	Recall	F-Score
SRL-BS	36.0 %	54.5%	43.3%
SRL-TS	78.0%	57.1%	66.0%

Table 1. Performances of our system and the baseline on the gold-standard dataset.

As shown in Table 1, our system performs much better than the baseline on the gold-standard dataset in terms of all metrics. We observe two types of errors that are often made by

[5] http://crfpp.sourceforge.net/

[6] http://verbs.colorado.edu/~mpalmer/projects/ace/PBguidelines.pdf

SRL-BS but not so often by our system, which largely explains the difference in performance.

The first type of errors, which accounts for 25.3% of the total errors made by SRL-BS, is caused by the informal written style, such as ellipsis, of news tweets. For instance, for the example Sentence 1 listed in Section 1, the SRL-BS incorrectly identify earth as the A0 argument of the predicate shorten. The other type of errors, which accounts for 10.2% of the total errors made by SRL-BS, is related to the discretionary combination of news snippets. For example, consider the following news tweet:

The Chile earthquake shifted the earth's axis, "shortened the length of an Earth day by 1.26 miliseconds". (S4)

We analyze the errors made by our system and find that 12.5% errors are attributed to the complex syntactic structures, suggesting that combining our system with systems on news domain is a promising direction. For example, our system cannot identify the A0 argument of the predicate *shortened*, because of its blindness of attributive clause; in contrast, SRL-BS works on this case.

wow..the earthquake that caused the 2004 Indian Ocean tsunami shortened the day by almost 3 microseconds..what does that even mean?! HOW? (S5)

We also find that 32.3% of the errors made by our system are more or less related to the training data, which has noise and cannot fully represent the knowledge of SRL on news tweets. For instance, our system fails to label the following sentence, partially because the predicate *strike* does not occur in the training set.

8.8-Magnitude-Earthquake-Strikes-Chile (S6)

We further study how the size of automatically labeled training data affects our system's performance, as illustrated in Figure 3. We conduct two sets of experiments: in the first set, the training data is automatically labeled and the testing data is the gold-standard dataset; in the second set, half of the news tweets from the gold-standard dataset are added to the training data, the remaining half forms the testing dataset. Curve 1 and 2 represent the experimental results of set 1 and 2, respectively.

From Curve 1, we see that our system's performance increases sharply when the training data size varies from 5,000 to 6,000; then increases relatively slowly with more training data; and finally reaches the highest when all training data is used. Curve 2 reveals a similar trend.

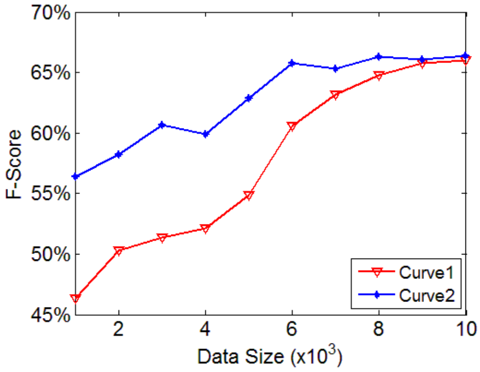


Figure 3. Performance on training data of varying size.

This phenomenon is largely due to the competing between two forces: the noise in the training data, and the knowledge of SRL encoded in the training data.

Interestingly, from Figure 3, we observe that the contribution of human labeled data is no longer significant after 6,000 automatically labeled training data is used, reaffirming the effectiveness of the training data.

5 Conclusions and Future Work

We propose to conduct SRL on news tweets for fine grained information extraction and retrieval. We present a self-supervised learning approach to train a domain specific SRL system for news tweets. Leveraging the SRL system on news domain and content similarity between news and news tweets, our approach automatically labels a large volume of training data by mapping SRL-BS generated results of news excerpts to news tweets. Experimental results show that our system outperforms the baseline and achieves the state-of-the-art performance.

In the future, we plan to enlarge training data size and test our system on a larger dataset; we also plan to further boost the performance of our system by incorporating tweets specific features such as hash tags, reply/re-tweet marks into our

CRF model, and by combining our system with SRL systems trained on news.

References

Bacchiani, Michiel and Brian Roark. 2003. Unsupervised language model adaptation. *Proceedings of the 2003 International Conference on Acoustics, Speech and Signal Processing,* volume 1, pages: 224-227

Carreras, Xavier, Lluís Màrquez, and Grzegorz Chrupała. 2004. Hierarchical recognition of propositional arguments with Perceptrons. *Proceedings of the Eighth Conference on Computational Natural Language Learning*, pages: 106-109.

Cohn, Trevor and Philip Blunsom. 2005. Semantic role labeling with tree conditional random fields. *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages: 169-172.

Daumé, Hal III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26(1), 101-126.

Fillmore, Charles J., Josef Ruppenhofer, Collin F. Baker. 2004. FrameNet and Representing the Link between Semantic and Syntactic Relations. *Computational Linguistics and Beyond, Institute of Linguistics, Academia Sinica.*

Kelly, Ryan, ed. 2009. *Twitter Study Reveals Interesting Results About Usage*. San Antonio, Texas: Pear Analytics.

Koomen, Peter, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized inference with multiple semantic role labeling systems. *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages: 181-184.

Lafferty, John D., Andrew McCallum, Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proceedings of the Eighteenth International Conference on Machine Learning*, pages: 282-289.

Manning, Christopher D., Prabhakar Raghavan and Hinrich Schtze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK.

Màrquez, Lluís, Jesus Gimènez Pere Comas and Neus Català. 2005. Semantic Role Labeling as Sequential Tagging. *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages: 193-196.

Màrquez, Lluís. 2009. *Semantic Role Labeling Past, Present and Future*, Tutorial of ACL-IJCNLP 2009.

Merlo, Paola and Gabriele Musillo. 2008. Semantic parsing for high-precision semantic role labelling. *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages: 1-8.

Meza-Ruiz, Ivan and Sebastian Riedel. 2009. Jointly Identifying Predicates, Arguments and Senses using Markov Logic. *Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL*, pages: 155-163.

Musillo, Gabriele and Paola Merlo. 2006. Accurate Parsing of the proposition bank. *Proceedings of the Human Language Technology Conference of the NAACL*, pages: 101-104.

Och, Franz Josef, Hermann Ney. Improved Statistical Alignment Models. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages: 440-447.

Porter, Martin F. 1980. An algorithm for suffix stripping. *Program,* 14(3), 130-137.

Punyakanok, Vasin, Dan Roth and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Journal of Computational Linguistics*, 34(2), 257-287.

Richardson, Matthew and Pedro Domingos. 2005. Markov logic networks. *Technical Report, University of Washington,* 2005.

Riedel, Sebastian and Ivan Meza-Ruiz. 2008. Collective semantic role labelling with Markov Logic. *Proceedings of the Twelfth Conference on Computational Natural Language Learning,* pages: 193-197.

Roark, Brian and Michiel Bacchiani. 2003. Supervised and unsupervised PCFG adaptation to novel domains. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology,* volume 1, pages: 126-133.

Surdeanu, Mihai, Sanda Harabagiu, JohnWilliams and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics,* volume 1, pages: 8-15.

Surdeanu, Mihai, Lluís Màrquez, Xavier Carreras and Pere R. Comas. 2007. Combination strategies for semantic role labeling. *Journal of Artificial Intelligence Research,* 29(1), 105-151.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. *Proceedings of the Twelfth Conference on Computational Natural Language Learning,* pages: 159-177.

Toutanova, Kristina, Aria Haghighi and Christopher D. Manning. 2005. Joint learning improves semantic role labeling. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics,* pages: 589-596.

Toutanova, Kristina, Aria Haghighi and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Journal of Computational Linguistics*, 34(2), 161-191.

Vickrey, David and Daphne Koller. 2008. Applying sentence simplification to the conll-2008 shared task. *Proceedings of the Twelfth Conference on Computational Natural Language Learning,* pages: 268-272

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. *Proceedings of the Conference on Empirical Methods in Natural Language Processing,* pages: 88-94.

Joint Parsing and Translation

Yang Liu and **Qun Liu**
Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
{yliu,liuqun} @ict.ac.cn

Abstract

Tree-based translation models, which exploit the linguistic syntax of source language, usually separate decoding into two steps: parsing and translation. Although this separation makes tree-based decoding simple and efficient, its translation performance is usually limited by the number of parse trees offered by parser. Alternatively, we propose to parse and translate jointly by casting tree-based translation as parsing. Given a source-language sentence, our joint decoder produces a parse tree on the source side and a translation on the target side simultaneously. By combining translation and parsing models in a discriminative framework, our approach significantly outperforms a forest-based tree-to-string system by 1.1 absolute BLEU points on the NIST 2005 Chinese-English test set. As a parser, our joint decoder achieves an F_1 score of 80.6% on the Penn Chinese Treebank.

1 Introduction

Recent several years have witnessed the rapid development of syntax-based translation models (Chiang, 2007; Galley et al., 2006; Shen et al., 2008; Quirk et al., 2005; Liu et al., 2006; Huang et al., 2006; Eisner, 2003; Zhang et al., 2008; Chiang, 2010), which incorporate formal or linguistic syntax into translation process. Depending on whether modeling the linguistic syntax of source language or not, we divide them into two categories: *string-based* and *tree-based* models. [1]

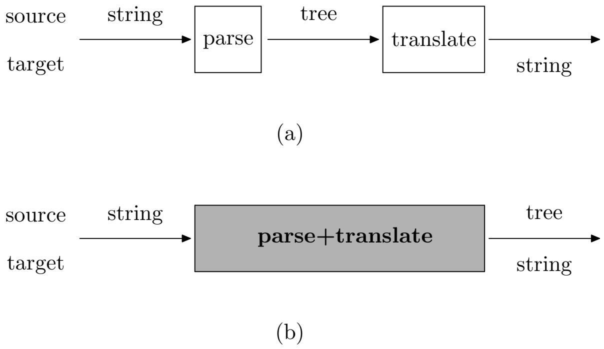


Figure 1: Tree-based decoding: (a) separate parsing and translation versus (b) joint parsing and translation.

String-based models include *string-to-string* (Chiang, 2007) and *string-to-tree* (Galley et al., 2006; Shen et al., 2008). Regardless of the syntactic information on the source side, they treat decoding as a parsing problem: the decoder parses a source-language sentence using the source projection of a synchronous grammar while building the target sub-translations in parallel.

Tree-based models include *tree-to-string* (Liu et al., 2006; Huang et al., 2006) and *tree-to-tree* (Quirk et al., 2005; Eisner, 2003; Zhang et al., 2008; Chiang, 2010). These models explicitly use source parse trees and divide decoding into two separate steps: parsing and translation. A parser first parses a source-language sentence into a parse tree, and then a decoder converts the tree to a translation on the target side (see Figure 1(a)).

Figure 2 gives a training example for tree-to-string translation, which consists of a Chinese tree, an English sentence, and the word alignment between them. Romanized Chinese words are given to facilitate identification. Table 1 shows

[1] Mi et al. (2008) also distinguish between string-based and tree-based models but depending on the type of input.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 707–715,
Beijing, August 2010

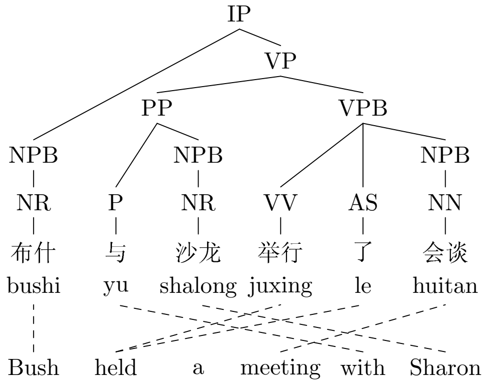


Figure 2: A training example that consists of a Chinese parse, an English sentence, and the word alignment between them.

(1)	IP(x_1:NPB VP(x_2:PP x_3:VPB))→x_1 x_3 x_2
(2)	NPB(NR(*bushi*))→*Bush*
(3)	PP(P(*yu*) x_1:NPB)→*with* x_1
(4)	NPB(NR(*shalong*))→*Sharon*
(5)	VPB(VV(*juxing*) AS(*le*) x_1:NPB)→*held a* x_1
(6)	NPB(NN(*huitan*))→*meeting*

Table 1: Tree-to-string rules extracted from Figure 2.

a set of tree-to-string rules obtained from Figure 2. The source side of a rule is a tree fragment and the target side is a string. We use x to denote non-terminals and the associated subscripts indicate the correspondence between non-terminals on both sides.

Conventionally, decoding for tree-to-string translation is cast as a *tree parsing* problem (Eisner, 2003). The tree parsing algorithm visits each node in the input source tree in a top-down order and tries to match each translation rule against the local sub-tree rooted at the node. For example, the first rule in Table 1 matches a sub-tree rooted at $\mathrm{IP}_{0,6}$ in Figure 2. The descendent nodes of $\mathrm{IP}_{0,6}$ (i.e., $\mathrm{NPB}_{0,1}$, $\mathrm{PP}_{1,3}$, and $\mathrm{VPB}_{3,6}$) can be further matched by other rules in Table 1. The matching procedure runs recursively until the entire tree is covered. Finally, the output on the target side can be taken as a translation.

Compared with its string-based counterparts, tree-based decoding is simpler and faster: there is no need for *synchronous binarization* (Huang et al., 2009b; Zhang et al., 2006) and tree parsing generally runs in linear time (Huang et al., 2006).

While separating parsing and translation makes tree-based decoding simple and efficient, its search space is limited by the number of parse trees offered by parser. Studies reveal that tree-based systems are prone to produce degenerate translations due to the propagation of parsing mistakes (Quirk and Corston-Oliver, 2006). This problem can be alleviated by offering more alternatives to the pipeline. An elegant solution is to replace 1-best trees with packed forests that encode exponentially many trees (Mi et al., 2008; Liu et al., 2009). Mi et al. (2008) present an efficient algorithm to match tree-to-string rules against packed forests that encode millions of trees. They prove that offering more alternatives to tree parsing improves translation performance substantially.

In this paper, we take a further step towards the direction of offering multiple parses to translation by proposing *joint parsing and translation*. As shown in Figure 1(b), our approach parses and translates jointly as it finds a parse tree and a translation of a source-language sentence simultaneously. We integrate the tree-to-string model (Liu et al., 2006; Huang et al., 2006), n-gram language model, probabilistic context-free grammar (PCFG), and Collins' Model 1 (Collins, 2003) in a discriminative framework (Och, 2003). Allowing parsing and translation to interact with each other, our approach obtains an absolute improvement of 1.1 BLEU points over a forest-based tree-to-string translation system (Mi et al., 2008) on the 2005 NIST Chinese-English test set. As a parser, our joint decoder achieves an F_1 score of 80.6% on the Penn Chinese Treebank.

2 Joint Parsing and Translation

2.1 Decoding as Parsing

We propose to integrate parsing and translation into a single step. To achieve joint parsing and translation, we cast tree-to-string decoding as a monolingual parsing problem (Melamed, 2004; Chiang, 2007; Galley et al., 2006): the decoder takes a source-language string as input and parses it using the source-projection of SCFG while building the corresponding sub-translations simultaneously.

For example, given the Chinese sentence *bushi yu sha long juxing le huitan* in Figure 2, the derivation in Table 1 explains how a Chinese tree, an English string, and the word alignment between them are generated synchronously. Unlike the string-based systems as described in (Chiang, 2007; Galley et al., 2006; Shen et al., 2008), we exploit the linguistic syntax on the source side explicitly. Therefore, the source parse trees produced by our decoder are meaningful from a linguistic point of view.

As tree-to-string rules usually have multiple non-terminals that make decoding complexity generally exponential, synchronous binarization (Huang et al., 2009b; Zhang et al., 2006) is a key technique for applying the CKY algorithm to parsing with tree-to-string rules. [2] Huang et al. (2009b) factor each tree-to-string rule into two SCFG rules: one from the root nonterminal to the subtree, and the other from the subtree to the leaves. In this way, one can uniquely reconstruct the original tree using a two-step SCFG derivation.

For example, consider the first rule in Table 1:

$$\mathrm{IP}(x_1\mathrm{:NPB\ VP}(x_2\mathrm{:PP}\ x_3\mathrm{:VPB}))\rightarrow x_1\ x_3\ x_2$$

We use a specific non-terminal, say, T, to uniquely identify the left-hand side subtree and produce two SCFG rules: [3]

$$\mathrm{IP} \rightarrow \langle \mathrm{T}_{\boxed{1}}, \mathrm{T}_{\boxed{1}} \rangle \quad (1)$$

$$\mathrm{T} \rightarrow \langle \mathrm{NPB}_{\boxed{1}}\mathrm{PP}_{\boxed{2}}\mathrm{VPB}_{\boxed{3}}, \mathrm{NPB}_{\boxed{1}}\mathrm{VPB}_{\boxed{3}}\mathrm{PP}_{\boxed{2}} \rangle \quad (2)$$

where the boxed numbers indicate the correspondence between nonterminals.

Then, the rule (2) can be further binarized into two rules that have at most two non-terminals:

$$\mathrm{T} \rightarrow \langle \mathrm{NPB}_{\boxed{1}}\ \mathrm{PP\text{-}VPB}_{\boxed{2}}, \mathrm{NPB}_{\boxed{1}}\ \mathrm{PP\text{-}VPB}_{\boxed{2}} \rangle \quad (3)$$

$$\mathrm{PP\text{-}VPB} \rightarrow \langle \mathrm{PP}_{\boxed{1}}\mathrm{VPB}_{\boxed{2}}, \mathrm{VPB}_{\boxed{2}}\mathrm{PP}_{\boxed{1}} \rangle \quad (4)$$

where $\mathrm{PP\text{-}VPB}$ is an intermediate *virtual non-terminal.*

We call rules the tree roots of which are virtual non-terminals *virtual rules* and others *natural rules*. For example, the rule (1) is a natural rule and the rules (3) and (4) are virtual rules. We follow Huang et al. (2009b) to keep the probabilities of a natural rule unchanged and set those of a virtual rule to 1. [4]

After binarizing tree-to-string rules into SCFG rules that have at most two non-terminals, we can use the CKY algorithm to parse a source sentence and produce its translation simultaneously as described in (Chiang, 2007; Galley et al., 2006).

2.2 Adding Parsing Models

As our decoder produces "genuine" parse trees during decoding, we can integrate parsing models as features together with translation features such as the tree-to-string model, n-gram language model, and word penalty into a discriminative framework (Och, 2003). We expect that parsing and translation could interact with each other: parsing offers linguistically motivated reordering to translation and translation helps parsing resolve ambiguity.

2.2.1 PCFG

We use the probabilistic context-free grammar (PCFG) as the first parsing feature in our decoder. Given a PCFG, the probability for a tree is the product of probabilities for the rules that it contains. That is, if a tree π is a context-free derivation that involves K rules of the form $\alpha_k \rightarrow \beta_k$, its probability is given by

$$\mathcal{P}(\pi) = \prod_{k=1...K} \mathcal{P}_{pcfg}(\alpha_k \rightarrow \beta_k) \quad (5)$$

For example, the probability for the tree in Figure 2 is

$$\begin{aligned}\mathcal{P}(\pi) \quad = \quad & \mathcal{P}_{pcfg}(\mathrm{IP} \rightarrow \mathrm{NPB\ VP}) \times \\ & \mathcal{P}_{pcfg}(\mathrm{NPB} \rightarrow \mathrm{NR}) \times \\ & \mathcal{P}_{pcfg}(\mathrm{NR} \rightarrow \textit{bushi}) \times \\ & \ldots \end{aligned} \quad (6)$$

[2] But CKY is not the only choice. The Earley algorithm can also be used to parse with tree-to-string rules (Zhao and Al-Onaizan, 2008). As the Earley algorithm binarizes multi-nonterminal rules implicitly, there is no need for synchronous binarization.

[3] It might look strange that the node VP disappears. This node is actually stored in the monolithic node T. Please refer to page 573 of (Huang et al., 2009b) for more details about how to convert tree-to-string rules to SCFG rules.

[4] This makes the scores of hypotheses in the same chart cell hardly comparable because some hypotheses are covered by a natural non-terminal and others covered by a virtual non-terminal. To alleviate this problem, we follow Huang et al. (2009b) to separate natural and virtual hypotheses in different beams.

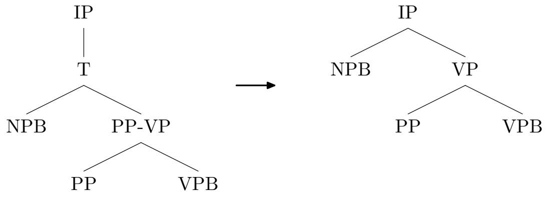


Figure 3: Reconstructing original tree from virtual rules. We first construct the tree on the left by substituting the trees of the rules (1), (3), and (4) and then restore the original tree on the right via the monolithic node T.

There are 13 PCFG rules involved. We omit the remaining 10 rules.

We formalize the decoding process as a deductive system to illustrate how to include a PCFG. Given a natural rule

$$\mathrm{VP} \rightarrow \langle \mathrm{PP}_{\boxed{1}}\ \mathrm{VPB}_{\boxed{2}}, \mathrm{VPB}_{\boxed{2}}\ \mathrm{PP}_{\boxed{1}} \rangle \tag{7}$$

the following deductive step grows an item in the chart by the rule

$$\frac{(\mathrm{PP}_{1,3}) : (w_1, e_1) \quad (\mathrm{VPB}_{3,6}) : (w_2, e_2)}{(\mathrm{VP}_{1,6}) : (w, e_2 e_1)} \tag{8}$$

where $\mathrm{PP}_{1,3}$ denotes the recognition of the non-terminal PP spanning from the substring from position 1 through 3 (i.e., *yu shalong* in Figure 2), w_1 and e_1 are the score and translation of the first antecedent item, respectively, and the resulting item score is calculated as: [5]

$$w = w_1 + w_2 + \log \mathcal{P}_{pcfg}(\mathrm{VP} \rightarrow \mathrm{PP\ VPB}) \tag{9}$$

As the PCFG probabilities of natural rules are fixed during decoding, they can be pre-computed and stored in the rule table. Therefore, including PCFG for natural rules hardly increases decoding complexity.

However, calculating the PCFG probabilities for virtual rules is quite different due to the presence of virtual non-terminals. For instance, using the rule (4) in Section 2.1 to generate an item leads to the following deductive step:

$$\frac{(\mathrm{PP}_{1,3}) : (w_1, e_1) \quad (\mathrm{VPB}_{3,6}) : (w_2, e_2)}{(\text{PP-VPB}_{1,6}) : (w, e_2 e_1)} \tag{10}$$

As PP-VPB is a virtual non-terminal, the subtree it dominates is a virtual tree, for which we cannot figure out its PCFG probability. Therefore, we have to postpone the calculation of PCFG probabilities until reaching a natural non-terminal such as IP. In other words, only when using the rule (1) to produce an item, the decoding algorithm can update PCFG probabilities because the original tree can be restored from the special node T now. Figure 3 shows how to reconstruct the original tree from virtual rules. We first construct the tree on the left by substituting the trees of the rules (1), (3), and (4) and then restore the original tree on the right via T. Now, we can calculate the PCFG probability of the original tree. [6] In practice, we pre-compute this PCFG probability and store it in the rule (1) to reduce computational overhead.

2.2.2 Lexicalized PCFG

Although widely used in natural language processing, PCFGs are often criticized for the lack of lexicalization, which is very important to capture the lexical dependencies between words. Therefore, we use Collins' Model 1 (Collins, 2003), a simple and effective lexicalized parsing model, as the second parsing feature in our decoder.

Following Collins (2003), we first lexicalize a tree by associating a *headword* h with each non-terminal. Figure 4 gives the lexicalized tree corresponding to Figure 2. The left-hand side of a rule in a lexicalized PCFG is $P(h)$ and the right-hand side has the form:

$$L_n(l_n) \dots L_1(l_1) H(h) R_1(\tau_1) \dots R_m(\tau_m) \tag{11}$$

where H is the head-child that inherits the headword h from its parent P, $L_1 \dots L_n$ and $R_1 \dots R_m$ are left and right modifiers of H, and $l_1 \dots l_n$ and $\tau_1 \dots \tau_m$ are the corresponding headwords. Either n or m may be zero, and $n = m = 0$ for unary rules. Collins (2003) extends the left and right sequences to include a terminating STOP symbol. Thus, $L_{n+1} = R_{m+1} = \mathrm{STOP}$.

[5] The logarithmic form of probability is used to avoid manipulating very small numbers for practical reasons. w_1 and w_2 take the PCFG probabilities of the two antecedent items into consideration.

[6] Postponing the calculation of PCFG probabilities also leads to the "hard-to-compare" problem mentioned in footnote 4 due to the presence of virtual non-terminals. We still maintain multiple beams for natural and virtual hypotheses (i.e., items) to alleviate this prblem.

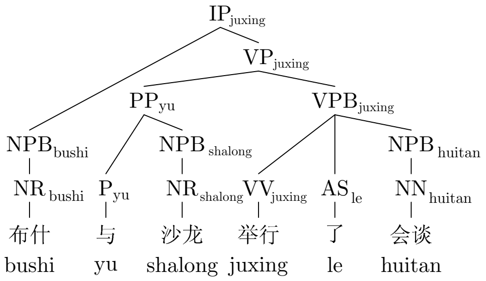


Figure 4: The lexicalized tree corresponding to Figure 2.

Collins (2003) breaks down the generation of the right-hand side of a rule into a sequence of smaller steps. The probability of a rule is decomposed as:

$$\begin{aligned}&\mathcal{P}_h(H|P(h)) \times \\ &\prod_{i=1...n+1} \mathcal{P}_l(L_i(l_i)|P(h),H,t,\Delta) \times \\ &\prod_{j=1...m+1} \mathcal{P}_r(R_j(\tau_j)|P(h),H,t,\Delta)\end{aligned} \quad (12)$$

where t is the POS tag of of the headword h and Δ is the distance between words that captures head-modifier relationship.

For example, the probability of the lexicalized rule $\text{IP}(\textit{juxing}) \rightarrow \text{NPB}(\textit{bushi})\ \text{VP}(\textit{juxing})$ can be computed as [7]

$$\begin{aligned}&\mathcal{P}_h(\text{VP}|\text{IP},\ \textit{juxing}) \times \\ &\mathcal{P}_l(\text{NPB}(\textit{bushi})|\text{IP},\text{VP},\ \textit{juxing}) \times \\ &\mathcal{P}_l(\text{STOP}|\text{IP},\text{VP},\ \textit{juxing}) \times \\ &\mathcal{P}_r(\text{STOP}|\text{IP},\text{VP},\ \textit{juxing})\end{aligned} \quad (13)$$

We still use the deductive system to explain how to integrate the lexicalized PCFG into the decoding process. Now, Eq. (8) can be rewritten as:

$$\frac{(\text{PP}^{\text{yu}}_{1,3}) : (w_1, e_1) \quad (\text{VPB}^{\text{juxing}}_{3,6}) : (w_2, e_2)}{(\text{VP}^{\text{juxing}}_{1,6}) : (w, e_2 e_1)} \quad (14)$$

where *yu* and *juxing* are the headwords attached to $\text{PP}_{1,3}$, $\text{VPB}_{3,6}$, and $\text{VP}_{1,6}$. The resulting item score is given by

$$\begin{aligned} w \;=\; & w_1 + w_2 + \log\mathcal{P}_h(\text{VPB}|\text{VP},\ \textit{juxing}) + \\ & \log\mathcal{P}_l(\text{PP}(\textit{yu})|\text{VP},\text{VPB},\ \textit{juxing}) + \\ & \log\mathcal{P}_l(\text{STOP}|\text{VP},\text{VPB},\ \textit{juxing}) + \\ & \log\mathcal{P}_r(\text{STOP}|\text{VP},\text{VPB},\ \textit{juxing})\end{aligned} \quad (15)$$

Unfortunately, the lexicalized PCFG probabilities of most natural rules cannot be pre-computed because the headword of a non-terminal must be determined on the fly during decoding. Consider the third rule in Table 1

$$\text{PP}(\text{P}(\textit{yu})\ x_1\text{:NPB}) \rightarrow \textit{with}\ x_1$$

It is impossible to know what the headword of NPB is in advance, which depends on the actual sentence being translated. However, we could safely say that the headword attached to PP is always *yu* because PP should have the same headword with its child P.

Similar to the PCFG scenario, calculating lexicalized PCFG for virtual rules is different from natural rules. Consider the rule (4) in Section 2.1, the corresponding deductive step is

$$\frac{(\text{PP}^{\text{yu}}_{1,3}) : (w_1, e_1) \quad (\text{VPB}^{\text{juxing}}_{3,6}) : (w_2, e_2)}{(\text{PP-VPB}^{-}_{1,6}) : (w, e_2 e_1)} \quad (16)$$

where "$-$" denotes that the headword of $\text{PP-VPB}_{1,6}$ is undefined.

We still need to postpone the calculation of lexicalized PCFG probabilities until reaching a natural non-terminal such as IP. In other words, only when using the rule (1) to produce an item, the decoding algorithm can update the lexicalized PCFG probabilities. After restoring the original tree from T, we need to visit backwards to frontier nodes of the tree to find headwords and calculate lexicalized PCFG probabilities. More specifically, updating lexicalized PCFG probabilities for the rule the rule (1) involves the following steps:

1. Reconstruct the original tree from the rules (1), (3), and (4) as shown in Figure 3;

2. Attach headwords to all nodes;

3. Calculate the lexicalized PCFG probabilities according to Eq. (12).

[7] For simplicity, we omit POS tag and distance in the presentation. In practice, we implemented the Collins' Model 1 exactly as described in (Collins, 2003).

Back-off level	$\mathcal{P}_h(H\|\dots)$	$\mathcal{P}_l(L_i(l_i)\|\dots)$ $\mathcal{P}_r(R_j(\tau_j)\|\dots)$
1	P, h, t	P, H, h, t, Δ
2	P, t	P, H, t, Δ
3	P	P, H, Δ

Table 2: The conditioning variables for each level of back-off.

As suggested by Collins (2003), we use back-off smoothing for sub-model probabilities during decoding. Table 2 shows the various levels of back-off for each type of parameter in the lexicalized parsing model we use. For example, $\mathcal{P}_h$ estimation p interpolates maximum-likelihood estimates $p_1 = \mathcal{P}_h(H|P, h, t)$, $p_2 = \mathcal{P}_h(H|P, t)$, and $p_3 = \mathcal{P}_h(H|P)$ as follows:

$$p_1 = \lambda_1 p_1 + (1 - \lambda_1)(\lambda_2 p_2 + (1 - \lambda_2) p_3) \quad (17)$$

where λ_1, λ_2, and λ_3 are smoothing parameters.

3 Experiments

In this section, we try to answer two questions:

1. Does tree-based translation by parsing outperform the conventional tree parsing algorithm? (Section 3.1)

2. How about the parsing performance of the joint decoder? (Section 3.2)

3.1 Translation Evaluation

We used a bilingual corpus consisting of 251K sentences with 7.3M Chinese words and 9.2M English words to extract tree-to-string rules. The Chinese sentences in the bilingual corpus were parsed by an in-house parser (Xiong et al., 2005), which obtains an F_1 score of 84.4% on the Penn Chinese Treebank. After running GIZA++ (Och and Ney, 2003) to obtain word alignments, we used the GHKM algorithm (Galley et al., 2004) and extracted 11.4M tree-to-string rules from the source-side parsed, word-aligned bilingual corpus. Note that the bilingual corpus does not contain the bilingual version of Penn Chinese Treebank. In other words, all tree-to-string rules were learned from noisy parse trees and alignments. We used the SRILM toolkit (Stolcke, 2002) to train a 4-gram language model on the Xinhua portion of the GIGAWORD corpus, which contains 238M English words. We trained PCFG and Collins' Model 1 on the Penn Chinese Treebank.

We used the 2002 NIST MT Chinese-English test set as the development set and the 2005 NIST test set as the test set. Following Huang (2008), we modified our in-house parser to produce and prune packed forests on the development and test sets. There are about 105M parse trees encoded in a forest of a sentence on average. We also extracted 1-best trees from the forests.

As the development and test sets have many long sentences ($\geq$ 100 words) that make our decoder prohibitively slow, we divided long sentences into short sub-sentences simply based on punctuation marks such as comma and period. The source trees and target translations of sub-sentences were concatenated to form the tree and translation of the original sentence.

We compared our parsing-based decoder with the tree-to-string translation systems based on the tree parsing algorithm, which match rules against either 1-best trees (Liu et al., 2006; Huang et al., 2006) or packed forests (Mi et al., 2008). All the three systems used the same rule set containing 11.4M tree-to-string rules. Given the 1-best trees of the test set, there are 1.2M tree-to-string rules that match fragments of the 1-best trees. For the forest-based system (Mi et al., 2008), the number of filtered rules increases to 1.9M after replacing 1-best trees with packed forests, which contain 105M trees on average. As our decoder takes a string as input, 7.7M tree-to-string rules can be used to parse and translate the test set. We binarized 99.6% of tree-to-string rules into 16.2M SCFG rules and discarded non-binarizable rules. As a result, the search space of our decoder is much larger than those of the tree parsing counterparts.

Table 3 shows the results. All the three systems used the conventional translation features such as relative frequencies, lexical weights, rule count, n-gram language model, and word count. Without any parsing models, the tree-based system achieves a BLEU score of 29.8. The forest-based system outperforms the tree-based system by $+1.8$ BLEU points. Note that each hyperedge

Algorithm	Input	Parsing model	# of rules	BLEU (%)	Time (s)
tree parsing	tree	-	1.2M	29.8	0.56
	forest	PCFG	1.9M	31.6	9.49
parsing	string	-	7.7M	32.0	51.41
		PCFG		32.4	55.52
		Lex		32.6	89.35
		PCFG + Lex		**32.7**	91.72

Table 3: Comparison of tree parsing and parsing for tree-to-string translation in terms of *case-insensitive* BLEU score and average decoding time (second per sentence). The column "parsing model" indicates which parsing models were used in decoding. We use "-" to denote using only translation features. "Lex" represents the Collins' Model 1. We excluded the extra parsing time for producing 1-best trees and packed forests.

Forest size	Exact match (%)	Precision (%)
1	0.55	41.5
390	0.74	47.7
5.8M	0.92	54.1
66M	1.48	62.0
105M	2.22	65.9

Table 4: Comparison of 1-best trees produced by our decoder and the parse forests produced by the monolingual Chinese parser. Forest size represents the average number of trees stored in a forest.

in a parse forest is assigned a PCFG probability. Therefore, the forest-based system actually includes PCFG as a feature (Mi et al., 2008). Without incorporating any parsing models as features, our joint decoder achieves a BLEU score of 32.0. Adding PCFG and Collins' Model 1 (i.e., "Lex" in Table 2) increases translation performance. When both PCFG and Collins' Model 1 are used, our joint decoder outperforms the tree parsing systems based on 1-best trees ($+2.9$) and packed forests ($+1.1$) significantly ($p < 0.01$). This result is also better than that of using only translation features significantly (from 32.0 to 32.7, $p < 0.05$).

Not surprisingly, our decoder is much slower than pattern matching on 1-best trees and packed forests (with the same beam size). In particular, including Collins' Model 1 increases decoding time significantly because its sub-model probabilities requires back-off smoothing on the fly.

How many 1-best trees produced by our decoder are included in the parse forest produced by a standard parser? We used the Chinese parser to generate five pruned packed forests with different sizes (average number of trees stored in a forest). As shown in Table 4, only 2.22% of the trees produced by our decoder were included in the biggest forest. One possible reason is that we used sub-sentence division to reduce decoding complexity. To further investigate the matching rate, we also calculated labeled precision, which indicates how many brackets in the parse match those in the packed forest. The labeled precision on the biggest forest is 65.9%, suggesting that the 1-best trees produced by our decoder are significantly different from those in the packed forests produced by a standard parser. [8]

3.2 Parsing Evaluation

We followed Petrov and Klein (2007) to divide the Penn Chinese Treebank (CTB) version 5 as follows: Articles 1-270 and 400-1151 as the training set, Articles 301-325 as the development set, and Articles 271-300 as the test set. We used max-F_1 training (Och, 2003) to train the feature weights. We did not use sub-sentence division as the sentences in the test set have no more than 40 words.

[8] The packed forest produced by our decoder ("rule" forest) might be different from the forest produced by a monolingual parser ("parser" forest). While tree-based and forest-based decoders search in the intersection of the two forests (i.e., matched forest), our decoder directly explores the "rule" forest, which represents the true search space of tree-to-string translation. This might be the key difference of our approach from forest-based translation (Mi et al., 2008). As sub-sentence division makes direct comparison of the two forests quite difficult, we leave this to future work.

Parsing model	F_1 (%)	Time (s)
-	62.7	23.9
PCFG	65.4	24.7
Lex	79.8	48.8
PCFG + Lex	$\mathbf{80.6}$	50.4

Table 5: Effect of parsing models on parsing performance ($\leq$ 40 words) and average decoding time (second per sentence). We use "-" to denote only using translation features.

Table 5 shows the results. Translation features were used for all configurations. Without parsing models, the F_1 score is 62.7. Adding Collins' Model 1 results in much larger gains than adding PCFG. With all parsing models integrated, our joint decoder achieves an F_1 score of 80.6 on the test set. Although lower than the F_1 score of the in-house parser that produces the noisy training data, this result is still very promising because the tree-to-string rules that construct trees in the decoding process are learned from noisy training data.

4 Related Work

Charniak et al. (2003) firstly associate lexicalized parsing model with syntax-based translation. They first run a string-to-tree decoder (Yamada and Knight, 2001) to produce an English parse forest and then use a lexicalized parsing model to select the best translation from the forest. As the parsing model operates on the target side, it actually serves as a syntax-based language model for machine translation. Recently, Shen et al. (2008) have shown that dependency language model is beneficial for capturing long-distance relations between target words. As our approach adds parsing models to the source side where the source sentence is fixed during decoding, our decoder does parse the source sentence like a monolingual parser instead of a syntax-based language model. More importantly, we integrate translation models and parsing models in a discriminative framework where they can interact with each other directly.

Our work also has connections to joint parsing (Smith and Smith, 2004; Burkett and Klein, 2008) and bilingually-constrained monolingual parsing (Huang et al., 2009a) because we use another language to resolve ambiguity for one language. However, while both joint parsing and bilingually-constrained monolingual parsing rely on the target sentence, our approach only takes a source sentence as input.

Blunsom and Osborne (2008) incorporate the source-side parse trees into their probabilistic SCFG framework and treat every source-parse PCFG rule as an individual feature. The difference is that they parse the test set before decoding so as to exploit the source syntactic information to guide translation.

More recently, Chiang (2010) has shown that ("exact") tree-to-tree translation as parsing achieves comparable performance with Hiero (Chiang, 2007) using much fewer rules. Xiao et al. (2010) integrate tokenization and translation into a single step and improve the performance of tokenization and translation significantly.

5 Conclusion

We have presented a framework for joint parsing and translation by casting tree-to-string translation as a parsing problem. While tree-to-string rules construct parse trees on the source side and translations on the target side simultaneously, parsing models can be integrated to improve both translation and parsing quality.

This work can be considered as a final step towards the continuum of tree-to-string translation: from single tree to forest and finally to the integration of parsing and translation. In the future, we plan to develop more efficient decoding algorithms, analyze forest matching systematically, and use more sophisticated parsing models.

Acknowledgement

The authors were supported by National Natural Science Foundation of China, Contracts 60736014 and 60903138, and 863 State Key Project No. 2006AA010108. We are grateful to the anonymous reviewers for their insightful comments. We thank Liang Huang, Hao Zhang, and Tong Xiao for discussions on synchronous binarization, Haitao Mi and Hao Xiong for providing and running the baseline systems, and Wenbin Jiang for helping us train parsing models.

References

Blunsom, Phil and Miles Osborne. 2008. Probabilistic inference for machine translation. In *Proc. of EMNLP 2008*.

Burkett, David and Dan Klein. 2008. Two languages are better than one (for syntactic parsing). In *Proc. of EMNLP 2008*.

Charniak, Eugene, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for statistical machine translation. In *Proc. of MT Summit IX*.

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Chiang, David. 2010. Learning to translate with source and target syntax. In *Proc. of ACL 2010*.

Collins, Michael. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Eisner, Jason. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proc. of ACL 2003*.

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. of NAACL 2004*.

Galley, Michel, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. of ACL 2006*.

Huang, Liang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. of AMTA 2006*.

Huang, Liang, Wenbin Jiang, and Qun Liu. 2009a. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proc. of EMNLP 2009*.

Huang, Liang, Hao Zhang, Daniel Gildea, and Kevin Knight. 2009b. Binarization of synchronous context-free grammars. *Computational Linguistics*, 35(4):559–595.

Huang, Liang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proc. of ACL 2008*.

Liu, Yang, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. of ACL 2006*.

Liu, Yang, Yajuan Lü, and Qun Liu. 2009. Improving tree-to-tree translation with packed forests. In *Proc. of ACL 2009*.

Melamed, I. Dan. 2004. Statistical machine translation by parsing. In *Proc. of ACL 2004*.

Mi, Haitao, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proc. of ACL 2008*.

Och, Franz J. and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, Franz. 2003. Minimum error rate training in statistical machine translation. In *Proc. of ACL 2003*.

Petrov, Slav and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proc. of NAACL 2007*.

Quirk, Chris and Simon Corston-Oliver. 2006. The impact of parsing quality on syntactically-informed statistical machine translation. In *Proc. of EMNLP 2006*.

Quirk, Chris, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. of ACL 2005*.

Shen, Libin, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL 2008*.

Smith, David and Noah Smith. 2004. Bilingual parsing with factored estimation: using english to parse korean. In *Proc. of EMNLP 2004*.

Stolcke, Andreas. 2002. Srilm - an extension language model modeling toolkit. In *Proc. of ICSLP 2002*.

Xiao, Xinyan, Yang Liu, Young-Sook Hwang, Qun Liu, and Shouxun Lin. 2010. Joint tokenization and translation. In *Proc. of COLING 2010*.

Xiong, Deyi, Shuanglong Li, Qun Liu, and Shouxun Lin. 2005. Parsing the penn chinese treebank with semantic knowledge. In *Proc. of IJCNLP 2005*.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. of ACL 2001*.

Zhang, Hao, Liang Huang, Daniel Gildea, and Kevin Knight. 2006. Synchronous binarization for machine translatio. In *Proc. of NAACL 2007*.

Zhang, Min, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proc. of ACL 2008*.

Zhao, Bing and Yaser Al-Onaizan. 2008. Generalizing local and non-local word-reordering patterns for syntax-based machine translation. In *Proc. of EMNLP 2008*.

Semantic Role Features for Machine Translation

Ding Liu
Department of Computer Science
University of Rochester

Daniel Gildea
Department of Computer Science
University of Rochester

Abstract

We propose semantic role features for a Tree-to-String transducer to model the reordering/deletion of source-side semantic roles. These semantic features, as well as the Tree-to-String templates, are trained based on a conditional log-linear model and are shown to significantly outperform systems trained based on Max-Likelihood and EM. We also show significant improvement in sentence fluency by using the semantic role features in the log-linear model, based on manual evaluation.

1 Introduction

Syntax-based statistical machine translation (SSMT) has achieved significant progress during recent years (Galley et al., 2006; May and Knight, 2007; Liu et al., 2006; Huang et al., 2006), showing that deep linguistic knowledge, if used properly, can improve MT performance. Semantics-based SMT, as a natural extension to SSMT, has begun to receive more attention from researchers (Liu and Gildea, 2008; Wu and Fung, 2009). Semantic structures have two major advantages over syntactic structures in terms of helping machine translation. First of all, semantic roles tend to agree better between two languages than syntactic constituents (Fung et al., 2006). This property motivates the approach of using the consistency of semantic roles to select MT outputs (Wu and Fung, 2009). Secondly, the set of semantic roles of a predicate models the *skeleton* of a sentence, which is crucial to the readability of MT output. By skeleton, we mean the main structure of a sentence including the verbs and their arguments. In spite of the theoretical potential of the semantic roles, there has not been much success in using them to improve SMT systems.

Liu and Gildea (2008) proposed a semantic role based Tree-to-String (TTS) transducer by adding semantic roles to the TTS templates. Their approach did not differentiate the semantic roles of different predicates, and did not always improve the TTS transducer's performance. Wu and Fung (2009) took the output of a phrase-based SMT system Moses (Koehn et al., 2007), and kept permuting the semantic roles of the MT output until they best matched the semantic roles in the source sentence. This approach shows the positive effect of applying semantic role constraints, but it requires re-tagging semantic roles for every permuted MT output and does not scale well to longer sentences.

This paper explores ways of tightly integrating semantic role features (SRFs) into an MT system, rather than using them in post-processing or n-best re-ranking. Semantic role labeling (SRL) systems usually use sentence-wide features (Xue and Palmer, 2004; Pradhan et al., 2004; Toutanova et al., 2005); thus it is difficult to compute target-side semantic roles incrementally during decoding. Noticing that the source side semantic roles are easy to compute, we apply a compromise approach, where the target side semantic roles are generated by projecting the source side semantic roles using the word alignments between the source and target sentences. Since this approach does not perform true SRL on the target string, it cannot fully evaluate whether the source and target semantic structures are consistent. However, the approach does capture the semantic-level re-ordering of the sentences. We assume here that the MT system is capable of providing word alignment (or equivalent) information during decoding, which is generally true for current statistical MT systems.

Specifically, two types of semantic role features are proposed in this paper: a semantic role reordering feature designed to capture the skeleton-level permutation, and a semantic role deletion fea-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 716–724,
Beijing, August 2010

ture designed to penalize missing semantic roles in the target sentence. To use these features during decoding, we need to keep track of the semantic role sequences (SRS) for partial translations, which can be generated based on the source-side semantic role sequence and the corresponding word alignments. Since the SRL system and the MT system are separate, a translation rule (e.g., a phrase pair in phrase-based SMT) could cover two partial source-side semantic roles. In such cases partial SRSs must be recorded in such a way that they can be combined later with other partial SRSs. Dealing with this problem will increase the complexity of the decoding algorithm. Fortunately, Tree-to-String transducer based MT systems (Liu et al., 2006; Huang et al., 2006) can avoid this problem by using the same syntax tree for both SRL and MT. Such an arrangement guarantees that a TTS template either covers parts of one source-side semantic role, or a few complete semantic roles. This advantage motivates us to use a TTS transducer as the MT system with which to demonstrate the use of the proposed semantic role features. Since it is hard to design a generative model to combine both the semantic role features and the TTS templates, we use a log-linear model to estimate the feature weights, by maximizing the conditional probabilities of the target strings given the source syntax trees. The log-linear model with latent variables has been discussed by Blunsom et al. (2008); we apply this technique to combine the TTS templates and the semantic role features.

The remainder of the paper is organized as follows: Section 2 describes the semantic role features proposed for machine translation; Section 3 describes how semantic role features are used and trained in a TTS transducer; Section 4 presents the experimental results; and Section 5 gives the conclusion.

2 Semantic Role Features for Machine Translation

2.1 Defining Semantic Roles

There are two semantic standards with publicly available training data: PropBank (Palmer et al., 2005) and FrameNet (Johnson et al., 2002). PropBank defines a set of semantic roles for the verbs in the Penn TreeBank using numbered roles. These roles are defined individually for each verb. For example, for the verb *disappoint*, the role name *arg1* means *experiencer*, but for the verb *wonder*, role name *arg1* means *cause*. FrameNet is motivated by the idea that a certain type of verbs can be gathered together to form a *frame*, and in the same frame, a set of semantic roles is defined and shared among the verbs. For example, the verbs *boil*, *bake*, and *steam* will be in frame *apply_heat*, and they have the semantic roles of *cook*, *food*, and *heating_instrument*. Of these two semantic standards, we choose PropBank over FrameNet for the following reasons:

1. PropBank has a simpler semantic definition than FrameNet and thus is easier for automatic labeling.

2. PropBank is built upon the Penn TreeBank and is more consistent with statistical parsers, most of which are trained on the Penn TreeBank.

3. PropBank is a larger corpus than FrameNet.

Note that the semantic standard/corpus is not crucial in this paper. Any training corpus that can be used to automatically obtain the set of semantic roles of a verb could be used in our approach.

2.2 Semantic Role Features

Ideally, we want to use features based on the true semantic roles of the MT candidates. Considering there is no efficient way of integrating SRL and MT, accurate target-side semantic roles can only be used in post-processing and re-ranking the MT outputs, where a limited number of MT candidates are considered. On the other hand, it is much easier to obtain reliable semantic roles for the source sentences. This paper uses a compromise approach, where the target-side semantic roles are projected from the source-side semantic roles using the word alignment derived from the translation process. More specifically, we define two types of semantic role features:

1. **Semantic Role Re-ordering (SRR)** This feature describes re-ordering of the source-side

semantic roles (including the predicate) in the target side. It takes the following form:

$$SrcPred : SrcRole_1, ..., SrcRole_n \Rightarrow TarRole_1, ..., TarRole_n$$

where $SrcPred$ and $SrcRole$ denotes the central verb and semantic roles in the source side, and $TarRole$ denotes the target-side roles. The source/target SRSs do not need be continuous, but there should be a one-to-one alignment between the roles in the two sides. Compared to the general re-ordering models used in statistical MT systems, this type of feature is capable of modeling skeleton-level re-ordering, which is crucial to the fluency of MT output. Because a predicate can have different semantic role sequences in different voices, *passive/active* are tagged for each occurrence of the verbs based on their POS and preceding words. Figure 1 shows examples of the feature SRR.

2. **Deleted Roles (DR)** are the individual source-side semantic roles which are deleted in the MT outputs, taking the form of:

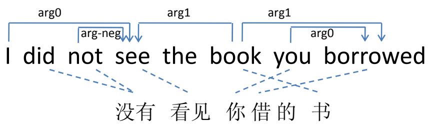
$$SrcPred : SrcRole \Rightarrow deleted$$

DR is meant to penalize the deletion of the semantic roles. Though most statistical MT systems have penalties for word deletion, it is still useful to make separate features for the deletion of semantic roles, which is considered more harmful than the deletion of non-core components (e.g., modifiers) and deserves more serious penalty. Examples of the deletion features can be found in Figure 1.

Both types of features can be made non-lexicalized by removing the actual verb but retaining its voice information in the features. Non-lexicalized features are used in the system to alleviate the problem of sparse verbs.

3 Using Semantic Role Features in Machine Translation

This section describes how to use the proposed semantic role features in a Tree-to-String transducer, assuming that the semantic roles have been tagged for the source sentences. We first briefly describe the basic Tree-to-String translation model used in our experiments, and then describe how to modify it to incorporate the semantic role features.



SRR:
see-active: arg-neg verb ➔ arg-neg verb
borrowed-active: arg1 arg0 ➔ arg0 arg1
borrowed-active: arg1 verb ➔ verb arg1
borrowed-active: arg0 verb ➔ arg0 verb
borrowed-active: arg1 arg0 verb ➔ arg0 verb arg1
DR:
see-active: arg0 ➔ deleted

Figure 1: Examples of the semantic role features

3.1 Basic Tree-to-String Transducer

A Tree-to-String transducer receives a syntax tree as its input and, by recursively applying TTS templates, generates the target string. A TTS template is composed of a left-hand side (LHS) and a right-hand side (RHS), where the LHS is a subtree pattern and the RHS is a sequence of variables and translated words. The variables in the RHS of a template correspond to the bottom level non-terminals in the LHS's subtree pattern, and their relative order indicates the permutation desired at the point where the template is applied to translate one language to another. The variables are further transformed, and the recursive process goes on until there are no variables left. The formal description of a TTS transducer is given by Graehl and Knight (2004), and our baseline approach follows the *Extended Tree-to-String Transducer* defined by Huang et al. (2006). For a given derivation (decomposition into templates) of a syntax tree, the translation probability is computed as the product of the templates which generate both the source syntax trees and the target translations.

$$Pr(S \mid T, D^*) = \prod_{t \in D^*} Pr(t)$$

Here, S denotes the target sentence, T denotes the source syntax tree, and D^* denotes the derivation of T. In addition to the translation model, the

```
function DECODE($T$)
    for tree node $v$ of $T$ in bottom-up order do
        for template $t$ applicable at $v$ do

            $\{c_1, c_2\}$=match($v, t$);
            $s.leftw = c_1.leftw$;
            $s.rightw = c_2.rightw$;
            $s.val = c_1.val \times c_2.val$;
            $s.val \times= Pr(t)$;
            $s.val \times= Pr(c_2.leftw|c_1.rightw)$;
            add $s$ to $v$'s beam;
```

Figure 2: Decoding algorithm for the standard Tree-to-String transducer. $leftw$/$rightw$ denote the left/right boundary word of s. c_1, c_2 denote the descendants of v, ordered based on RHS of t.

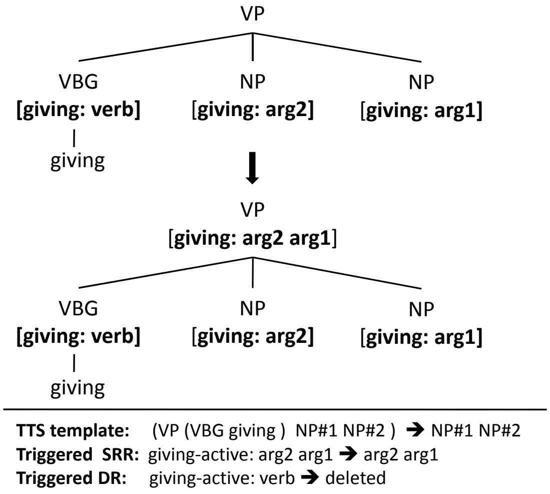
TTS system includes a trigram language model, a deletion penalty, and an insertion bonus. The bottom-up decoding algorithm for the TTS transducer is sketched in Figure 2. To incorporate the n-gram language model, states in the algorithm denote a tree node's best translations with different left and right boundary words. We use standard beam-pruning to narrow the search space. To simplify the description, we assume in Figure 2 that a bigram language model is used and all the TTS templates are binarized. It is straightforward to generalize the algorithm for larger n-gram models and TTS templates with any number of children in the bottom using target-side binarized combination (Huang et al., 2006).

3.2 Modified Tree-to-String Transducer with Semantic Role Features

Semantic role features can be used as an auxiliary translation model in the TTS transducer, which focuses more on the skeleton-level permutation. The model score, depending on not only the input source tree and the derivation of the tree, but also the semantic roles of the source tree, can be formulated as:

$$Pr(S \mid T, D^*) = \prod_{f \in F(S, T.role, D^*)} Pr(f)$$

where T denotes the source syntax tree with semantic roles, $T.role$ denotes the semantic role sequence in the source side and $F(S.role, T.role, D^*)$ denotes the set of defined semantic role features over $T.role$ and the target side semantic role sequence $S.role$. Note that given $T.role$ and the derivation D^*, $S.role$ can be easily derived. Now we show how to incorporate the two types of semantic role features into a TTS transducer. To use the semantic role re-ordering feature SRR, the states in the decoding algorithm need to be expanded to encode the target-side SRSs. The SRSs are initially attached to the translation states of the source tree con-



TTS template: (VP (VBG giving) NP#1 NP#2) → NP#1 NP#2
Triggered SRR: giving-active: arg2 arg1 → arg2 arg1
Triggered DR: giving-active: verb → deleted

Figure 3: An example showing the combination of the semantic role sequences of the states. Above/middle is the state information before/after applying the TTS template, and bottom is the used TTS template and the triggered SRFs during the combination.

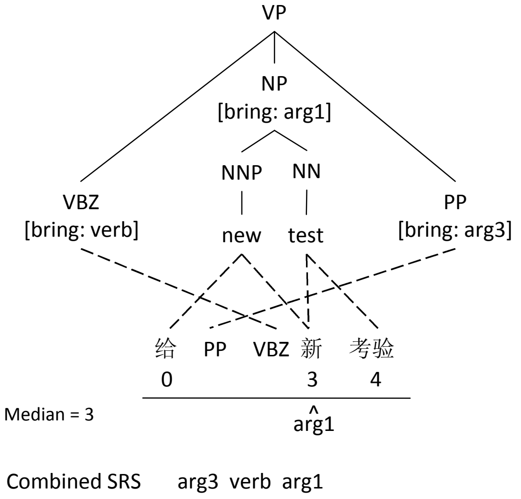


Figure 4: An example showing how to compute the target side position of a semantic role by using the median of its aligning points.

stituents which are labeled as semantic roles for some predicate. These semantic roles are then accumulated with re-ordering and deletion operations specified by the TTS templates as the decoding process goes bottom-up. Figure 5 shows the decoding algorithm incorporating the SRR features. The model component corresponding to the feature SRR is computed when combining two translation states. I.e., the probabilities of the SRR features composed based on the semantic roles of the two combining states will be added into the combined state. See Figure 3 for examples. The theoretical upper bound of the decoding complexity is $O(NM^{4(n-1)}R(\sum_{i=0}^{C} \frac{C!}{i!})^V)$, where N is the number of nodes in the source syntax tree, M is the vocabulary size of the target language, n is the order of the n-gram language model, R is the maximum number of TTS templates which can be matched at a tree node, C is the maximum number of roles of a verb, and V is the maximum number of verbs in a sentence. In this formula, $\sum_{i=0}^{C} \frac{C!}{i!}$ is the number of role sequences obtained by first choosing i out of C possible roles and then permuting the i roles. This theoretical upper bound is not reached in practice, because the number of possible TTS templates applicable at a tree node is very limited. Furthermore, since we apply beam pruning at each tree node, the running time is controlled by the beam size, and is linear in the size of the tree.

The re-ordering of the semantic roles from source to target is computed for each TTS template as part of the template extraction process, using the word-level alignments between the LHS/RHS of the TTS template (e.g., Figure 3). This is usually straightforward, with the exception of the case where the words that are aligned to a particular role's span in the source side are not continuous in the target side, as shown in Figure 4. Since we are primarily interested in the relative order of the semantic roles, we approximate each semantic role's target side position by the median of the word positions that is aligned to. If more than one semantic role is mapped to the same position in the target side, their source side order will be used as their target side order, i.e., monotonic translation is assumed for those semantic roles. Figure 4 shows an example of calculating the target side SRS based on a complicated TTS template. The word alignments in the TTS templates are also used to compute the deletion feature DR. Whenever a semantic role is deleted in a TTS template's RHS, the corresponding deletion penalty will be applied.

```
function DECODE($T$)
    for tree node $v$ of $T$ in bottom-up order do
        for template $t$ applicable at $v$ do
            $\{c_1, c_2\}$=match($v, t$);
            $s.leftw = c_1.leftw$;
            $s.rightw = c_2.rightw$;
            $s.role$ = concatenate($c_1.role$, $c_2.role$);
            if $v$ is a semantic role then
                set $s.role$ to $v.role$;
            $s.val = c_1.val \times c_2.val$;
            $s.val \times= Pr(t)$;
            $s.val \times= Pr(c_2.leftw|c_1.rightw)$;
▷ Compute the probabilities associated with semantic roles
            $s.val \times= \prod_{f \in Sema(c_1.role, c_2.role, t)} Pr(f)$;
            add $s$ to $v$'s beam;
```

Figure 5: Decoding algorithm using semantic role features. $Sema(c_1.role, c_2.role, t)$ denotes the triggered semantic role features when combining two children states, and examples can be found in Figure 3.

3.3 Training

We describe two alternative methods for training the weights for the model's features, including both the individual TTS templates and the semantic role features. The first method maximizes data likelihood as is standard in EM, while the second method maximizes conditional likelihood for a log-linear model following Blunsom et al. (2008).

3.3.1 Maximizing Data Likelihood

The standard way to train a TTS translation model is to extract the minimum TTS templates using GHKM (Galley et al., 2004), and then normalize the frequency of the extracted TTS templates (Galley et al., 2004; Galley et al., 2006; Liu et al., 2006; Huang et al., 2006). The probability of the semantic features SRR and DR can be computed similarly, given that SRR and DR can be derived from the paired source/target sentences and the word alignments between them. We refer to this model as max-likelihood training and normalize the counts of TTS templates and semantic features based on their roots and predicates respectively.

We wish to overcome noisy alignments from GIZA++ and learn better TTS rule probabilities by re-aligning the data using EM within the TTS

```
E-step:
for all pair of syntax tree $T$ and target string $S$ do
    for all TTS Template $t$, semantic features $f$ do
        $EC(t)$ += $\frac{\sum_{D:t\in D} Pr(S,T,D)}{\sum_{D'} Pr(S,T,D')}$;
        $EC(f)$ += $\frac{\sum_{D:f\in D} Pr(S,T,D)}{\sum_{D'} Pr(S,T,D')}$;
M-step:
for all TTS Template $t$, semantic features $f$ do
    $Pr(t) = \frac{EC(t)}{\sum_{t':t'.root=t.root} EC(t')}$;
    $Pr(f) = \frac{EC(f)}{\sum_{f':f'.predicate=t.predicate} EC(f')}$;
```

Figure 6: EM Algorithm For Estimating TTS Templates and Semantic Features

```
function COMPUTE_PARTITION($T$)
    for tree node $v$ of $T$ in bottom-up order do
        for template $t$ applicable at $v$ do
            for $\{s_1, s_2\}$=Match($v$, $t$) do
                $s.sum$ += $s_1.sum \times s_2.sum \times$
$exp(\lambda_t + \sum_{f\in Sema(s_1,s_2,t)} \lambda_f)$;
                $s.role$ = concatenate($s_1.role$, $s_2.role$);
                add $s$ to $v$;
    for state $s$ in $root$ do $res$ += $s.sum$;
    return $res$;
```

Figure 7: Computing the partition function of the conditional probability $Pr(S|T)$. $Sema(s_1, s_2, t)$ denotes all the semantic role features generated by combining s_1 and s_2 using t.

framework (May and Knight, 2007). We can estimate the expected counts of the TTS templates and the semantic features by formulating the probability of a pair of source tree and target string as:

$$\sum_D Pr(S,T,D) = \sum_D \left(\prod_{t\in D} Pr(t) \prod_{f\in F(S,T.role,D)} Pr(f) \right)$$

Though the above formulation, which makes the total probability of all the pairs of trees and strings less than 1, is not a strict generative model, we can still use the EM algorithm (Dempster et al., 1977) to estimate the probability of the TTS templates and the semantic features, as shown in Figure 6.

The difficult part of the EM algorithm is the *E-step*, which computes the expected counts of the TTS templates and the semantic features by summing over all possible derivations of the source trees and target strings. The standard inside-outside algorithm (Graehl and Knight, 2004) can be used to compute the expected counts of the TTS templates. Similar to the modification made in the TTS decoder, we can add the target-side semantic role sequence to the dynamic programming states of the inside-outside algorithm to compute the expected counts of the semantic features. This way each state (associated with a source tree node) represents a target side span and the partial SRSs. To speed up the training, a beam is created for each target span and only the top rated SRSs in the beam are kept.

3.3.2 Maximizing Conditional Likelihood

A log-linear model is another way to combine the TTS templates and the semantic features together. Considering that the way the semantic role features are defined makes it impossible to design a sound generative model to incorporate these features, a log-linear model is also a theoretically better choice than the EM algorithm. If we directly translate the EM algorithm into the log-linear model, the problem becomes maximizing the data likelihood represented by feature weights instead of feature probabilities:

$$Pr(S,T) = \frac{\sum_D \exp \sum_i \lambda_i f_i(S,T,D)}{\sum_{S',T'} \sum_{D'} \exp \sum_i \lambda_i f_i(S',T',D')}$$

where the features f include both the TTS templates and the semantic role features. The numerator in the formula above can be computed using the same dynamic programming algorithm used to compute the expected counts in the EM algorithm. However, the partition function (denominator) requires summing over all possible source trees and target strings, and is infeasible to compute. Instead of approximating the partition function using methods such as sampling, we change the objective function from the data likelihood to the conditional likelihood:

$$Pr(S \mid T) = \frac{\sum_D \exp \sum_i \lambda_i f_i(S,T,D)}{\sum_{S'\in all(T)} \sum_{D'} \exp \sum_i \lambda_i f_i(S',T,D')}$$

where $all(T)$ denotes all the possible target strings which can be generated from the source tree T. Given a set of TTS templates, the new partition function can be efficiently computed using the dynamic programming algorithm shown in Figure 7. Again, to simplify the illustration, only binary TTS templates are used. Using the conditional probability as the objective function not only reduces the computational cost, but also corresponds better to the TTS decoder, where the best MT output is selected only among the possible candidates which can be generated from the input source tree using TTS templates.

The derivative of the logarithm of the objective function (over the entire training corpus) w.r.t. a feature weight can be computed as:

$$\frac{\partial \log \prod_{S,T} Pr(S \mid T)}{\partial \lambda_i} = \sum_{S,T} \{EC_{D|S,T}(f_i) - EC_{S'|T}(f_i)\}$$

where $EC_{D|S,T}(f_i)$, the expected count of a feature over all derivations given a pair of tree and string, can be computed using the modified inside-outside algorithm described in Section 3.2, and $EC_{S'|T}(f_i)$, the expected count of a feature over all possible target strings given the source tree, can be computed in a similar way to the partition function described in Figure 7. With the objective function and its derivatives, a variety of optimization methods can be used to obtain the best feature weights; we use LBFGS (Zhu et al., 1994) in our experiments. To prevent the model from overfitting the training data, a weighted Gaussian prior is used with the objective function. The variance of the Gaussian prior is tuned based on the development set.

4 Experiments

We train an English-to-Chinese translation system using the FBIS corpus, where 73,597 sentence pairs are selected as the training data, and 500 sentence pairs with no more than 25 words on the Chinese side are selected for both the development and test data.[1] Charniak (2000)'s parser, trained on the Penn Treebank, is used to generate the English syntax trees. To compute the semantic roles for the source trees, we use an in-house max-ent classifier with features following Xue and Palmer (2004) and Pradhan et al. (2004). The semantic role labeler is trained and tuned based on sections 2–21 and section 24 of PropBank respectively. The standard role-based F-score of our semantic role labeler is 88.70%. Modified Kneser-Ney trigram models are trained using SRILM (Stolcke, 2002) on the Chinese portion of the training data. The model (n-gram language model, TTS templates, SRR, DR) weights of the transducer are tuned based on the development set using a grid-based line search, and the translation results are evaluated based on a single Chinese reference using BLEU-4 (Papineni et al., 2002). Huang et al. (2006) used character-based BLEU as a way of normalizing inconsistent Chinese word segmentation, but we avoid this problem as the training, development, and test data are from the same source.

The baseline system in our experiments uses the TTS templates generated by using GHKM and the union of the two single-direction alignments generated by GIZA++. Unioning the two single-direction alignments yields better performance for the SSMT systems using TTS templates (Fossum et al., 2008) than the two single-direction alignments and the heuristic diagonal combination (Koehn et al., 2003). The two single-direction word alignments as well as the union are used to generate the initial TTS template set for both the EM algorithm and the log-linear model. The initial TTS templates' probabilities/weights are set to their normalized counts based on the root of the TTS template (Galley et al., 2006). To test semantic role features, their initial weights are set to their normalized counts for the EM algorithm and to 0 for the log-linear model. The performance of these systems is shown in Table 1. We can see that the EM algorithm, based only on TTS templates, is slightly better than the baseline system. Adding semantic role features to the EM algorithm actually hurts the performance, which is not surprising since the combination of the TTS templates and semantic role features does not yield a sound generative model. The log-linear model based on TTS templates achieves significantly better results than both the baseline system and the EM algorithm. Both improvements are significant at $p < 0.05$ based on 2000 iterations of paired bootstrap resampling of the test set (Koehn, 2004).

Adding semantic role features to the log-linear model further improves the BLEU score. One problem in our approach is the sparseness of the verbs, which makes it difficult for the log-linear model to tune the lexicalized semantic role features. One way to alleviate this problem is to make features based on verb classes. We first tried using the verb

[1] The total 74,597 sentence pairs used in experiments are those in the FBIS corpus whose English part can be parsed using Charniak (2000)'s parser.

	TTS Templates	+ SRF	+ Verb Class
Union	15.6	–	–
EM	15.9	15.5	15.6
Log-linear	17.1	17.4	17.6

Table 1: BLEU-4 scores of different systems

	equal	better	worse
With SRF vs. W/O SRF	72%	20.2%	7.8%

Table 2: Distribution of the sentences where the semantic role features give no/positive/negative impact to the sentence fluency in terms of the completeness and ordering of the semantic roles.

Source	Launching$_1$ New$_2$ Diplomatic$_3$ Offensive$_4$
SRF On	实施$_1$ 新的$_2$ 外交$_3$ 攻势$_4$
SRF Off	新的$_2$ 外交$_3$ 攻势$_4$

Source	It$_1$ is$_2$ therefore$_3$ necessary$_4$ to$_5$ speed$_6$ up$_7$ the$_8$ transformation$_9$ of$_{10}$ traditional$_{11}$ industries$_{12}$ with$_{13}$ high$_{14}$ technologies$_{15}$
SRF On	所以$_{123}$ 要$_4$ 加快$_{6,7}$ 高新技术$_{14,15}$ 改造$_9$ 传统产业$_{11,12}$
SRF Off	所以$_{123}$ 要$_4$ 高技术$_{14,15}$ ，加快$_{6,7}$ 传统产业$_{11,12}$ 改造$_9$

Source	A$_1$ gratifying$_2$ change$_3$ also$_4$ occurred$_5$ in$_6$ the$_7$ structure$_8$ of$_9$ ethnic$_{10}$ minority$_{11}$ cadres$_{12}$
SRF On	少数民族$_{10,11}$ 结构$_8$ 也$_4$ 发生$_5$ 可喜的$_2$ 变化$_3$
SRF Off	一个$_1$ 可喜的$_2$ 变化$_3$，还在$_4$ 少数民族$_{10,11}$ 干部的 结构$_8$

Figure 8: Examples of the MT outputs with and without SRFs. The first and second example shows that SRFs improve the completeness and the ordering of the MT outputs respectively, the third example shows that SRFs improve both properties. The subscripts of each Chinese phrase show their aligned words in English.

classes in VerbNet (Dang et al., 1998). Unfortunately, VerbNet only covers about 34% of the verb tokens in our training corpus, and does not improve the system's performance. We then resorted to automatic clustering based on the aspect model (Hofmann, 1999; Rooth et al., 1999). The training corpus used in clustering is the English portion of the selected FBIS corpus. Though automatically obtained verb clusters lead to further improvement in BLEU score, the total improvement from the semantic role features is not statistically significant. Because BLEU-4 is biased towards the adequacy of the MT outputs and may not effectively evaluate their fluency, it is desirable to give a more accurate evaluation of the sentence's fluency, which is the property that semantic role features are supposed to improve. To do this, we manually compare the outputs of the two log-linear models with and without the semantic role features. Our evaluation focuses on the completeness and ordering of the semantic roles, and *better*, *equal*, *worse* are tagged for each pair of MT outputs indicating the impact of the semantic role features. Table 2 shows the manual evaluation results based on the entire test set, and the improvement from SRF is significant at $p < 0.005$ based on a t-test. To illustrate how SRF impacts the translation results, Figure 8 gives 3 examples of the MT outputs with and without the SRFs.

5 Conclusion

This paper proposes two types of semantic role features for a Tree-to-String transducer: one models the reordering of the source-side semantic role sequence, and the other penalizes the deletion of a source-side semantic role. These semantic features and the Tree-to-String templates, trained based on a conditional log-linear model, are shown to significantly improve a basic TTS transducer's performance in terms of BLEU-4. To avoid BLEU's bias towards the adequacy of the MT outputs, manual evaluation is conducted for sentence fluency and significant improvement is shown by using the semantic role features in the log-linear model. Considering our semantic features are the most basic ones, using more sophisticated features (e.g., the head words and their translations of the source-side semantic roles) provides a possible direction for further experimentation.

Acknowledgments This work was funded by NSF IIS-0546554 and IIS-0910611.

References

Blunsom, Phil, Trevor Cohn, and Miles Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, Columbus, Ohio.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL-01*, pages 132–139.

Dang, Hoa Trang, Karin Kipper, Martha Palmer, and

Joseph Rosenzweig. 1998. Investigating regular sense extensions based on intersective Levin classes. In *COLING/ACL-98*, pages 293–299, Montreal. ACL.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the *EM* algorithm. *Journal of the Royal Statistical Society*, 39(1):1–21.

Fossum, Victoria, Kevin Knight, and Steven Abney. 2008. Using syntax to improveword alignment precision for syntax-based machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, Columbus, Ohio. ACL.

Fung, Pascale, Zhaojun Wu, Yongsheng Yang, and Dekai Wu. 2006. Learning of Chinese/English semantic structure mapping. In *IEEE/ACL 2006 Workshop on Spoken Language Technology*, Aruba.

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of NAACL-04*, pages 273–280.

Galley, Michel, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of COLING/ACL-06*, pages 961–968, July.

Graehl, Jonathan and Kevin Knight. 2004. Training tree transducers. In *Proceedings of NAACL-04*.

Hofmann, Thomas. 1999. Probabilistic latent semantic analysis. In *Uncertainity in Artificial Intelligence, UAI'99*, Stockholm.

Huang, Liang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas (AMTA)*, Boston, MA.

Johnson, Christopher R., Charles J. Fillmore, Miriam R. L. Petruck, Collin F. Baker, Michael Ellsworth, Josef Ruppenhofer, and Esther J. Wood. 2002. FrameNet: Theory and practice. Version 1.0, http://www.icsi.berkeley.edu/framenet/.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL-03*, Edmonton, Alberta.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL, Demonstration Session*, pages 177–180.

Koehn, Philipp. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395, Barcelona, Spain, July.

Liu, Ding and Daniel Gildea. 2008. Improved tree-to-string transducers for machine translation. In *ACL Workshop on Statistical Machine Translation (ACL08-SMT)*, pages 62–69, Columbus, Ohio.

Liu, Yang, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING/ACL-06*, Sydney, Australia, July.

May, Jonathan and Kevin Knight. 2007. Syntactic re-alignment models for machine translation. In *Proceedings of EMNLP*.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL-02*.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James Martin, , and Dan Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of NAACL-04*.

Rooth, Mats, Stefan Riezler, Detlef Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via EM-based clustering. In *Proceedings of the 37th Annual Meeting of the ACL*, pages 104–111, College Park, Maryland.

Stolcke, Andreas. 2002. SRILM - an extensible language modeling toolkit. In *International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Toutanova, Kristina, Aria Haghighi, and Christopher Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of ACL-05*, pages 589–596.

Wu, Dekai and Pascale Fung. 2009. Semantic roles for smt: A hybrid two-pass model. In *Proceedings of the HLT-NAACL 2009: Short Papers*, Boulder, Colorado.

Xue, Nianwen and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of EMNLP*.

Zhu, Ciyou, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. 1994. L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. Technical report, ACM Trans. Math. Software.

TimeML Events Recognition and Classification: Learning CRF Models with Semantic Roles

Hector Llorens, Estela Saquete, Borja Navarro-Colorado
Natural Language Processing Group
University of Alicante
{hllorens,stela,borja}@dlsi.ua.es

Abstract

This paper analyzes the contribution of semantic roles to TimeML event recognition and classification. For that purpose, an approach using conditional random fields with a variety of morphosyntactic features plus semantic roles features is developed and evaluated. Our system achieves an F1 of 81.4% in recognition and a 64.2% in classification. We demonstrate that the application of semantic roles improves the performance of the presented system, especially for nominal events.

1 Introduction

Event recognition and classification has been pointed out to be very important to improve complex natural language processing (NLP) applications such as automatic summarization (Daniel et al., 2003) and question answering (QA) (Pustejovsky, 2002). Natural language (NL) texts often describe sequences of events in a time line. In the context of summarization, extracting such events may aid in obtaining better summaries when these have to be focused on specific happenings. In the same manner, the access to such information is crucial for QA systems attempting to address questions about events.

The analysis of events as well as the classification of the different forms they adopt in NL text is not a new issue (Vendler, 1967). It relates not only to linguistics but different scientific areas such as philosophy, psychology, etc.

In NLP, different definitions of event can be found regarding the target application.

On the one hand, in topic detection and tracking (Allan, 2002), event is defined as an instance of a topic identified at document level describing something that happen (e.g., "wars"). The aim of this task is to cluster documents on the same topic, that is to say, the same event.

On the other hand, information extraction (IE) provides finer granularity event definitions. IE proposes standard schemes to annotate the individual events within the scope of a document. STAG scheme (2000) was aimed to identify events in news and their relationship with points in a temporal line. More recently, TimeML (Pustejovsky et al., 2003a) presented a rich specification for annotating events in NL text extending the features of the previous one.

This paper is focused on the TimeML view of events. TimeML defines events as situations that *happen or occur*, or elements describing *states or circumstances* in which something obtains or holds the truth. These events are generally expressed by tensed or untensed verbs, nominalizations, adjectives, predicative clauses or prepositional phrases. TimeML guidelines define seven classes of events:

- *Reporting*. Action of a person or organization declaring or narrating an event (e.g., "say")
- *Perception*. Physical perception of another event (e.g., "see", "hear")
- *Aspectual*. Aspectual predication of another event (e.g., "start", "continue")
- *I_Action*. Intensional action (e.g., "try")
- *I_State*. Intensional state (e.g., "feel", "hope")
- *State*. Circumstance in which something holds the truth (e.g., "war", "in danger")
- *Occurrence*. Events that describe things that happen (e.g., "erupt", "arrive")

The following sentence shows an example of an occurrence event and a state event.

```
It's <EVENT class="OCCURRENCE">turning</EVENT>
out to be another <EVENT class="STATE">bad</EVENT>
financial week.
```

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 725–733,
Beijing, August 2010

The automatic annotation of events has been addressed with different data-driven approaches. Current approaches are mainly based on morphosyntactic information. Our hypothesis is that semantic roles, as higher language level analysis information, may be useful as additional feature to improve the performance of such approaches.

Within this setting, the main objective of this paper is to analyze (1) the contribution of semantic roles, as additional feature, and (2) the influence of conditional random fields (CRFs), as machine learning (ML) technique, in the events automatic recognition and classification task.

This paper is structured as follows. Firstly, related work in the task is reviewed in Section 2. The next section provides a detailed description of our proposal to address event recognition and classification. After that, Section 4 includes an evaluation of the proposal, and a comparative analysis of the results. Finally, conclusions are drawn in Section 5.

2 Related Work

There is only one corpus available annotated with TimeML events: TimeBank (Pustejovsky et al., 2003b). Hence, all the approaches regarding TimeML events extraction have been evaluated using this corpus.

EVITA system (Saurí et al., 2005) recognizes events by combining linguistic and statistical techniques. The main features used to manually encode event recognition rules are the following: part-of-speech (PoS) tagging, lemmatizing, chunking, lexical lookup and contextual parsing. Furthermore, WordNet information combined with Bayesian learned disambiguation was used to identify nominal events. EVITA obtained 74.03% precision, 87.31% recall, and 80.12% $F_{\beta=1}$ in event recognition over TimeBank.

Boguraev and Ando (2005) present an evaluation on automatic TimeML events annotation. They set out the task as a classification problem and used a robust risk minimization (RRM) classifier to solve it. The $F_{\beta=1}$ results obtained by a 5-fold cross validation over TimeBank were 78.6% for recognition and 61.3% for classification. Moreover, they evaluated the impact of applying word-profiling techniques over their approach to exploit unannotated data. Using this additional information, the $F_{\beta=1}$ results improved to 80.3% and 64.0%. In this evaluation, neither precision nor recall were given.

STEP (Bethard and Martin, 2006) is a system for TimeML event recognition and classification. This approach uses a rich set of textual, morphological, dependency and WordNet hypernymy features to build a Support Vector Machine (SVM) model. The model was trained using 9/10 of the TimeBank. The test, carried out using the remaining 1/10 of the corpus, obtained a 82.0% precision, 70.6% recall and 75.9% $F_{\beta=1}$ for recognition and a 66.7% precision, 51.2% recall and 57.9% $F_{\beta=1}$ for classification.

Finally, March and Baldwin (2008) present an evaluation on event recognition using a multiclass classifier (BSVM). The main features used to train the classifier are word and PoS context window, stop words removal and feature generalization through words grouping (numbers, named entities, etc.). The result for the best feature combination in a 10-fold cross validation over TimeBank was 76.4% $F_{\beta=1}$.

It is worth mentioning that there are two versions of the TimeBank corpus, 1.1 and 1.2. The latest version is the current gold standard. Both versions consist of the same documents[1], mainly news articles and transcribed broadcast news from different domains. EVITA is the only reference which used TimeBank 1.2 while the rest of reviewed references used TimeBank 1.1.

3 Our proposal: semantic roles enhancing a CRF model

In this section, the motivation for our proposal, and our specific approach are presented.

3.1 Motivation

The next two subsections describe the main feature (semantic roles) and the ML algorithm (CRFs) we selected to address event recognition and classification; and the reasons why we think they could be useful in that task.

[1] Except 3 documents removed in TimeBank 1.2

3.1.1 Semantic roles

Semantic role labeling (SRL) has achieved important results in the last years (Gildea and Jurafsky, 2002). For each predicate in a sentence, semantic roles identify all constituents, determining their arguments (agent, patient, etc.) and their adjuncts (locative, temporal, etc.). Currently, there exist different role sets aimed to cover opposed requirements. They range from more specific, such as FrameNet (Baker et al., 1998), to more general like PropBank (Palmer et al., 2005). Figure 1 illustrates a semantic role labeled sentence.

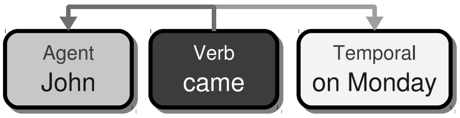


Figure 1: Semantic roles example

Many research efforts into the application of semantic roles demonstrated that this information is useful for different NLP purposes (Melli et al., 2006). Focusing on TimeML, semantic roles have been applied to temporal expressions recognition (Llorens et al., 2009), and temporal links classification (Hagège and Tannier, 2007). However, they have not been used to recognize and classify TimeML events.

Semantic roles provide structural relations of the predicates in which events may participate. Beyond syntactic relations expressed by means of the different types of phrases, semantic roles give further information about semantic relations between the arguments of a predicate. Therefore, as richer information, roles may better distinguish tokens to be candidate events. In addition, different semantic role settings may represent specific event classes.

Example 1 shows four sentences annotated with PropBank semantic roles (in square brackets) in which the noun "control" participates. In the sentences 1 and 2, "control" does not represent an event, while in the sentences 3 and 4, it represents an *state* event. It can be seen that the noun "control", when it is contained by A1 role it may represent an event. However, it is not an event when contained by A0 or AM-MNR roles. The analysis may also take into account the governing verb. In the example, we could specify that "control" represents an event when contained by A1 role of "seek" and "obtain" verbs; and the opposite for the A0 role of "emerge" and the AM-MNR of "had".

(1) 1. "[Control procedures A0] will emerge"
2. "[Iraq A0] had [thousands of Americans A1] [under its control AM-MNR]"
3. "[Crane Co. A0] may obtain [control of Milton Roy Corp. A1]"
4. "[Pattison's A0] decided to seek [control A1]"

Our hypothesis is that semantic roles, as additional information, may help in the recognition and classification of events. The information about the role of a token and the verb it depends on, or the set of roles of the sentence, could be useful for determining whether a token or a sequence of tokens is an event or not. Due to the fact that roles represent high level information in NL text, they are more independent from word tokens. Hence, roles may aid in learning more general models that could improve the results of approaches focused on lower level information.

3.1.2 CRF probabilistic model

Conditional Random Fields is a popular and efficient ML technique for supervised sequence labeling (Lafferty et al., 2001). CRFs are undirected graphical models, a special case of conditionally-trained finite state machines. A key advantage of CRFs is their flexibility to include a wide variety of arbitrary, non-independent features of the input.

We see the task set out in this paper as a sequence labeling problem. Assume X is a random variable over data sequences to be labeled, and Y is a random variable over the corresponding label sequences (hidden), being all Y components (Y_i) members of a finite label alphabet γ. X might range over NL sentences and Y range over event annotations of those sentences, with γ the set of possible event IOB2[2] labels. The following example illustrates the event recognition problem.

(2)

X	Y		
was	?		
another	?		B-EVENT
bad	?	? =	I-EVENT
week	?		O

[2] IOB2 format: (B)egin, (I)nside, and (O)utside

The variables X and Y are jointly distributed over both label and observation sequences. However, unlike Hidden Markov Models (generative) in which $p(X,Y)$, CRFs (discriminative) construct a conditional model from paired observation and label sequences: $p(Y|X)$. Graphically, CRFs are represented by undirected graphs, $G = (V,E)$ such that $Y = (Y_v),\ v \epsilon V$, so that Y is indexed by the vertices of G. Then (X,Y) is a *conditional random field* if Y_v variables obey the *Markov property* with respect to the graph when conditioned on X:

$$P(Y_v|X,Y_w,v \neq w) = P(Y_v|X,Y_w,v \sim w),$$

where $v \sim w$ means that Y_v and Y_w are connected neighbors in G.

To extend the problem to event classification, the alphabet γ must be extended with the event classes (state, aspectual, etc.).

CRFs have been successfully applied to many sequence labeling tasks (Sha and Pereira, 2003; McCallum and Li, 2003).

From our point of view, the task addressed in this paper is well suited for this ML technique. Events may depend on structural properties of NL sentences. Not only the word sequence, but morphological, syntactic and semantic information is related with the event structure (Tenny and Pustejovsky, 2000).

For example, sequences of verbs may represent *i_action+occurrence* or *aspectual+occurrence* events (see Example 3).

(3) "The president will <EVENT class="i_action"> try </EVENT> to <EVENT class="occurrence"> assist </EVENT> to the <EVENT class="occurrence"> conference </EVENT>"

"Saddam will <EVENT class="aspectual"> begin </EVENT> <EVENT class="occurrence"> withdrawing </EVENT> troops from Iranian territory on Friday"

In addition, for instance, many *state* event instances are represented by "to be" plus a variable quality (see Example 4).

(4) "It is <EVENT class="occurrence"> turning </EVENT> out to be another <EVENT class="state"> bad </EVENT> financial week."

Given this analysis, our hypothesis is that CRFs will be useful in the recognition of events in which the sequential and structural properties are relevant.

3.2 Approach description

This paper proposes CRFs as learning method to infer an event recognition and classification model. Our system includes CRF++ toolkit[3] for training and testing our approach. The learning process was done using *CRF-L2* algorithm and hyper-parameter C=1.

The definition of the features is crucial for the architecture of the system. The features used in our approach are grouped in two feature sets. On the one hand, general features, which comprise morphosyntactic and ontological information. On the other hand, semantic roles features, which are the main focus of this paper.

The general features used to train our CRF model are described regarding each language analysis level.

- **Morphological**: The lemma and PoS context, in a 5-window (-2,+2), was employed. This basic linguistic feature showed good results in different NLP tasks, as well as in event recognition and classification (March and Baldwin, 2008). Tokenization, PoS and lemmatization were obtained using TreeTagger (Schmid, 1994).

- **Syntactic**: Different events are contained in particular types of phrases and syntactic dependencies. This feature tries to tackle this by considering syntactic information. Charniak parser (Charniak and Johnson, 2005) was used to obtain the syntactic tree.

- **Lexical semantics**: WordNet (Fellbaum, 1998) top ontology classes have been widely used to represent word meaning at ontological level, and demonstrated its worth in many tasks. We obtained the four top classes for each word.

The specific semantic roles features used to enhance the training framework of the CRF model were developed considering PropBank role set. PropBank was applied in our system due to the high coverage it offers in contrast to FrameNet. In order to get PropBank semantic roles, the CCG

[3] http://crfpp.sourceforge.net/

SRL tool (Punyakanok et al., 2004) was used for labeling the corpus.

- **Role**: For each token, we considered the role regarding the verb the token depends on. Semantic roles information may be useful for distinguish particular lemmas that are events only when appearing under a precise role.
- **Governing verb**: The verb to which the current token holds a particular role. This may distinguish tokens appearing under the influence of different verbs.
- **Role+verb combination**: The previous two features were combined to capture the relation between them. This introduces new classification information by distinguishing roles depending on different verbs. The importance of this falls especially on the numbered roles of PropBank (A0, A1, ...) holding different meanings when depending on different verbs.
- **Role configuration**: This consists of the set of roles depending on the verb the token depends on. This may be particularly useful for distinguish different sentence settings and thus, whether a token denotes an event in a particular sentence type.

The system consists of two main processes. Firstly, given TimeML annotated text, it obtains the defined features plus the IOB2 tags of the annotated events. Then, using this data the system learns (trains) a model for event recognition and a model for event classification. Secondly, given plain text, it automatically gets the defined features using the described tools. With this data, the system applies the learned models to recognize and classify TimeML events.

4 Evaluation

In this section, firstly, the corpus, criteria and measures are defined. Secondly, the results obtained by our approach are presented. After that, the contribution of our approach is measured through different experiments: (1) general contribution, (2) semantic roles contribution, and (3) CRFs contribution. And finally, our approach is compared to the state of the art systems.

4.1 Corpus, criteria and measures

For the evaluation, the TimeBank 1.2 corpus (7881 events) was used without modification. All the results reported in this evaluation were obtained using a 5-fold cross validation. The n-fold train-test sets were built sorting the corpus files alphabetically and then sequentially select each set regarding the documents size. It is important to highlight the latter because if the n-folds were made regarding the number of documents, the sets had not been homogeneous due to the differences in TimeBank document sizes.

Only annotations matching the exact event span were considered as *correct* in recognition and classification, requiring also the class matching in the second case.

The following measures were used to score the evaluated approaches.

- Precision $\frac{correct\_annotations}{total\_approach\_annotations}$
- Recall $\frac{correct\_annotation}{total\_corpus\_annotations}$
- $F_{\beta=1}$ $\frac{2 * precision * recall}{precision + recall}$

4.2 Our approach results

Table 1 shows the results obtained by our approach for both recognition and classification of events. The last column (BF) indicates the best $F_{\beta=1}$ results obtained in the individual folds.

	Precision	**Recall**	$\mathbf{F}_{\beta=1}$	BF
Recognition	83.43	79.54	**81.40**	82.43
Classification	68.84	60.15	**64.20**	69.68

Table 1: Our approach (CRF+Roles) results

The results show a high $F_{\beta=1}$ score in both recognition and classification, showing a good balance between precision and recall. This indicates that our approach is appropriate to address this task.

Focusing on classification task, Table 2 shows the detailed scores for each event class.

Looking at the specific class results, *reporting* obtained the best results. This is due to the fact that 80% of *reporting* events are represented by lemmas “say” and “report” with PoS “VBD” and “VBZ”. *Occurrence, perception, aspectual* and *i_state* obtained classification results over 50%.

Class (instances)	**Precision**	**Recall**	**F$_{\beta=1}$**
Reporting (1021)	91.90	89.18	90.51
Perception (48)	65.93	66.83	66.37
Aspectual (258)	81.35	47.00	59.57
I_Action (673)	51.40	29.30	37.32
I_State (582)	68.44	43.70	53.34
State (1107)	50.01	24.84	33.19
Occurrence (4192)	66.73	72.07	69.29

Table 2: CRF+Roles 5-fold detailed results

Although *perception* and *aspectual* are quite restricted to some lemmas, they obtained results below *reporting*. This is due to the fact that TimeBank contains very few examples of these classes. *I_action* and *state* show poorer results. In the case of the former, this is because some nonintensional verbs (e.g., "look") appear in the corpus as *i_action* under certain conditions, for example, when there is modality or these verbs appear in conditional sentences. This suggests the necessity of incorporating a word sense disambiguation (WSD) technique. Our approach did not take into account this information and thus the results are lower for this event class. In the case of *state*, the reasons for the low performance are the richness of this event class by means of lemmas, PoS, and phrases.

Finally, Table 3 shows the results of our approach by word class.

		Precision	**Recall**	**F$_{\beta=1}$**
Recognition	*Verb*	91.56	92.15	91.33
	Noun	72.67	48.26	58.42
	Adj.	66.78	38.09	48.35
Classification	*Verb*	73.86	74.21	73.51
	Noun	62.73	41.33	49.53
	Adj.	55.69	31.12	40.41

Table 3: CRF+Roles 5-fold word class results

It may be seen that the best results in both recognition and classification are obtained in verb events, followed by noun and adjective.

4.3 Contribution analysis

This subsection details the contribution of each aspect of our approach through three comparative experiments.

First experiment: general contribution

This experiment measures the general contribution of our approach by comparing its results with a baseline. TimeBank was analyzed to find a basic general rule to annotate events. The events are mainly denoted by verbs, pertaining to *occurrence* class. Hence, we propose a baseline that annotates all verbs as *occurrence* events. Table 4 shows results obtained by this baseline for both recognition and classification of events.

		Prec.	**Recall**	**F$_{\beta=1}$**
Our approach	*Recog.*	83.43	79.54	81.40
	Class.	68.84	60.15	64.20
Baseline	*Recog.*	72.50	65.20	68.60
	Class.	46.01	53.19	49.34

Table 4: Our approach vs Baseline results

Given the simplicity of the baseline, the results obtained are quite high. However, our approach F$_{\beta=1}$ significantly improves baseline by 19% for recognition and 30% for classification.

Second experiment: roles contribution

The main objective of this paper is to determine the impact of semantic roles in this task. To quantify it, a non-roles version of our approach was evaluated. This version only uses the general features described in section 3. Table 5 shows the results obtained.

		Precision	**Recall**	**F$_{\beta=1}$**
Our approach	*Recog.*	83.43	79.54	81.40
	Class.	68.84	60.15	64.20
Non-roles	*Recog.*	82.96	74.81	78.67
	Class.	67.53	54.80	60.50

Table 5: Our approach vs Non-roles results

Comparing these results with the ones obtained by our full featured approach, the application of roles improved especially the recall. Specifically, recall improved by 6% and 10% for recognition and classification respectively. The main improvement was achieved by *state* and *occurrence* classes (60% of the total improvement), especially, nominal events of that classes that concentrate around the 70% of the total contribution.

To illustrate corpus examples that have been improved by roles, Example 5 shows two sentences containing state events that were correctly tagged by the roles approach and missed by the non-roles. In the examples, the TimeML events annotation and below the semantic roles annotation is reported.

(5) "There are still few buyers and the mood is `<EVENT class=STATE>gloomy</EVENT>`"
"[There A0] are [still AM-TMP] [few buyers A1] and [the mood A0] is [gloomy AM-MNR]"

"Security is now `<EVENT>better</EVENT>`"
"[Security A0] is [now AM-TMP] [better AM-MNR]"

In these cases, AM-MNR role information lead to a correct *state* event recognition.

Third experiment: CRFs contribution

In order to measure the CRFs contribution to this task, an extra experiment was carried out. This consisted of comparing, under the same setting, CRFs with a popular learning technique: support vector machines (SVM). As in Bethard and Martin (2006), YamCha[4] software was used (parameters: C=1 and polynomial_degree=2).

Table 6 shows the results obtained by the SVM-based approach in recognition and Table 7 reports the improvement (CRFs over SVM) distribution in the different word classes.

	Precision	**Recall**	**$F_{\beta=1}$**
Our approach (CRF)	83.43	79.54	81.40
SVM	80.00	75.10	77.40

Table 6: Our approach (CRF) vs SVM results

	Verb	**Noun**	**Adj.**	**Adv.**	**Prep.**
General	22%	71%	5%	1%	1%

Table 7: CRF improvement distribution among the word classes

These results verify that CRF improves SVM $F_{\beta=1}$ by 5% in this task. Furthermore, especially noun events take advantage of using CRF.

Finally, Figure 2 illustrates the results of our approach over the described experiments.

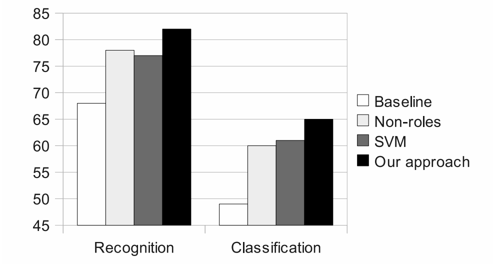


Figure 2: $F_{\beta=1}$ Results

4.4 Comparison with the state of the art

Most systems found in the literature are data-driven approaches using morphosyntactic features. SVM based approaches (Bethard and Martin, 2006; March and Baldwin, 2008) achieved, approximately, 76% and 58% $F_{\beta=1}$ in event recognition and classification respectively. Boguraev and Ando (2005) used a robust risk minimization classifier to address this task and obtained 78.6% and 61% (without exploiting unannotated data). These results are very similar to the ones obtained by our non-roles approach. This suggests that using, apart from morphosyntactic features, additional features based on semantic roles could improve the approaches.

EVITA system (Saurí et al., 2005) combines linguistic and statistical techniques. On the one hand, it consists of a set of manually encoded rules based on morphosyntactic information. On the other hand, it includes a Bayesian learned disambiguation module to identify nominal events. The later was trained and tested using the whole corpus, therefore, the results could be inflated by this fact. For that reason, Bethard and Martin (2006) presented an EVITA implementation (Sim-Evita) to compare the results. Sim-Evita obtains an 73% and 51% $F_{\beta=1}$ in event recognition and classification respectively. These results suggest that data-driven improve rule-based approaches.

Only STEP evaluation showed detailed classification results. We agree that *state* events are the most complex and heterogeneous ones. Focusing on such events, our $F_{\beta=1}$ results (33%) improve Bethard's (25%) by 32%. Regarding the results obtained for each word class. Bethard's results presented good performance on classifying verb events (71%), but lower results in noun events (34%). Our approach results for noun events (49%) improve theirs by 44%. This suggests that the application of semantic roles enables our approach on making more general predictions. In this manner, our system may recog-

[4] http://chasen.org/˜taku/software/YamCha/

nize unseen nominal event instances as long as they share, with the seen instances, some semantic roles features.

5 Conclusions and Further Work

This paper presented an approach for the recognition and classification of TimeML events consisting of a CRF model learned using semantic roles as main feature. In addition to morphosyntactic features, the model was enhanced including extra semantic information, semantic role labeling, used for other applications with satisfactory results, but never employed before for this purpose. Our proposal was evaluated using the gold standard corpus, TimeBank 1.2, and the results obtained were analyzed and compared to measure the impact of both semantic roles and CRFs in the described task.

The obtained $F_{\beta=1}$ results demonstrated that semantic roles are useful to recognize (81.43%) and classify (64.20%) TimeML events, improving the presented baseline by 19% for recognition and 30% for classification. Specifically, Semantic roles employed as additional feature improved the recall of the non-roles version by 6% and 10% for recognition and classification respectively. This indicates that roles features led to more general models capable of better annotating unseen instances. The roles contribution was more significant in *state* and *occurrence* classes of noun events, concentrating around the 70% of the improvement.

Furthermore, it was verified that CRFs achieve higher results than models learned using other ML techniques such as SVM (5% improvement), contributing especially to nominal events. This demonstrated that CRF models are appropriate to face the task.

Finally, to the extent our results are comparable to state of the art evaluations, ours outperform the $F_{\beta=1}$ scores in both recognition and classification. Especially, our approach showed better performance than related works in *state* (32% improvement) and nominal events (44% improvement). Hence, the extension of the current approaches with semantic roles features could benefit their performance.

The main difficulties found in the task addressed in this paper are related to *i_action* and *state* events. In the former, we detected that modality and the word senses are important and must be treated to distinguish such events. In the later, although they were improved by our approach, *state* events are still the most complex class of events due to their richness in contrast to the reduced size of the training data. We agree with related literature that event classification results are still below other tasks performance, which indicates that this task is inherently complex and more training data may lead to significant improvements.

As further work we propose, firstly, improving the *i_action* results by taking into account the modality considering the *AM-MOD* role, and the word senses using a WSD technique. Secondly, the application of FrameNet role set (finer granularity) to determine which kind of roles are better to improve the current event annotation systems.

Acknowledgments

This paper has been supported by the Spanish Government, projects TIN-2006-15265-C06-01, TIN-2009-13391-C04-01 and PROMETEO/2009/119, where Hector Llorens is funded (BES-2007-16256).

References

Allan, James. 2002. *Topic Detection and Tracking: Event-Based Information Organization*. Kluwer Academic Publishers, Norwell, MA, USA.

Baker, Collin F., Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *COLING-ACL*, pages 86–90.

Bethard, Steven and James H. Martin. 2006. Identification of event mentions and their semantic class. In *EMNLP: Proceedings of the Conference on Empirical Methods in NLP*, pages 146–154. ACL.

Boguraev, Branimir and Rie Kubota Ando. 2005. Effective Use of TimeBank for TimeML Analysis. In *Annotating, Extracting and Reasoning about Time and Events 05151*.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *43rd Annual Meeting of the ACL*.

Daniel, Naomi, Dragomir Radev, and Timothy Allison. 2003. Sub-event based multi-document summarization. In *HLT-NAACL Text summarization workshop*, pages 9–16. ACL.

Fellbaum, Christiane. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. MIT Press.

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Hagège, Caroline and Xavier Tannier. 2007. XRCE-T: XIP temporal module for TempEval campaign. In *TempEval (SemEval)*, pages 492–495, Prague, Czech Republic. ACL.

Lafferty, John D., Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th ICML*, pages 282–289. Morgan Kaufmann.

Llorens, Hector, Borja Navarro, and Estela Saquete. 2009. Using Semantic Networks to Identify Temporal Expressions from Semantic Roles. In *VI RANLP*, pages 219–224.

March, Olivia and Timothy Baldwin. 2008. Automatic event reference identification. In *ALTA 2008*, pages 79–87, Australia.

McCallum, Andrew and Wei Li. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *HLT-NAACL*, pages 188–191.

Melli, G., Y. Liu Z. Shi, Y. Wang, and F. Popowich. 2006. Description of SQUASH, the SFU Question Answering Summary Handler for the DUC-2006 Summarization Task. In *DUC*.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31.

Punyakanok, Vasin, Dan Roth, W. Yih, D. Zimak, and Y. Tu. 2004. Semantic role labeling via generalized inference over classifiers. In *HLT-NAACL (CoNLL)*, pages 130–133. ACL.

Pustejovsky, James, José M. Castaño, Robert Ingria, Roser Saurí, Robert Gaizauskas, Andrea Setzer, and Graham Katz. 2003a. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *IWCS-5*.

Pustejovsky, James, Patrik Hanks, Roser Saurí, A. See, Robert Gaizauskas, Andrea Setzer, Dragomir R. Radev, Beth Sundheim, David Day, Lisa Ferro, and M. Lazo. 2003b. The TIMEBANK Corpus. In *Corpus Linguistics*, pages 647–656.

Pustejovsky, James. 2002. TERQAS: Time and Event Recognition for Question Answering Systems. In *ARDA Workshop*.

Saurí, Roser, Robert Knippen, Marc Verhagen, and James Pustejovsky. 2005. Evita: A robust event recognizer for qa systems. In *HLT/EMNLP*. ACL.

Schmid, Helmut. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49.

Setzer, Andrea and Robert Gaizauskas. 2000. Annotating Events and Temporal Information in Newswire Texts. In *LREC 2000*, pages 1287–1294.

Sha, Fei and Fernando C. N. Pereira. 2003. Shallow parsing with conditional random fields. In *HLT-NAACL*.

Tenny, Carol and James Pustejovsky. 2000. *Events as Grammatical Objects. The Converging Perspectives of Lexical Semantics and Syntax*. CSLI.

Vendler, Zeno, 1967. *Linguistics and philosophy*, chapter Verbs and times, pages 97–121. Cornell University Press, Ithaca, NY.

Exploiting Structured Ontology to Organize Scattered Online Opinions

Yue Lu, Huizhong Duan, Hongning Wang, ChengXiang Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign
{yuelu2,duan9,wang296,czhai}@illinois.edu

Abstract

We study the problem of integrating scattered online opinions. For this purpose, we propose to exploit structured ontology to obtain well-formed relevant aspects to a topic and use them to organize scattered opinions to generate a structured summary. Particularly, we focus on two main challenges in implementing this idea, (1) how to select the most useful aspects from a large number of aspects in the ontology and (2) how to order the selected aspects to optimize the readability of the structured summary. We propose and explore several methods for solving these challenges. Experimental results on two different data sets (US Presidents and Digital Cameras) show that the proposed methods are effective for selecting aspects that can represent the major opinions and for generating coherent ordering of aspects.

1 Introduction

The explosive growth of online opinions raises interesting challenges for opinion integration and summarization. It is especially interesting to integrate and summarize scattered opinions in blog articles and forums as they tend to represent the general opinions of a large number of people and get refreshed quickly as people dynamically generate new content, making them valuable for understanding the current views of a topic.

However, opinions in blogs and forums are usually fragmental, scattered around, and buried among other off-topic content, so it is quite challenging to organize them in a meaningful way. Traditional text summarization techniques generate an unstructured list of sentences as a summary, which cannot reveal representative opinions on different aspects of a topic or effectively facilitate navigation into the huge opinion space. To address this limitation, recent work has shown the usefulness of generating a structured summary of opinions, in which related opinions are grouped into topical aspects with explicit labeling of all the aspects. A major challenge in producing such a structured summary is how to generate these aspects for an arbitrary topic (e.g., products, political figures, policies, etc.). Intuitively, the aspects should be concise phrases that can both be easily interpreted in the context of the topic under consideration and capture the major opinions. However, where can we find such phrases and which phrases should we select as aspects? Furthermore, once we selected aspects, how should we order them to improve the readability of a structured summary? One way to generate aspects is to cluster all the opinion sentences and then identify representative phrases in each cluster. Although aspects selected in this way can effectively capture the major opinions, a major limitation is that it is generally hard to ensure that the selected phrases are well connected with the given topic (Chen and Dumais, 2000).

In this paper, we propose a novel approach to generating aspects by leveraging the ontologies with structured information that are available online, such as open domain knowledge base in Freebase[1]. Such kind of ontology data is not in small scale by any measure. For example, Freebase alone contains more than 10 million topics, 3000 types, and 30,000 properties; moreover, it is constantly growing as people collaboratively contribute. Freebase provides different properties for different types of topics such as personal information for a “US President” and product features for a “Digital Camera”. Since this kind of resources can provide related entities/relations for a

[1] http://www.freebase.com

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 734–742,
Beijing, August 2010

wide range of topics , our general idea is to leverage them as guidance for more informed organization of scattered online opinions, and in particular, to select the most important properties of a topic from such structured ontology as aspects to generate a structured opinion summary. A significant advantage of this approach to aspect generation is that the selected aspects are guaranteed to be very well connected with the topic, but it also raises an additional challenge in selecting the aspects to best capture the major opinions from a large number of aspects provided for each topic in the ontology. Different from some existing work on exploiting ontologies, e.g., (Sauper and Barzilay, 2009), which relies on training data, we focus on exploring *unsupervised* approaches, which can be applied to a larger scope of topics.

Specifically, given a topic with entries in an ontology and a collection of scattered online opinions about the topic, our goal is to generate a structured summary where representative major opinions are organized with well aligned aspects and in an order easy for human to follow. We propose the following general approach: First, retrieval techniques are employed to align opinions to relevant aspects. Second, a subset of most interesting aspects are selected. Third, we will further order the selected aspects to present them in a reasonable order. Finally, for the opinions uncovered by the selected aspects from the ontology, we use a phrase ranking method to suggest new aspects to add to the ontology for increasing its coverage.

Implementing the second and third steps involves special challenges. In particular, without any training data, it is unclear how we should show the most interesting aspects in ontology with major opinions aligned and which presentation order of aspects is natural and intuitive for human. Solving these two challenges is the main focus of this paper. We propose three methods for aspect selection, i.e., size-based, opinion coverage-based, and conditional entropy-based methods, and two methods for aspect ordering, i.e., ontology-ordering and coherence ordering. We evaluate our methods on two different types of topics: US Presidents and Digital Cameras. Qualitative results demonstrate the utility of integrating opinions based on structured ontology as well as the generalizability of proposed methods. Quantitative evaluation is also conducted to show the effectiveness of our methods.

Note that we use the term ”opinion” to broadly refer to any discussion in opinionated sources such as blogs and reviews. This allows us to formulate and solve the problem in a general way. Indeed, the main goal of our work is to extract and organize the major opinions about a topic that are buried in many scattered opinionated sources rather than perform deeper understanding of opinions (e.g., distinguishing positive from negative opinions), which can be done by using any existing sentiment analysis technique as an orthogonal post-processing step after applying our method.

2 Related Work

Aspect summarization, i.e., structured opinion summarization over topical aspects, has attracted much attention recently. Existing work identifies aspects using frequent-pattern/association-rule mining, e.g. (Liu et al., 2005; Popescu and Etzioni, 2005), sentence clustering, e.g. (Gamon et al., 2005; Leouski and Croft, 1996), or topic modeling, e.g. (Mei et al., 2006; Titov and McDonald, 2008). After that, meaningful and prominent phrases need to be selected to represent the aspects, e.g. (Zhao and He, 2006; Mei et al., 2007). However, these methods suffer from the problem of producing trivial aspects. Consequently, some of the aspects generated are very difficult to interpret (Chen and Dumais, 2000). In this paper, we propose a different kind of approach that is to use aspects provided by ontology which are known to be relevant and easy to interpret.

Ontology is used in (Carenini et al., 2005) but only for mapping product features. The closest work to ours are (Lu and Zhai, 2008; Sauper and Barzilay, 2009); both try to use well-written articles for summarization. However, (Lu and Zhai, 2008) assumes the well-written article is structured with explicit or implicit aspect information, which does not always hold in practice, while (Sauper and Barzilay, 2009) needs a relatively large amount of training data in the given domain. In comparison, our work only needs the ontology information for the given topic which is much easier to obtain from resources such as Freebase.

3 Methods

Given (1) an input topic T, (2) a large number of aspects/properties $A = \{A_1, ..., A_m\}$ from an ontology that are related to T, and (3) a huge collection of scattered opinion sentences about the topic $D_T = \{s_1, \ldots, s_n\}$, our goal is to generate a structured organization of opinions that are both aligned well with the interesting aspects and representative of major opinions about the topic.

The envisioned structured organization consists of a sequence of selected aspects from ontology ordered to optimize readability and a set of sentences matching each selected aspect. Once we obtain a set of sentences in each aspect, we can easily apply a standard text summarization method to further summarize these sentences, thus the unique challenges related to our main idea of exploiting ontology are the following, which are also the main focus of our study:

Aspect Selection: How can we select a subset of aspects $A' \subset A$ to capture the *major* opinions in our opinion set D_T?

Aspect Ordering: How can we order a subset of selected aspects A' so as to present them in an order $\pi(A')$ that is most natural with respect to human perception?

New Aspects Suggestion: Can we exploit the opinions in D_T to suggest new aspects to be added to the ontology?

3.1 Aspect Selection

In order to align the scattered opinions to the most relevant aspects, we first use each aspect label $A_i \in A$ as a query to retrieve a set of relevant opinions in the collection $S_i \subseteq D_T$ with a standard language modeling approach, i.e., the KL-divergence retrieval model (Zhai and Lafferty, 2001). Up to 1000 opinion sentences are retrieved for each aspect; each opinion sentence can be potentially aligned to several aspects. In this way, scattered online discussion are linked to the most relevant aspects in the ontology, which enables a user to use aspects as "semantic bridges" to navigate into the opinion space..

However, there are usually a lot of candidate aspects in an ontology, and only some are heavily commented in online discussions, so showing all the aspects is not only unnecessary, but also overwhelming for users. To solve this problem, we propose to utilize the aligned opinions to further select a subset of the most interesting aspects $A' \subset A$ with size k. Several approaches are possible for this subset selection problem.

Size-based: Intuitively, the selected subset A' should reflect the major opinions. So a straightforward method is to order the aspects A_i by the size of the aligned opinion sentences S_i, i.e., the number of relevant opinion sentences, and then select the top k ones.

Opinion Coverage-based: The previous method does not consider possible redundancy among the aspects. A better approach is to select the subset that covers as many *distinct* opinion sentences as possible. This can be formulated as a maximum coverage problem, for which a greedy algorithm is known to be a good approximation: we select one aspect at a time that is aligned with the largest number of uncovered sentences.

Conditional Entropy-based: Aspects from a structured ontology are generally quite meaningful, but they are not designed specifically for organizing the opinions in our data set. Thus, they do not necessarily correspond well to the natural clusters in scattered opinions. To obtain aspects that are aligned well with the natural clusters in scattered opinions, we can first cluster D_T into l clusters $C = \{C_1, \ldots, C_l\}$ using K-means with $TF \times IDF$ as features, and then choose the subset of aspects that minimize Conditional Entropy of the cluster label given the aspect:

$$\begin{aligned} A' &= \arg\min H(C|A') = \arg\min \\ &\left[- \sum_{A_i \in A', C_i \in C} p(A_i, C_i) \log \frac{p(A_i, C_i)}{p(A_i)} \right] \end{aligned}$$

This Conditional Entropy measures the uncertainty about the cluster label of a sentence given the knowledge of its aspect. Intuitively, if the aspects are aligned well with the clusters, we would be able to predict well the cluster label of a sentence if we know its aspect, thus there would be less uncertainty about the cluster label. In the extreme case when the cluster label can be completely determined by the aspect, the conditional entropy would reach its minimum (i.e., 0). Intuitively, the conditional entropy-based method essentially selects the most appropriate aspects from

Algorithm 1 Greedy Algorithm for Conditional Entropy Based Aspect Selection

```
Input: $A = \{A_1, ..., A_m\}$
Output: $k$-sized $A' \subseteq A$
$A' = \{\cup_{i=1}^{m} A_i\}$
for j=1 to $k$ do
    $bestH = \infty$; $bestA = A_0$
    for each $A_i$ in $A$ do
        $tempA' = \{A_i, A' \setminus A_i\}$
        if $H(C|tempA') < bestH$ then
            $bestH = H(C|tempA')$
            $bestA = A_i$
    $A' = \{bestA, A' \setminus bestA\}$
output $A'$
```

the ontology to label clusters of opinions.

The exact solution of this combinatorial optimization problem is NP-complete, so we employ a polynomial time greedy algorithm to approximate it: in the i-th iteration, we select the aspect that can minimize the conditional entropy given the previous $i - 1$ selected aspects. Pseudo code is given in Algorithm 1.

3.2 Aspect Ordering

In order to present the selected aspects to users in a most natural way, it is important to obtain a coherent order of them, i.e., generating an order consistent with human perception. To achieve this goal, our idea is to use human written articles on the topic to learn how to organize the aspects automatically. Specifically, we would order aspects so that the relative order of the sentences in all the aspects would be as consistent with their order in the original online discussions as possible.

Formally, the input is a subset of selected aspects A'; each $A_i \in A'$ is aligned with a set of relevant opinion sentences $S_i = \{S_{i,1}, S_{i,2}, ...\}$. We define a coherence measurement function over sentence pairs $Co(S_{i,k}, S_{j,l})$, which is set to 1 iff $S_{i,k}$ appears before $S_{j,l}$ in the same article. Otherwise, it is set to 0. Then a coherence measurement function over an aspect pair can be calculated as

$$Co(A_i, A_j) = \frac{\sum_{S_{i,k} \in S_i, S_{j,l} \in S_j} Co(S_{i,k}, S_{j,l})}{|S_i||S_j|}$$

As an output, we would like to find a permutation $\hat{\pi}(A')$ that maximizes the coherence of all pairwise aspects, i.e.,

$$\hat{\pi}(A') = \arg\max_{\pi(A')} \sum_{A_i, A_j \in A', A_i \prec A_j} Co(A_i, A_j)$$

Algorithm 2 Greedy Algorithm for Coherence Based Aspect Ordering

```
Input: $A$
Output: $\pi(A)$
for each $A_i, A_j$ in $A$ do
    calculate $Co(A_i, A_j)$
for $p = 1$ to $len = A.size()$ do
    $Max = A[1]$
    for each aspect $A_i$ in $A$ do
        $A_i.coherence = 0$
        for each aspect $A_j$ in $\pi(A)$ do
            $A_i.coherence+ = Co(A_j, A_i)$
        for each aspect $A_j$ in $A, j \neq i$ do
            $A_i.coherence+ = Co(A_i, A_j)$
        if $A_i.coherence > Max.coherence$ then
            $Max = A_i$
    remove $Max$ from $A$; add $Max$ to $\pi(A)$
output $\pi(A)$
```

where $A_i \prec A_j$ means that A_i is before A_j. It is easy to prove that the problem is NP-complete. Therefore, we resort to greedy algorithms to find approximations of the solution. Particularly we view the problem as a ranking problem. The algorithm proceeds by finding at each ranking position an aspect that can maximize the coherence measurement, starting from the top of the rank list. The detailed algorithm is given in Algorithm 2.

3.3 New Aspects Suggestion

Finally, if the opinions cover more aspects than in the ontology, we also want to identify informative phrases to label such extra aspects; such phrases can also be used to further augment the ontology with new aspects.

This problem is similar to existing work on generating labels for clusters (Zeng et al., 2004) or topic models (Mei et al., 2007). Here we employ a simple but representative technique to demonstrate the feasibility of discovering interesting new aspects for augmenting the ontology. We first extract named entities from scattered opinions D_T using Stanford Named Entity Recognizer (Finkel et al., 2005). After that, we rank the phrases by pointwise Mutual Information (MI):

$$MI(T, ph) = \log \frac{P(T, ph)}{P(T)P(ph)}$$

where T is the given topic and ph refers to a candidate entity phrase. $P(T, ph)$ is proportional to the number of opinion sentences they co-occur; $P(T)$ or $P(ph)$ are proportional to the number of times T or ph appears. A higher MI value indicates a

Statistics	Category 1 US president	Category 2 Digital Camera
Number of Topics	36	110
Number of Aspects	65±26	32±4
Number of Opinions	1001±1542	170±249

Table 1: Statistics of Data Sets

stronger association. We can then suggest the top ranked entity phrases that are not in the selected aspects as new aspects.

4 Experiments

4.1 Data Sets

To examine the generalizability of our methods, we test on two very different categories of topics: *US Presidents* and *Digital Cameras*.[2] For the ontology, we leverage Freebase, downloading the structured ontology for each topic. For the opinion corpus, we use blog data for US Presidents and customer reviews for Digital Cameras. The blog entries for US Presidents were collected by using Google Blog Search[3] with the name of a president as the query. Customer reviews for Digital Cameras were crawled from CNET[4]. The basic statistics of our data sets is shown in Table 1. For all the data collections, Porter stemmer (Porter, 1997) is applied and stop words are removed.

4.2 Sample Results

We first show sample results of automatic organization of online opinions. We use the opinion coverage-based algorithm to select 10 aspects (10-20 aspects were found to be optimal in (Käki, 2005)) and then apply the coherence-based aspect ordering method. The number of clusters is set so that there are on average 15 opinions per cluster.

Opinion Organization: Table 2 and Table 3 present sample results for President Ronald Reagan and Sony Cybershot DSC-W200 camera respectively[5]. We can see that (1) although Freebase aspects provide objective and accurate information about the given topics, extracted opinion sentences offer additional subjective information; (2) aligning scattered opinion sentences to most relevant aspects in the ontology helps digestion and navigation; and (3) the support number, which is the number of opinion sentences aligned to an aspect, can show the popularity of the aspect in the online discussions.

Adaptability of Aspect Selection: Being unsupervised is a significant advantage of our methods over most existing work. It provides flexibility of applying the methods in different domains without the requirement of training data, benefiting from both the ontology based template guidance as well as data-driven approaches. As a result, we can generate different results for different topics even in the same domain. In Table 4, we show the top three selected and ordered aspects for Abraham Lincoln and Richard Nixon. Although they belong to the same category, different aspects are picked up due to the differences in online opinions. People talk a lot about Lincoln's role in American Civil War and his famous quotation, but when talking about Nixon, people focus on ending the Vietnam war and the Watergate scandal. “Date of birth” and “Government position” are ranked first because people tend to start talking from these aspects, which is more natural than starting from aspects like “Place of death”.

Baseline Comparison: We also show below the aspects for Lincoln generated by a representative approach using clustering method (e.g. (Gamon et al., 2005)). i.e., we label the largest clusters by selecting phrases with top mutual information. We can see that although some phrases make sense, not all are well connected with the given topic; using aspects in ontology circumvents this problem. This example confirms the finding in previous work that the popular existing clustering-based approach to aspects generation cannot generate meaningful labels (Chen and Dumais, 2000).

```
Vincent
New Salem State Historic Site
USS Abraham Lincoln
Martin Luther King Jr
Gettysburg
John F.
```

New Aspect Discovery: Finally, in Table 5 we show some phrases ranked among top 10 using the method described in Section 3.3. They reveal additional aspects covered in online discussions and serve as candidate new aspects to be added to Freebase. Interestingly, John Wilkes Booth, who assassinated President Lincoln, is not explicitly

[2] We have made our data sets available at http://timan.cs.uiuc.edu/downloads.html .

[3] http://blogsearch.google.com

[4] http://www.cnet.com

[5] Due to space limit, we only show the first few aspects as output by our methods.

FreeBase Aspects	**Supt**	**Representative Opinion Sentences**
Appointees: - Martin Feldstein - Chief Economic Advisor	897	Martin Feldstein, whose criticism of Reagan era deficits has not been forgotten. Reagan's first National Security advisor was quoted as declaring...
Government Positions Held: - President of the United States - Jan 20, 1981 to Jan 20, 1989	967	1981 Jan 20, Ronald Reagan was sworn in as president as 52 American hostages boarded a plane in Tehran and headed toward freedom. 40th president of the US Ronald Reagan broke the so called "20 year curse"...
Vice president: - George H. W. Bush	847	8 years, 1981-1988 George H. W. Bush as vice president under Ronald Reagan... ...exception to the rule was in 1976, when George H W Bush beat Ronald.

Table 2: Opinion Organization Result for President Ronald Reagan

FreeBase Aspects	**Supt**	**Representative Opinion Sentences**
Format: - Compact	13	Quality pictures in a compact package. ... amazing is that this is such a small and compact unit but packs so much power.
Supported Storage Types: - Memory Stick Duo	11	This camera can use Memory Stick Pro Duo up to 8 GB Using a universal storage card and cable (c'mon Sony)
Sensor type: - CCD	10	I think the larger ccd makes a difference. but remember this is a small CCD in a compact point-and-shoot.
Digital zoom: -2×	47	once the digital :smart" zoom kicks in you get another 3x of zoom I would like a higher optical zoom, the W200 does a great digital zoom translation...

Table 3: Opinion Organization Result for Sony Cybershot DSC-W200 Camera

listed in Freebase, but we can find it in people's online discussion using mutual information.

4.3 Evaluation of Aspect Selection

Measures: Aspect selection is a new challenge, so there is no standard way to evaluate it. It is also very hard for human to read all of the aspects and opinions and then select a gold standard subset. Therefore, we opt to use indirect measures capturing different characteristics of the aspect selection problem (1) *Aspect Coverage* (AC): we first assign each aspect A_i to the cluster C_j that has the most overlapping sentences with A_i, approximating the cluster that would come into mind when a reader sees A_i. Then AC is defined as the percentage of the clusters covered by at least one aspect. (2) *Aspect Precision* (AP): for each covered cluster C_i, AP measures the Jaccard similarity between C_i as a set of opinions and the union of all aspects assigned to C_i. (3) *Average Aspect Precision* (AAP): defines averaged AP for all clusters where an uncovered C_i has a zero AP; it essentially combines AC and AP. We also report *Sentence Coverage* (SC), i.e., how many distinct opinion sentences can be covered by the selected aspects and *Conditional Entropy* (H), i.e., how well the selected aspects align with the natural clusters in the opinions; a smaller H value indicates a better alignment.

Results: We summarize the evaluation results in Table 6. In addition to the three methods described in Section 3.1, we also include one baseline of averaging 10 runs of random selection. The best performance by each measure on each data set is highlighted in bold font. Not surprisingly, opinion coverage-based approach has the best sentence coverage (SC) performance and conditional entropy-based greedy algorithm achieves the lowest H. Size-based approach is best in aspect precision but at the cost of lowest aspect coverage. The trade-off between AP and AC is comparable to that between precision and recall as in information retrieval while AAP summarizes the combination of these two. The greedy algorithm based on conditional entropy outperforms all other approaches in AC and also in AAP, suggesting that it can provide a good balance between AP and AC.

Measures	**SC**	**H**	**AC**	**AP**	**AAP**
PRESIDENTS					
Random	503	1.9069	0.5140	0.0933	0.1223
Size-based	500	1.9656	0.3108	**0.1508**	0.0949
Opin Cover	**746**	1.8852	0.5463	0.0913	0.1316
Cond Ent.	479	**1.7687**	**0.5770**	0.0856	**0.1552**
CAMERAS					
Random	55	1.6389	0.6554	0.0871	0.1271
Size-based	70	1.6463	0.6071	**0.1077**	0.1340
Opin Cover	**82**	1.5866	0.6998	0.0914	0.1564
Cond Ent.	70	**1.5598**	**0.7497**	0.0789	**0.1574**

Table 6: Evaluation Results for Aspect Selection

Supt	**Richard-Nixon**	Supt	**Abraham-Lincoln**
50	Date of birth: - Jan 9, 1913	419	Government Positions Held: - United States Representative Mar 4,1847-Mar 3,1849
108	Tracks Recorded: - 23-73 Broadcast: End of the Vietnam War	558	Military Commands: - American Civil War - United States of America
120	Works Written About This Topic: - Watergate	810	Quotations: - Nearly all men can stand adversity, but if you want to test a man's character, give him power.

Table 4: Comparison of Aspect Selection for Two Presidents (aligned opinions are omitted here)

Suggested Phrases	**Supporting Opinion Sentences**
Abraham Lincoln Presidential Library	CDB projects include the Abraham Lincoln Presidential Library and Museum
Abraham Lincoln Memorial	..., eventually arriving at Abraham Lincoln Memorial.
John Wilkes Booth	John Wilkes Booth shoots President Abraham Lincoln at Ford's Theatre ...

Table 5: New Phrases for Abraham Lincoln

4.4 Evaluation of Aspect Ordering

Human Annotation: In order to quantitatively evaluate the effectiveness of aspect ordering, we conduct user studies to establish gold standard ordering. Three users were each given k selected aspects and asked to perform two tasks for each US President: (1) identify clusters of aspects that are more natural to be presented together (*cluster constraints*) and (2) identify aspect pairs where one aspect is preferred to appear before the other from the viewpoint of readability. (*order constraints*). We did not ask them to provide a full order of the k aspects, because we suspect that there are usually more than one "perfect" order. Instead, identifying partial orders or constraints is easier for human to perform, thus provides more robust gold standard.

Human Agreement: After obtaining the human annotation results, we first study human consensus on the ordering task. For both types of human identified constraints, we convert them into pairwise relations of aspects, e.g., "A_i and A_j should be presented together" or "A_i should be displayed before A_j". Then we calculate the agreement percentage among the three users. In Table 7, we can see that only a very small percentage of pair-wise partial orders (15.92% of the cluster constraints and none of the order constraints) are agreed by all the three users, though the agreement of clustering is much higher than that of ordering. This indicates that ordering the aspects is a subjective and difficult task.

AgreedBy	**Cluster Constraint**	**Order Constraint**
1	37.14%	89.22%
2	46.95%	10.78%
3	15.92%	0.00%

Table 7: Human Agreement on Ordering

Measures: Given the human generated gold standard of partial constraints, we use the following measures to evaluate the automatically generated full ordering of aspects: (1) *Cluster Precision* (pr_c): for all the aspect pairs placed in the same cluster by human, we calculate the percentage of them that are also placed together in the system output. (2) *Cluster Penalty* (p_c): for each aspect pair placed in the same cluster by human, we give a linear penalty proportional to the number of aspects in between the pair that the system places; p_c can be interpreted as the average number of aspects between aspect pairs that should be presented together in the case of misordering. Smaller penalty corresponds to better ordering performance. (3) *Order Precision* (pr_o): the percentage of correctly predicted aspect pairs compared with human specified order.

Results: In Table 8, we report the ordering performance based on two selection algorithms: opinion coverage-based and conditional entropy-based. Different selection algorithms provide different subsets of aspects for the ordering algorithms to operate on. For comparison with our coherence-based ordering algorithm, we include a random baseline and Freebase ontology ordering. Note that Freebase order is a very strong baseline because it is edited by human even though the purpose was not for organizing opinions. To take into account the variation of human annotation, we use four versions of gold standard: three are from the individual annotators and one from the union of their annotation. We did not include the gold stan-

Selection	**Gold**	**Cluster**	**Precision**	**(pr_c)**	**Cluster**	**Penalty**	**(p_c)**	**Order**	**Precision**	**(pr_o)**
Algo	**STD**	**Random**	**Freebase**	**Coherence**	**Random**	**Freebase**	**Coherence**	**Random**	**Freebase**	**Coherence**
Opin Cover	1	0.3290	**0.9547**	0.9505	1.8798	0.1547	**0.1068**	0.4804	**0.7059**	0.4510
Opin Cover	2	0.3266	**0.9293**	0.8838	1.7944	0.3283	**0.1818**	**0.4600**	0.4000	0.4000
Opin Cover	3	0.2038	**0.4550**	0.4417	2.5208	**1.3628**	1.7994	0.5202	0.4561	**0.5263**
Opin Cover	union	0.3234	**0.7859**	0.7237	1.8378	0.6346	**0.4609**	**0.4678**	0.4635	0.4526
Cond Entropy	1	0.2540	**0.9355**	0.8978	2.0656	0.2957	**0.2016**	0.5106	**0.7111**	0.5444
Cond Entropy	2	0.2535	0.7758	**0.8323**	2.1790	0.7530	**0.5222**	0.4759	**0.6759**	0.5093
Cond Entropy	3	0.2523	0.4030	**0.5545**	2.3079	2.1328	**1.1611**	0.5294	0.7143	**0.8175**
Cond Entropy	union	0.3067	0.7268	**0.7488**	1.9735	1.0720	**0.7196**	0.5006	0.6500	**0.6833**

Table 8: Evaluation Results on Aspect Ordering

dard that is the intersection of three annotators because that would leave us with too little overlap.

We have several observations: (1) In general, results show large variations when using different versions of gold standard, indicating the subjective nature of the ordering task. (2) Coherence-based ordering shows similar performance to Freebase order-based in cluster precision (pr_c), but when we take into consideration the distance-based penalty (p_c) of separating aspects pairs in the same cluster, coherence-based ordering is almost always significantly better except in one case. This shows that our method can effectively learn the coherence of aspects based on how their aligned opinion sentences are presented in online discussions. (3) Order precision (pr_o) can hardly distinguish different ordering algorithm. This indicates that people vary a lot in their preferences as which aspects should be presented first. However, in cases when the random baseline outperforms others the margin is fairly small, while Freebase order and coherence-based order have a much larger margin of improvement when showing superior performance.

5 Conclusions and Future Work

A major challenge in automatic integration of scattered online opinions is how to organize all the diverse opinions in a meaningful way for any given topic. In this paper, we propose to solve this challenge by exploiting related aspects in structured ontology which are guaranteed to be meaningful and well connected to the topic. We proposed three different methods for selecting a subset of aspects from the ontology that can best capture the major opinions, including size-based, opinion coverage-based, and conditional entropy-based methods. We also explored two ways to order aspects, i.e., ontology-order and coherence optimization. In addition, we also proposed appropriate measures for quantitative evaluation of both aspect selection and ordering.

Experimental evaluation on two data sets (US President and Digital Cameras) shows that by exploiting structured ontology, we can generate interesting aspects to organize scattered opinions. The conditional entropy method is shown to be most effective for aspect selection, and the coherence optimization method is more effective than ontology-order in optimizing the coherence of the aspect ordering, though ontology-order also appears to perform reasonably well. In addition, by extracting salient phrases from the major opinions that cannot be covered well by any aspect in an existing ontology, we can also discover interesting new aspects to extend the existing ontology.

Complementary with most existing summarization work, this work proposes a new direction of using structured information to organize and summarize unstructured opinions, opening up many interesting future research directions. For instance, in order to focus on studying aspect selection and ordering, we have not tried to optimize sentences matching with aspects in the ontology; it would be very interesting to further study how to accurately retrieve sentences matching each aspect. Another promising future work is to organize opinions using both structured ontology information and well-written overview articles.

Acknowledgment

We thank the anonymous reviewers for their useful comments. This paper is based upon work supported in part by an IBM Faculty Award, an Alfred P. Sloan Research Fellowship, an AFOSR MURI Grant FA9550-08-1-0265, and by the National Science Foundation under grants IIS-0347933, IIS-0713581, IIS-0713571, and CNS-0834709.

References

Carenini, Giuseppe, Raymond T. Ng, and Ed Zwart. 2005. Extracting knowledge from evaluative text. In *K-CAP '05: Proceedings of the 3rd international conference on Knowledge capture*, pages 11–18, New York, NY, USA. ACM.

Chen, Hao and Susan Dumais. 2000. Bringing order to the web: automatically categorizing search results. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 145–152, New York, NY, USA. ACM.

Finkel, Jenny Rose, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA. Association for Computational Linguistics.

Gamon, Michael, Anthony Aue, Simon Corston-Oliver, and Eric K. Ringger. 2005. Pulse: Mining customer opinions from free text. In Famili, A. Fazel, Joost N. Kok, José María Peña, Arno Siebes, and A. J. Feelders, editors, *IDA*, volume 3646 of *Lecture Notes in Computer Science*, pages 121–132. Springer.

Käki, Mika. 2005. Optimizing the number of search result categories. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1517–1520, New York, NY, USA. ACM.

Leouski, Anton V. and W. Bruce Croft. 1996. An evaluation of techniques for clustering search results. Technical report.

Liu, Bing, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 342–351, New York, NY, USA. ACM.

Lu, Yue and Chengxiang Zhai. 2008. Opinion integration through semi-supervised topic modeling. In Huai, Jinpeng, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors, *WWW*, pages 121–130. ACM.

Mei, Qiaozhu, Chao Liu, Hang Su, and ChengXiang Zhai. 2006. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 533–542.

Mei, Qiaozhu, Xuehua Shen, and ChengXiang Zhai. 2007. Automatic labeling of multinomial topic models. In Berkhin, Pavel, Rich Caruana, and Xindong Wu, editors, *KDD*, pages 490–499. ACM.

Pang, Bo and Lillian Lee. 2007. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Popescu, Ana-Maria and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *HLT '05*, pages 339–346, Morristown, NJ, USA. Association for Computational Linguistics.

Porter, M. F. 1997. An algorithm for suffix stripping. pages 313–316.

Sauper, Christina and Regina Barzilay. 2009. Automatically generating wikipedia articles: A structure-aware approach. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 208–216, Suntec, Singapore, August. Association for Computational Linguistics.

Titov, Ivan and Ryan McDonald. 2008. Modeling online reviews with multi-grain topic models. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 111–120, New York, NY, USA. ACM.

Zeng, Hua-Jun, Qi-Cai He, Zheng Chen, Wei-Ying Ma, and Jinwen Ma. 2004. Learning to cluster web search results. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217, New York, NY, USA. ACM.

Zhai, Chengxiang and John Lafferty. 2001. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of CIKM 2001*, pages 403–410.

Zhao, Jing and Jing He. 2006. Learning to generate labels for organizing search results from a domain-specified corpus. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 390–396, Washington, DC, USA. IEEE Computer Society.

Enhancing Morphological Alignment for Translating Highly Inflected Languages [*]

Minh-Thang Luong
School of Computing
National University of Singapore
luongmin@comp.nus.edu.sg

Min-Yen Kan
School of Computing
National University of Singapore
kanmy@comp.nus.edu.sg

Abstract

We propose an unsupervised approach utilizing only raw corpora to enhance morphological alignment involving highly inflected languages. Our method focuses on *closed-class morphemes*, modeling their influence on nearby words. Our language-independent model recovers important links missing in the IBM Model 4 alignment and demonstrates improved end-to-end translations for English-Finnish and English-Hungarian.

1 Introduction

Modern statistical machine translation (SMT) systems, regardless of whether they are word-, phrase- or syntax-based, typically use the word as the atomic unit of translation. While this approach works when translating between languages with limited morphology such as English and French, it has been found inadequate for morphologically-rich languages like Arabic, Czech and Finnish (Lee, 2004; Goldwater and McClosky, 2005; Yang and Kirchhoff, 2006). As a result, a line of SMT research has worked to incorporate morphological analysis to gain access to information encoded within individual words.

In a typical MT process, word aligned data is fed as training data to create a translation model. In cases where a highly inflected language is involved, the current word-based alignment approaches produce low-quality alignment, as the statistical correspondences between source and target words are diffused over many morphological forms. This problem has a direct impact on end translation quality.

Our work addresses this shortcoming by proposing a morphologically sensitive approach to word alignment for language pairs involving a highly inflected language. In particular, our method focuses on a set of *closed-class morphemes* (CCMs), modeling their influence on nearby words. With the model, we correct erroneous alignments in the initial IBM Model 4 runs and add new alignments, which results in improved translation quality.

After reviewing related work, we give a case study for morpheme alignment in Section 3. Section 4 presents our four-step approach to construct and incorporate our CCM alignment model into the grow-diag process. Section 5 describes experiments, while Section 6 analyzes the system merits. We conclude with suggestions for future work.

2 Related Work

MT alignment has been an active research area. One can categorize previous approaches into those that use language-specific syntactic information and those that do not. Syntactic parse trees have been used to enhance alignment (Zhang and Gildea, 2005; Cherry and Lin, 2007; DeNero and Klein, 2007; Zhang et al., 2008; Haghighi et al., 2009). With syntactic knowledge, modeling long distance reordering is possible as the search space is confined to plausible syntactic variants. However, they generally require language-specific tools and annotated data, making such approaches infeasible for many languages. Works that follow non-syntactic approaches, such as (Matusov et al.,

[*] This work was supported by a National Research Foundation grant "Interactive Media Search" (grant # R-252-000-325-279)

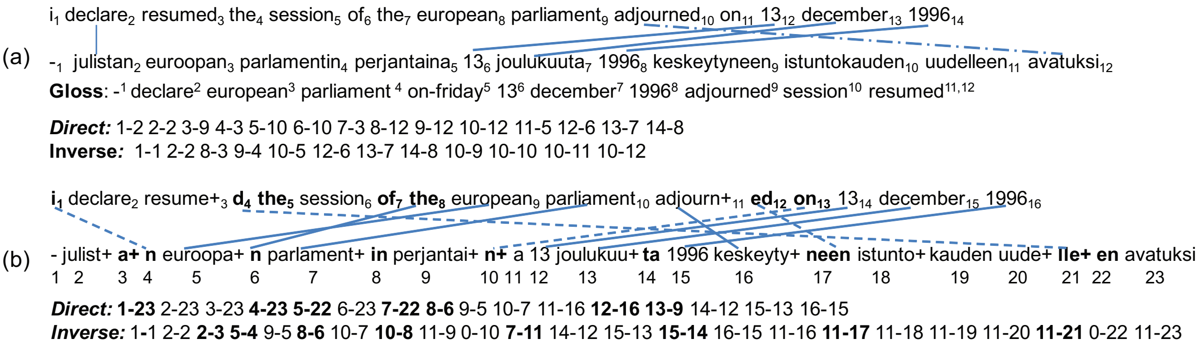

i1 declare2 resumed3 the4 session5 of6 the7 european8 parliament9 adjourned10 on11 13 12 december13 1996 14
(a)
-1 julistan2 euroopan3 parlamentin4 perjantaina5 13 6 joulukuuta7 1996 8 keskeytyneen9 istuntokauden10 uudelleen11 avatuksi12
Gloss: -1 declare2 european3 parliament4 on-friday5 13 6 december7 1996 8 adjourned9 session10 resumed11,12
Direct: 1-2 2-2 3-9 4-3 5-10 6-10 7-3 8-12 9-12 10-12 11-5 12-6 13-7 14-8
Inverse: 1-1 2-2 8-3 9-4 10-5 12-6 13-7 14-8 10-9 10-10 10-11 10-12
i1 declare2 resume+3 d4 the5 session6 of7 the8 european9 parliament10 adjourn+11 ed12 on13 13 14 december15 1996 16
(b)
- julist+ a+ n euroopa+ n parlament+ in perjantai+ n+ a 13 joulukuu+ ta 1996 keskeyty+ neen istunto+ kauden uude+ lle+ en avatuksi
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Direct: 1-23 2-23 3-23 4-23 5-22 6-23 7-22 8-6 9-5 10-7 11-16 12-16 13-9 14-12 15-13 16-15
Inverse: 1-1 2-2 2-3 5-4 9-5 8-6 10-7 10-8 11-9 0-10 7-11 14-12 15-13 15-14 16-15 11-16 11-17 11-18 11-19 11-20 11-21 0-22 11-23

Figure 1: **Sample English-Finnish IBM Model 4 alignments**: (a) word-level and (b) morpheme-level. Solid lines indicate intersection alignments, while the exhaustive asymmetric alignments are listed below. In (a), translation glosses for Finnish are given; the dash-dot line is the incorrect alignment. In (b), bolded texts are closed-class morphemes (CCM), while bolded indices indicate alignments involving CCMs. The dotted lines are correct CCM alignments not found by IBM Model 4.

2004; Liang et al., 2006; Ganchev et al., 2008), which aim to achieve symmetric word alignment during training, though good in many cases, are not designed to tackle highly inflected languages.

Our work differs from these by taking a middle road. Instead of modifying the alignment algorithm directly, we preprocess asymmetric alignments to improve the input to the symmetrizing process later. Also, our approach does not make use of specific language resources, relying only on unsupervised morphological analysis.

3 A Case for Morpheme Alignment

The notion that morpheme based alignment would be useful in highly inflected languages is intuitive. Morphological inflections might indicate tense, gender or number that manifest as separate words in largely uninflected languages. Capturing these subword alignments can yield better word alignments that otherwise would be missed.

Let us make this idea concrete with a case study of the benefits of morpheme based alignment. We show the intersecting alignments of an actual English (source) → Finnish (target) sentence pair in Figure 1, where (a) word-level and (b) morpheme-level alignments are shown. The morpheme-level alignment is produced by automatically segmenting words into morphemes and running IBM Model 4 on the resulting token stream.

Intersection links (*i.e.*, common to both direct and inverse alignments) play an important role in creating the final alignment (Och and Ney, 2004). While there are several heuristics used in the symmetrizing process, the *grow-diag(onal)* process is common and prevalent in many SMT systems, such as Moses (Koehn et al., 2007). In the grow-diag process, intersection links are used as seeds to find other new alignments within their neighborhood. The process continues iteratively, until no further links can be added.

In our example, the morpheme-level intersection alignment is better as it has no misalignments and adds new alignments. However it misses some key links. In particular, the alignments of closed-class morphemes (CCMs; later formally defined) as indicated by the dotted lines in (b) are overlooked in the IBM Model 4 alignment. This difficulty in aligning CCMs is due to:

1. Occurrences of *garbage-collector* words (Moore, 2004) that attract CCMs to align to them. Examples of such links in (b) are 1–23 or 11–21 with the occurrences of rare words $\texttt{adjourn+}_{11}$ and $\texttt{avatuksi}_{23}$. We further characterize such errors in Section 6.1.

2. Ambiguity among CCMs of the same surface that causes incorrect matchings. In (b), we observe multiple occurrence of `the` and `n` on the source and target sides respectively. While the link 8–6 is correct, 5–4 is not as $\texttt{i}_1$ should be aligned to $\texttt{n}_4$ instead. To resolve such ambiguity, context information should be considered as detailed in Section 4.3.

The fact that rare words and multiple affixes often occur in highly inflected languages exacerbates this problem, motivating our focus on improving CCM alignment. Furthermore, having access to the correct CCM alignments as illustrated in Figure 1 guides the grow-diag process in finding the remaining correct alignments. For example, the addition of CCM links $\texttt{i}_1$–$\texttt{n}_4$ and $\texttt{d}_4$–$\texttt{lle}_{21}$ helps to identify $\texttt{declare}_2$–$\texttt{julist}_2$ and $\texttt{resume}_3$–$\texttt{avatuksi}_{23}$ as admissible alignments, which would otherwise be missed.

4 Methodology

Our idea is to enrich the standard IBM Model 4 alignment by modeling closed-class morphemes (CCMs) more carefully using global statistics and context. We realize our idea by proposing a four-step method. First, we take the input parallel corpus and convert it into morphemes before training the IBM Model 4 morpheme alignment. Second, from the morpheme alignment, we induce automatically bilingual CCM pairs. The core of our approach is in the third and fourth steps. In Step 3, we construct a *CCM alignment model*, and apply it on the segmented input corpus to obtain an automatic CCM alignment. Finally, in Step 4, we incorporate the CCM alignment into the symmetrizing process via our *modified grow-diag process*.

4.1 Step 1: Morphological Analysis

The first step presupposes morphologically segmented input to compute the IBM Model 4 morpheme alignment. Following Virpioja et al. (2007), we use *Morfessor*, an unsupervised analyzer which learns morphological segmentation from raw tokenized text (Creutz and Lagus, 2007).

The tool segments input words into labeled morphemes: `PRE` (prefix), `STM` (stem), and `SUF` (suffix). Multiple affixes can be proposed for each word; word compounding is allowed as well, e.g., `uncarefully` is analyzed as `un/PRE+ care/STM+ ful/SUF+ ly/SUF`. We append a "+" sign to each nonfinal tag to distinguish word-internal morphemes from word-final ones, e.g., "x/STM" and "x/STM+" are considered different tokens. The "+" annotation enables the restoration of the original words, a key point to enforce word boundary constraints in our work later.

4.2 Step 2: Bilingual CCM Pairs

We observe that low and highly inflected languages, while intrinsically different, share more in common at the morpheme level. The many-to-one relationships among words on both sides is often captured better by one-to-one correspondences among morphemes. We wish to model such bilingual correspondence in terms of closed-class morphemes (CCM), similar to Nguyen and Vogel (2008)'s work that removes nonaligned affixes during the alignment process. Let us now formally define CCM and an associative measure to gauge such correspondence.

en	**fi**	**en**	**fi**	**en**	**fi**
the$_1$	-n$^{\dagger}_{1}$	in$_6$	-ssa$^{\ddagger}_{15}$	me$_{166}$	-ni$^{\ddagger}_{60}$
-s$_2$	-t$^{\ddagger}_{9}$	is$_7$	on$^{\ddagger}_{2}$	me$_{166}$	minun$^{\dagger}_{282}$
to$_3$	-ä$_6$	that$_8$	että$^{\ddagger}_{7}$	why$_{168}$	siksi$^{\ddagger}_{187}$
to$_3$	maan$_{91}$	that$_8$	ettei$^{\ddagger}_{283}$	view$_{172}$	mieltä$^{\dagger}_{162}$
of$_4$	-a$_4$	we$_{10}$	-mme$^{\ddagger}_{10}$	still$_{181}$	vielä$^{\ddagger}_{108}$
of$_4$	-en$^{\dagger}_{5}$	we$_{10}$	meidän$^{\dagger}_{52}$	where$_{183}$	jossa$^{\ddagger}_{209}$
of$_4$	-sta$^{\dagger}_{19}$	we$_{10}$	me$^{\ddagger}_{113}$	same$_{186}$	samaa$^{\ddagger}_{334}$
and$_5$	ja$^{\ddagger}_{3}$	we$_{10}$	emme$_{123}$	he$_{187}$	hän$^{\ddagger}_{184}$
and$_5$	sekä$^{\ddagger}_{122}$	we$_{10}$	meillä$^{\dagger}_{231}$	good$_{189}$	hyvä$^{\ddagger}_{321}$
and$_5$	eikä$_{203}$	...	...	over-$_{408}$	yli-$^{\ddagger}_{391}$

Table 1: **English(en)-Finnish(fi) Bilingual CCM pairs** (N=128). Shown are the top 19 and last 10 of 168 bilingual CCM pairs extracted. Subscript i indicates the i^{th} most frequent morpheme in each language. ‡ marks exact correspondence linguistically, whereas † suggests rough correspondence w.r.t `http://en.wiktionary.org/wiki/`.

Definition 1. *Closed-class Morphemes (CCM)* are a *fixed set of stems and affixes* that exhibit grammatical functions just like closed-class words. In highly inflected languages, we observe that grammatical meanings may be encoded in morphological stems and affixes, rather than separate words. While we cannot formally identify valid CCMs in a language-independent way (as by definition they manifest language-dependent grammatical functions), we can devise a good approximation. Following Setiawan et al. (2007), we induce the set of CCMs for a language as the top N frequent stems together with all affixes[1].

Definition 2. *Bilingual Normalized PMI (biPMI)* is the averaged normalized PMI computed on the asymmetric morpheme alignments. Here, normalized PMI (Bouma, 2009), known to be less biased towards low-frequency data, is defined as: $\mathit{nPMI}(x, y) = \ln \frac{p(x,y)}{p(x)p(y)})/\text{-}\ln p(x, y)$, where $p(x)$, $p(y)$, and $p(x, y)$ follow definitions in the standard PMI formula. In our case, we only

[1] Note that we employ length and vowel sequence heuristics to filter out corpus-specific morphemes.

compute the scores for x, y being morphemes frequently aligned in both asymmetric alignments.

Given these definitions, we now consider a pair of source and target CCMs related and termed a **bilingual CCM pair** (CCM pair, for short) if they exhibit positive correlation in their occurrences (*i.e.*, positive nPMI[2] and frequent cooccurrences).

We should note that relying on a hard threshold of N as in (Setiawan et al., 2007) is brittle as the CCM set varies in sizes across languages. Our method is superior in the use of N as a starting point only; the bilingual correspondence of the two languages will ascertain the final CCM sets.

Take for example the *en* and *fi* CCM sets with 154 and 214 morphemes initially (each consisting of N=128 stems). As morphemes not having their counterparts in the other language are spurious, we remove them by retaining only those in the CCM pairs. This effectively reduces the respective sizes to 91 and 114. At the same time, these final CCMs cover a much larger range of top frequent morphemes than N, up to 408 *en* and 391 *fi* morphemes, as evidenced in Table 1.

4.3 Step 3: The CCM Alignment Model

The goal of this model is to predict when appearances of a CCM pair should be deemed as linking.

With an identified set of CCM pairs, we know when source and target morphemes correspond. However, in a sentence pair there can be many instances of both the source and target morphemes. In our example, the `the`–`n` pair corresponds to definite nouns; there are two `the` and three -`n` instances, yielding 2 × 3=6 possible links.

Deciding which instances are aligned is a decision problem. To solve this, we inspect the IBM Model 4 morpheme alignment to construct a CCM alignment model. The CCM model labels whether an instance of a CCM pair is deemed semantically related (linked). We cast the modeling problem as supervised learning, where we choose a maximum entropy (ME) formulation (Berger et al., 1996).

We first discuss sample selection from the IBM Model 4 morpheme alignment, and then give details on the features extracted. The processes described below are done per sentence pair with f_1^m, e_1^n and U denoting the source, target sentences and the union alignments, respectively.

Class labels. We base this on the initial IBM Model 4 alignment to label each CCM pair instance as a positive or negative example: *Positive examples* are simply CCM pairs in U. To be precise, links j–i in U are positive examples if f_j–e_i is a CCM pair. To find *negative examples*, we inventory other potential links that share the same lexical items with a positive one. That is, a link j'–i' not in U is a negative example, if a positive link j–i such that $f_j = f'_j$ and $e_i = e'_i$ exists.

We stress that our collection of positive examples contains high-precision but low-recall IBM Model 4 links, which connect the reliable CCM pairs identified before. The model then generalizes from these samples to detect incorrect CCM links and to recover the correct ones, enhancing recall. We later detail this process in §4.4.

Feature Set. Given a CCM pair instance, we construct three feature types: lexical, monolingual, and bilingual (See Table 2). These features capture the global statistics and contexts of CCM pairs to decide if they are true alignment links.

• **Lexical features** reflect the tendency of the CCM pair being aligned to themselves. We use *biPMI*, which aggregates the global alignment statistics, to determine how likely source and target CCMs are associated with each other.

• **Monolingual context features** measure the association among tokens of the same language, capturing what other stems and affixes co-occur with the source/target CCM:

1. within the same word (*intra*). The aim is to disambiguate affixes as necessary in highly inflected languages where same stems could generate different roles or meanings.
2. outside the CCM's word boundary (*inter*). This potentially capture cues such as tense, or number agreement. For example, in English, the 3sg agreement marker on verbs `-s` often co-occurs with nearby pronouns e.g., `he`, `she`, `it`; whereas the same marker on nouns (`-s`), often appears with plural determiners e.g., `these`, `those`, `many`.

To accomplish this, we compute two monolingual nPMI scores in the same spirit as biPMI, but using the morphologically segmented input from

[2] *nPMI* has a bounded range of $[-1, 1]$ with values 1 and 0 indicating perfect positive and no correlation, respectively.

Feature Description	Examples
Lexical — *biPMI*: None $[-1, 0]$, Low $(0, 1/3]$, Medium $(1/3, 2/3]$, High $(2/3, 1]$	pmi$_{d-lle}$=Low
Monolingual Context — Capture morpheme cooccurrence with the src/tgt CCM	
Intra – **W**ithin the same word	srcW$_{d-lle}$=resume, tgtW$_{d-lle}$=en, tgtW$_{d-lle}$=uude
Inter – To the **L**eft & **R**ight, in different words	srcL$_{d-lle}$=i, srcR$_{d-lle}$=the, tgtR$_{d-lle}$=avatuksi
Bilingual context — Capture neighbor links' cooccurrence with the CCM pair link	
bi0 – Most descriptive, capturing in terms of surface forms only → maybe sparse	bi0$_{d-lle}$=resume–avatuksi
bi1 – Generalizes morphemes into relative locations (**L**eft, **W**ithin, **R**ight)	bi1$_{d-lle}$=W–avatuksi, bi1$_{d-lle}$=resume–R
bi2 – Most general, coupling token types (**C**lose, **O**pen) /w relative positions	bi2$_{d-lle}$=O–WR

Table 2: **Maximum entropy feature set.** Shown are feature types, descriptions and examples. Most examples are given for the alignment d$_4$–lle+$_{21}$ of the same running example in §3. Note that we only partially list the bilingual context features.

each language separately. Two morphemes are "linked" if within a context window of w_c words.

- **Bilingual context features** model cross-lingual reordering, capturing the relationships between the CCM pair link and its neighbor[3] links. Consider a simple translation between an English phrase of the form we ⟨verb⟩ and the Finnish one ⟨verb⟩ -mme, where -mme is the 1pl verb marker. We aim to capture movements such as "the open-class morphemes on the right of we and on the left of -mme are often aligned". These will function as evidence for the ME learner to align the CCM pair (we, -mme). We encode the bilingual context at three different granularities, from most specific to most general ones (cf Table 2).

4.4 Step 4: Incorporate CCM Alignment

At test time, we apply the trained CCM alignment model to all CCM pairs occurring in each sentence pair to find CCM links. On our running example in Figure 1, the CCM classifier tests 17 CCM pairs, identifying 6 positive CCM links of which 4 are true positives (dotted lines in (b)).

Though mostly correct, we note that some of the predicted links conflict: (d$_4$–lle$_{21}$, ed$_{12}$–neen$_{17}$ and ed$_{12}$–lle$_{21}$) share alignment endpoints. Such sharing in CCM alignments is rare and we believe should be disallowed. This motivates us to resolve all CCM link conflicts before incorporating them into the symmetrizing process.

Resolving link conflicts. As CCM pairs are classified independently, they possess classification probabilities which we use as evidence to resolve the conflicts. In our example, the classification probabilities for (d$_4$–lle$_{21}$, ed$_{12}$–neen$_{17}$, ed$_{12}$–lle$_{21}$) are $(0.99, 0.93, 0.79)$ respectively.

We use a simple, "best-first" greedy approach to determine which links are kept and which are dropped to satisfy our assumption. In our case, we pick the most confident link, d$_4$–lle$_{21}$ with probability 0.99. This precludes the incorrect link, ed$_{12}$–lle$_{21}$, but admits the other correct one ed$_{12}$–neen$_{17}$, probability 0.93. As a result, this resolution successfully removes the incorrect link.

Modifying grow-diag. We incorporate the set of conflict-resolved CCM links into the grow-diag process. This step modifies the input alignments as well as the growing process. U and I denote the IBM Model 4 union and intersection alignments.

In our view, the resolved CCM links can serve as a quality mark to "upgrade" links before input into the grow-diag process. We upgrade resolved CCM links: (a) those $\in U \rightarrow$ part of I, treating them as alignment seeds; (b) those $\notin U \rightarrow$ part of U, using them for exploration and growing. To reduce spurious alignments, we discarded links in U that conflict with the resolved CCM links.

In the usual grow-diag, links immediately adjacent to a seed link l are considered candidates to be appended into the alignment seeds. While suitable for word-based alignment, we believe it is too small a context when the input are morphemes.

For morpheme alignment, the candidate context makes more sense in terms of word units. We thus *enforce word boundaries* in our modified grow-diag. We derive word boundaries for end points in l using the morphological tags and the "+" word-end marker mentioned in §4.1. Using such boundaries, we can then extend the grow-diag to consider candidate links within a neighborhood of w_g words; hence, enhancing the candidate coverage.

5 Experiments

We use English-Finnish and English-Hungarian data from past shared tasks (WPT05 and WMT09)

[3] Within a context window of w_c words as in monolingual.

to validate our approach. Both Finnish and Hungarian are highly inflected languages, with numerous verbal and nominal cases, exhibiting agreement. Dataset statistics are given in Table 3.

	en-fi	#	en-hu	#
Train	Europarl-v1	714K	Europarl-v4	1,510K
LM	Europarl-v1-fi	714K	News-hu	4,209K
Dev	wpt05-dev	2000	news-dev2009	2051
Test	wpt05-test	2000	news-test2009	3027

Table 3: **Dataset Statistics:** the numbers of parallel sentences for training, LM training, development and test sets.

We use the Moses SMT framework for our work, creating both our CCM-based systems and the baselines. In all systems built, we obtain the IBM Model 4 alignment via GIZA++ (Och and Ney, 2003). Results are reported using case-insensitive BLEU (Papineni et al., 2001).

Baselines. We build two SMT baselines:

w-system: This is a standard phrase-based SMT, which operates at the word level. The system extracts phrases of maximum length 7 words, and uses a 4-gram word-based LM.

w_m-system: This baseline works at the word level just like the w-system, but differs at the alignment stage. Specifically, input to the IBM Model 4 training is the morpheme-level corpus, segmented by *Morfessor* and augmented with "+" to provide word-boundary information (§4.1). Using such information, we constrain the alignment symmetrization to extract phrase pairs of 7 words or less in length. The morpheme-based phrase table is then mapped back to word forms. The process continues identically as in the w-system.

CCM-based systems. Our CCM-based systems are similar in spirit to the w_m system: train at the morpheme, but decode at the word level. We further enhance the w_m-system at the alignment stage. First, we train our CCM model based on the initial IBM Model 4 morpheme alignment, and apply it to the morpheme corpus to obtain CCM alignment, which are input to our modified grow-diag process. The CCM approach defines the setting of three parameters: $\langle N, w_c, w_g\rangle$ (Section 4). Due to our resource constraints, we set N=128, similar to (Setiawan et al., 2007), and w_c=1 experimentally. We only focus on the choice of w_g, testing w_g=$\{1, 2\}$ to explore the effect of enforcing word boundaries in the grow-diag process.

5.1 English-Finnish results

We test the translation quality of both directions (*en-fi*) and (*fi-en*). We present results in Table 4 for 7 systems, including: our baselines, three CCM-based systems with word-boundary knowledge w_g=$\{0, 1, 2\}$ and two w_m-systems w_g=$\{1, 2\}$.

Results in Table 4 show that our CCM approach effectively improves the performance. Compared to the w_m-system, it chalks up a gain of 0.46 BLEU points for *en-fi*, and a larger improvement of 0.93 points for the easier, reverse direction.

Further using word boundary knowledge in our modified grow-diag process demonstrates that the additional flexibility consistently enhances BLEU for $w_g = 1, 2$. We achieve the best performance at $w_g = 2$ with improvements of 0.67 and 1.22 BLEU points for *en-fi* and *fi-en*, respectively.

	en-fi	fi-en
w-system	14.58	23.56
w_m-system	14.47	22.89
w_m-system + CCM	$14.93_{+0.46}$	$23.82_{+0.93}$
w_m-system + CCM + $w_g = 1$	15.01	23.95
w_m-system + CCM + $w_g = 2$	$\mathbf{15.14}_{+0.67}$	$\mathbf{24.11}_{+1.22}$
w_m-system + $w_g = 1$	14.44	22.92
w_m-system + $w_g = 2$	14.28	23.01
(Ganchev, 2008) - Base	14.72	22.78
(Ganchev, 2008) - Postcat	14.74	$23.43_{+0.65}$
(Yang, 2006) - Base	N/A	22.0
(Yang, 2006) - Backoff	N/A	$22.3_{+0.3}$

Table 4: **English/Finnish results.** Shown are BLEU scores (in %) with subscripts indicating absolute improvements with respect to the w_m-system baseline.

Interestingly, employing the word boundary heuristic alone in the original grow-diag does not yield any improvement for *en-fi*, and even worsens as w_g is enlarged (as seen in Rows 6–7). There are only slight improvements for *fi-en* with larger w_g.This attests to the importance of combining the CCM model and the modified grow-diag process.

Our best system outperforms the w-system baseline by 0.56 BLEU points for *en-fi*, and yields an improvement of 0.55 points for *fi-en*.

Compared to works experimenting *en/fi* translation, we note the two prominent ones by Yang and Kirchhoff (2006) and recently by Ganchev et al. (2008). The former uses a simple back-off method experimenting only *fi-en*, yielding an improvement of 0.3 BLEU points. Work in the opposite direction (*en-fi*) is rare, with the latter paper extending the EM algorithm using posterior constraints, but showing no improvement; for *fi-en*, they demonstrate a gain of 0.65 points. Our CCM method compares favorably against both approaches, which use the same datasets as ours.

5.2 English-Hungarian results

To validate our CCM method as language-independent, we also perform preliminary experiments on *en-hu*. Table 5 shows the results using the same CCM setting and experimental schemes as in *en/fi*. An improvement of 0.35 BLEU points is shown using the CCM model. We further improve by 0.44 points with word boundary w_g=1, but performance degrades for the larger window. Due to time constraints, we leave experiments for the reverse, easier direction as future work. Though modest, the best improvement for *en-hu* is statistical significant at p<0.01 according to Collins' sign test (Collins et al., 2005).

System	**BLEU**
w-system	9.63
w$_m$-system	9.47
w$_m$-system + CCM	9.82 $_{+0.35}$
w$_m$-system + CCM + $w_g = 1$	**9.91** $_{+0.44}$
w$_m$-system + CCM + $w_g = 2$	9.87

Table 5: **English/Hungarian results.** Subscripts indicate absolute improvements with respect to the w$_m$-system.

We note that MT experiments for *en/hu* [4] are very limited, especially for the *en* to *hu* direction. Novák (2009) obtained an improvement of 0.22 BLEU with no distortion penalty; whereas Koehn and Haddow (2009) enhanced by 0.5 points using monotone-at-punctuation reordering, minimum Bayes risk and larger beam size decoding.

While not directly comparable in the exact settings, these systems share the same data source and splits similar to ours. In view of these community results, we conclude that our CCM model does perform competitively in the *en-hu* task, and indeed seems to be language independent.

6 Detailed Analysis

The macroscopic evaluation validates our approach as improving BLEU over both baselines, but how do the various components contribute? We first analyze the effects of Step 4 in producing the CCM alignment, and then step backward to examine the contribution of the different feature classes in Step 3 towards the ME model.

6.1 Quality of CCM alignment

To evaluate the quality of the predicted CCM alignment, we address the following questions:

Q1: What is the portion of CCM pairs being misaligned in the IBM Model 4 alignment?

Q2: How does the CCM alignment differ from the IBM Model 4 alignment?

Q3: To what extent do the new links introduced by our CCM model address Q1?

Given that we do not have linguistic expertise in Finnish or Hungarian, it is not possible to exhaustively list all misaligned CCM pairs in the IBM Model 4 alignment. As such, we need to find other form of approximation in order to address Q1.

We observe that correct links that do not exist in the original alignment could be entirely missing, or mistakenly aligned to neighboring words. With morpheme input, we can also classify mistakes with respect to intra- or inter-word errors. Figure 2 characterizes errors T_1, T_2 and T_3, each being a more severe error class than the previous. Focusing on e_i in the figure, links connecting e_i to $f_{j'}$ or $f_{j''}$ are deemed T_1 errors (misalignments happen on one side). A T_2 error aligns f''_j within the same word, while a T_3 error aligns it outside the current word but still within its neighborhood. This characterization is automatic, cheap and has the advantage of being language-independent.

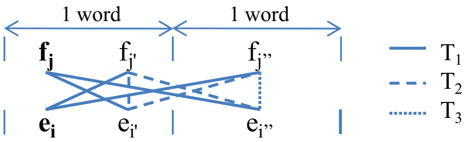


Figure 2: **Categorization of CCM missing links.** Given that a CCM pair link (f_j–e_i) is not present in the IBM Model 4, occurrences of any nearby link of the types T$_{[1-3]}$ can be construed as evidence of a potential misalignment.

Statistics in Table 6(ii) answers Q1, suggesting a fairly large number of missing CCM links: $3,418K$ for *en/fi* and $6,216K$ for *en/hu*, about 12.35% and 12.06% of the IBM Model 4 union alignment respectively. We see that T_1 errors con-

[4] Hungarian was used in the ACL shared task 2008, 2009.

stitute the majority, a reasonable reflection of the *garbage- collector*[5] effect discussed in Section 3.

	General (i)		Missing CCM links (ii)		
	en/fi	en/hu		en/fi	en/hu
Direct	17,632K	34,312K	T_1	2,215K	3,487K
Inverse	18,681K	34,676K	T_2	358K	690K
$D \cap I$	8,643K	17,441K	T_3	845K	2,039K
$D \cup I$	27,670K	51,547K	**Total**	**3,418K**	**6,216K**

Table 6: **IBM Model 4 alignment statistics.** (i) General statistics. (ii) Potentially missing CCM links.

Q2 is addressed by the last column in Table 7. Our CCM model produces about 11.98% (1,035K/8,643K) new CCM links as compared to the size of the IBM Model 4 intersection alignment for en/fi, and similarly, 9.52% for en/hu.

	Orig.	Resolved	I	U\I	New
en/fi	5,299K	3,433K	1065K	1,332K	**1,035K**
en/hu	9,425K	6,558K	2,752K	2,146K	**1,660K**

Table 7: **CCM vs IBM Model 4 alignments.** *Orig.* and *Resolved* give # CCM links predicted in Step 4 before and after resolving conflicts. Also shown are the number of resolved links present in the **I**ntersection, Union excluding I (**U****I**) of the IBM Model 4 alignment and **New** CCM links.

Lastly, figures in Table 8 answer Q3, revealing that for *en/fi*, 91.11% (943K/1,035K) of the new CCM links effectively cover the missing CCM alignments, recovering 27.59% (943K/3,418K) of all missing CCM links. Our modified *grow-diag* realizes a majority 76.56% (722K/943K) of these links in the final alignment.

We obtain similar results in the *en/hu* pair for link recovery, but a smaller percentage 22.59% (330K/1,461K) are realized through the modified symmetrization. This partially explains why improvements are modest for *en/hu.*

	New CCM Links (i)		Modified grow-diag (ii)	
	en/fi	en/hu	en/fi	en/hu
T_1	707K	1,002K	547K	228K
T_2	108K	146K	79K	22K
T_3	128K	313K	96K	80K
Total	**943K**	**1,461K**	**722K**	**330K**

Table 8: **Quality of the newly introduced CCM links**. Shown are # new CCM links addressing the three error types before (i) and after (ii) the modified grow-diag process.

6.2 Contributions of ME Feature Classes

We also evaluate the effectiveness the ME features individually through ablation tests. For brevity, we only examine the more difficult translation direction, *en* to *fi*. Results in Table 9 suggest that all our features are effective, and that removing any feature class degrades performance. Balancing specificity and generality, `bi1` is the most influential feature in the bilingual context group. For monolingual context, `inter`, which captures larger monolingual context, outperforms `intra`. The most important feature overall is `pmi`, which captures global alignment preferences. As feature groups, bilingual and monolingual context features are important sources of information, as removing them drastically decreases system performance by 0.23 and 0.16 BLEU, respectively.

System			BLEU
all (w_m-system+CCM)			14.93
−bi2	14.90	−intra	14.89
−bi1	$14.84^{*}_{-0.09}$	−pmi	$14.81^{*}_{-0.12}$
−bi0	14.89	−bi{2/1/0}	$14.70^{*}_{-0.23}$
−inter	14.85	−in{ter/tra}	$14.77^{*}_{-0.16}$

Table 9: **ME feature ablation tests for English-Finnish experiments.** $*$ mark results statistically significant at $p < 0.05$, differences are subscripted.

7 Conclusion and Future Work

In this work, we have proposed a language-independent model that addresses morpheme alignment problems involving highly inflected languages. Our method is unsupervised, requiring no language specific information or resources, yet its improvement on BLEU is comparable to much semantically richer, language-specific work. As our approach deals only with input word alignment, any downstream modifications of the translation model also benefit.

As alignment is a central focus in this work, we plan to extend our work over different and multiple input alignments. We also feel that better methods for the incorporation of CCM alignments is an area for improvement. In the en/hu pair, a large proportion of discovered CCM links are discarded, in favor of spurious links from the union alignment. Automatic estimation of the correctness of our CCM alignments may improve end translation quality over our heuristic method.

[5] E.g., e_i prefers f'_j or f''_j (garbage collectors) over f_j.

References

Berger, Adam L., Stephen D. Della Pietra, and Vincent J. D. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Bouma, Gerlof. 2009. Normalized (pointwise) mutual information in collocation extraction. In *Proceedings of the Biennial GSCL Conference*, Tübingen, Gunter Narr Verlag.

Cherry, Colin and Dekang Lin. 2007. Inversion transduction grammar for joint phrasal translation modeling. In *SSST*.

Collins, Michael, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *ACL*.

Creutz, Mathias and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1):3.

DeNero, John and Dan Klein. 2007. Tailoring word alignments to syntactic machine translation. In *ACL*.

Ganchev, Kuzman, João V. Graça, and Ben Taskar. 2008. Better alignments = better translations? In *ACL-HLT*.

Goldwater, Sharon and David McClosky. 2005. Improving statistical mt through morphological analysis. In *HLT*.

Haghighi, Aria, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised itg models. In *ACL*.

Koehn, Philipp and Barry Haddow. 2009. Edinburgh's submission to all tracks of the WMT2009 shared task with reordering and speed improvements to Moses. In *EACL*.

Koehn, Philipp, Hieu Hoang, Alexandra Birch Mayne, Christopher Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL, Demonstration Session*.

Lee, Young-Suk. 2004. Morphological analysis for statistical machine translation. In *HLT-NAACL*.

Liang, Percy, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *HLT-NAACL*.

Matusov, Evgeny, Richard Zens, and Hermann Ney. 2004. Symmetric word alignments for statistical machine translation. In *COLING*.

Moore, Robert C. 2004. Improving IBM word-alignment model 1. In *ACL*.

Nguyen, Thuy Linh and Stephan Vogel. 2008. Context-based Arabic morphological analysis for machine translation. In *CoNLL*.

Novák, Attila. 2009. MorphoLogic's submission for the WMT 2009 shared task. In *EACL*.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, Franz Josef and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Morristown, NJ, USA. Association for Computational Linguistics.

Setiawan, Hendra, Min-Yen Kan, and Haizhou Li. 2007. Ordering phrases with function words. In *ACL*.

Virpioja, Sami, Jaakko J. Vyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology-aware statistical machine translation based on morphs induced in an unsupervised manner. In *MT Summit XI*.

Yang, Mei and Katrin Kirchhoff. 2006. Phrase-based backoff models for machine translation of highly inflected languages. In *EACL*.

Zhang, Hao and Daniel Gildea. 2005. Stochastic lexicalized inversion transduction grammar for alignment. In *ACL*.

Zhang, Hao, Chris Quirk, Robert C. Moore, and Daniel Gildea. 2008. Bayesian learning of non-compositional phrases with synchronous parsing. In *ACL-HLT*.

Automatic analysis of semantic similarity in comparable text through syntactic tree matching

Erwin Marsi
TiCC, Tilburg University
e.c.marsi@uvt.nl

Emiel Krahmer
TiCC, Tilburg University
e.j .krahmer@uvt.nl

Abstract

We propose to analyse semantic similarity in comparable text by matching syntactic trees and labeling the alignments according to one of five semantic similarity relations. We present a Memory-based Graph Matcher (MBGM) that performs both tasks simultaneously as a combination of exhaustive pairwise classification using a memory-based learner, followed by global optimization of the alignments using a combinatorial optimization algorithm. The method is evaluated on a monolingual treebank consisting of comparable Dutch news texts. Results show that it performs substantially above the baseline and close to the human reference.

1 Introduction

Natural languages allow us to express essentially the same underlying meaning as many alternative surface forms. In other words, there are often many similar ways to say the same thing. This characteristic poses a problem for many natural language processing applications. Automatic summarizers, for example, typically rank sentences according to their informativity and then extract the top n sentences, depending on the required compression rate. Although the sentences are essentially treated as independent of each other, they typically are not. Extracted sentences may have substantial semantic overlap, resulting in unintended redundancy in the summaries. This is particularly problematic in the case of multi-document summarization, where sentences extracted from related documents are very likely to express similar information in different ways (Radev and McKeown, 1998). Therefore, if semantic similarity between sentences could be detected automatically, this would certainly help to avoid redundancy in summaries.

Similar arguments can be made for many other NLP applications. Automatic duplicate and plagiarism detection beyond obvious string overlap requires recognition of semantic similarity. Automatic question-answering systems may benefit from clustering semantically similar candidate answers. Intelligent document merging software, which supports a minimal but lossless merge of several revisions of the same text, must handle cases of paraphrasing, restructuring, compression, etc. Recognizing textual entailments (Dagan et al., 2005) could arguably be seen as a specific instance of detecting semantic similarity.

In addition to merely *detecting* semantic similarity, we can ask to what extent two expressions share meaning. For instance, the meaning of one sentence can be fully contained in that of another, the meaning of one sentence can overlap only partly with that of another, etc. This requires an *analysis* of the semantic similarity between a pair of expressions. Like detection, automatic analysis of semantic similarity can play an important role in NLP applications. To return to the case of multi-document summarization, analysing the semantic similarity between sentences extracted from different documents provides the basis for *sentence fusion*, a process where a new sentence is generated that conveys all common information from both sentences without introducing redundancy (Barzilay and McKeown, 2005; Marsi and Krahmer, 2005b).

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 752–760,
Beijing, August 2010

Analysis of semantic similarity can be approached from different angles. A basic approach is to use string similarity measures such as the Levenshtein distance or the Jaccard similarity coefficient. Although cheap and fast, this fails to account for less obvious cases such as synonyms or syntactic paraphrasing. At the other extreme, we can perform a deep semantic analysis of two expressions and rely on formal reasoning to derive a logical relation between them. This approach suffers from issues with coverage and robustness commonly associated with deep linguistic processing. We therefore think that the middle ground between these two extremes offers the best option. In this paper we present a new method for analysing semantic similarity in comparable text. It relies on a combination of morphological and syntactic analysis, lexical resources such as word nets, and machine learning from examples. We propose to analyse semantic similarity between sentences by aligning their syntax trees, where each node is matched to the most similar node in the other tree (if any). In addition, we label these alignments according to the type of similarity relation that holds between the aligned phrases. The labeling supports further processing. For instance, Marsi & Krahmer (2005b; 2008) describe how to generate different types of sentence fusions on the basis of this relation labeling.

In the next Section we provide a more formal definition of the task of matching syntactic trees and labeling alignments, followed by a discusion of related work in Section 3. Section 4 describes a parallel, monolingual treebank used for developing and testing our approach. In Section 5 we propose a new algorithm for simultaneous node alignment and relation labeling. The results of several evaluation experiments are presented in Section 6. We finish with a conclusion.

2 Problem statement

Aligning a pair of similar syntactic trees is the process of pairing those nodes that are most similar. More formally: let v be a node in the syntactic tree T of sentence S and v' a node in the syntactic tree T' of sentence S'. A *labeled node alignment* is a tuple $< v, v', r >$ where r is a label from a set of relations. A *labeled tree alignment* is a set of labeled node alignments. A *labeled tree matching* is a tree alignment in which each node is aligned to at most one other node.

For each node v, its terminal *yield* $\textsc{str}(v)$ is defined as the sequence of all terminal nodes reachable from v (i.e., a substring of sentence S). Aligning node v to v' with label r indicates that relation r holds between their yields $\textsc{str}(v)$ and $\textsc{str}(v')$. We label alignments according to a small set of *semantic similarity relations*. As an example, consider the following Dutch sentences:

(1) a. Dagelijks koffie vermindert risico op
Daily coffee diminishes risk on
Alzheimer en Dementie.
Alzheimer and Dementia.

b. Drie koppen koffie per dag reduceert
Three cups coffee a day reduces
kans op Parkinson en Dementie.
chance on Parkinson and Dementia.

The corresponding syntax trees and their (partial) alignment is shown in Figure 1. We distinguish the following five mutually exclusive similarity relations:

1. v **equals** v' iff lower-cased $\textsc{str}(v)$ and lower-cased $\textsc{str}(v')$ are identical – example: *Dementia* equals *Dementia*;
2. v **restates** v' iff $\textsc{str}(v)$ is a proper paraphrase of $\textsc{str}(v')$ – example: *diminishes* restates *reduces*;
3. v **generalizes** v' iff $\textsc{str}(v)$ is more general than $\textsc{str}(v')$ – example: *daily coffee* generalizes *three cups of coffee a day*;
4. v **specifies** v' iff $\textsc{str}(v)$ is more specific than $\textsc{str}(v')$ – example: *three cups of coffee a day* specifies *dailly coffee*;
5. v **intersects** v' iff $\textsc{str}(v)$ and $\textsc{str}(v')$ share meaning, but each also contains unique information not expressed in the other – example: *Alzheimer and Dementia* intersects *Parkinson and Dementia*.

Our interpretation of these relations is one of common sense rather than strict logic, akin to the definition of entailment employed in the RTE challenge (Dagan et al., 2005). Note also that relations are prioritized: *equals* takes precedence

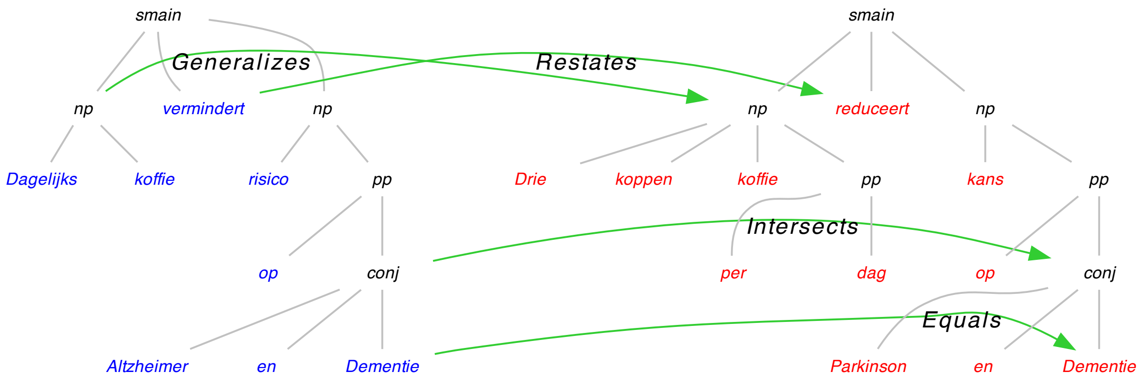


Figure 1: Example of two aligned and labeled syntactic trees. For expository reasons the alignment is not exhaustive.

over *restates*, etc. Furthermore, *equals*, *restates* and *intersects* are symmetrical, whereas *generalizes* is the inverse of *specifies*. Finally, nodes containing unique information, such as *Alzheimer* and *Parkinson*, remain unaligned.

3 Related work

Many syntax-based approaches to machine translation rely on bilingual treebanks to extract transfer rules or train statistical translation models. In order to build bilingual treebanks a number of methods for automatic tree alignment have been developed, e.g., (Gildea, 2003; Groves et al., 2004; Tinsley et al., 2007; Lavie et al., 2008). Most related to our approach is the work on discriminative tree alignment by Tiedemann & Kotzé (2009). However, these algorithms assume that source and target sentences express the same information (i.e. *parallel* text) and cannot cope with comparable text where parts may remain unaligned. See (MacCartney et al., 2008) for further arguments and empirical evidence that MT alignment algorithms are not suitable for aligning parallel monolingual text.

MacCartney, Galley, and Manning (2008) describe a system for monolingual phrase alignment based on supervised learning which also exploits external resources for knowledge of semantic relatedness. In contrast to our work, they do not use syntactic trees or similarity relation labels. Partly similar semantic relations are used in (MacCartney and Manning, 2008) for modeling semantic containment and exclusion in natural language inference. Marsi & Krahmer (2005a) is closely related to our work, but follows a more complicated method: first a dynamic programming-based tree alignment algorithm is applied, followed by a classification of similarity relations using a supervised-classifier. Other differences are that their data set is much smaller and consists of parallel rather than comparable text. A major drawback of this algorithmic approach it that it cannot cope with crossing alignments. We are not aware of other work that combines alignment with semantic relation labeling, or algorithms which perform both tasks simultaneously.

4 Data collection

For developing our alignment algorithm we use the DAESO corpus[1]. This is a Dutch parallel monolingual treebank of 1 million words, half of which were manually annotated. The corpus consists of pairs of sentences with different levels of semantic overlap, ranging from high (different Dutch translations of books from Darwin, Montaigne and Saint-Exupéry) to low (different press releases from the two main news agencies in The Netherlands, ANP and NOVUM). For this paper, we concentrate on the latter part of the DAESO corpus, where the proportion of Equals and Restates is relatively low. This corpus segment consists of 8,248 pairs of sentences, containing 162,361 tokens (ignoring punctuation). All sentences were tokenized and tagged, and subsequently parsed by the Alpino dependency parser for Dutch (Bouma et al., 2001). Two annota-

[1] `http://daeso.uvt.nl`

		Alignment:	Labeling:						
			Eq:	Re:	Spec:	Gen:	Int:	Macro:	Micro:
Words:	F:	95.38	95.48	58.50	65.81	65.00	25.85	62.11	88.72
	SD:	2.16	2.69	7.63	13.05	11.25	18.74		
Full trees:	F:	88.31	95.83	71.38	60.21	66.71	62.67	71.36	81.92
	SD:	1.15	2.27	3.77	7.63	8.17	6.14		

Table 1: Average F-scores (in percentages, with Standard Deviations) for the six human annotators on alignment and semantic relation labeling, for words and for full syntactic trees.

tors determined which sentences in the comparable news reports contained semantic overlap. Six other annotators produced manual alignments of words and phrases in matched sentence pairs, which resulted in 86,227 aligned pairs of nodes.

A small sample of 10 similar press releases comprising a total of 48 sentence pairs was independently annotated by all six annotators to determine inter-annotator agreement. We used precision, recall and F-score on alignment. To calculate these scores for relation labeling, we simply restrict the set of alignments to those labeled with a particular relation, ignoring all others. Likewise, we restrict these sets to terminal node alignments in order to get scores on word alignment.

Given the six annotations $\mathrm{A}_1, \ldots, \mathrm{A}_6$, we repeatedly took one as the $True$ annotation against which the five other annotations were evaluated. We then computed the average scores over these $6 * 5 = 30$ scores (note that with this procedure, precision, recall and F score end up being equal). Table 1 summarizes the results, both for word alignments and for full syntactic tree alignment. It can be seen that for alignment of words an average F-score of over 95 % was obtained, while alignment for full syntactic trees results in an F-score of 88%. For relation labeling, the scores differed per relation, as is to be expected: the average F-score for Equals was over 95% for both word and full tree alignment[2], and for the other relations average F-scores between 0.6 and 0.7 were obtained. The exception to note is Intersects on word level, which only occurred a few times according to a few of the annotators. The macro and micro (weighted) F-score averages on labeled alignment are 62.11% and 88.72% for words, and 71.36% and 81.92% for full syntactic trees.

5 Memory-based Graph Matcher

In order to automatically perform the alignment and labeling tasks described in Section 2, we cast these tasks simultaneously as a combination of exhaustive pairwise classification using a supervised machine learning algorithm, followed by global optimization of the alignments using a combinatorial optimization algorithm. Input to the tree matching algorithm is a pair of syntactic trees consisting of a source tree T_s and a target tree T_t.

Step 1: Feature extraction For each possible pairing of a source node n_s in tree T_s and a target node n_t in tree T_t, create an instance consisting of feature values extracted from the input trees. Features can represent properties of individual nodes, e.g. the category of the source node is NP, or relations between nodes, e.g. source and target node share the same part-of-speech.

Step 2: Classification A generic supervised classifier is used to predict a class label for each instance. The class is either one of the semantic similarity relations or the special class *none*, which is interpreted as *no alignment*. Our implementation employs the memory-based learner TiMBL (Daelemans et al., 2009), a freely available, efficient and enhanced implementation of k-nearest neighbour classification. The classifier is trained on instances derived according to Step 1 from a parallel treebank of aligned and labeled syntactic trees.

[2] At first sight, it may seem that labeling Equals is a trivial and deterministic task, for which the F-score should always be close to 100%. However, the same word may occur multiple times in the source or target sentences, which introduces ambiguity. This frequently occurs with function words such as determiners and prepositions. Moreover, choosing among several equivalent Equals alignments may sometimes involve a somewhat arbitrary decision. This situation arises, for instance, when a proper noun is mentioned just once in the source sentence but twice in the target sentence.

Step 3: Weighting Associate a cost with each prediction so that high costs indicate low confidence in the predicted class and vice versa. We use the normalized entropy of the class labels in the set of nearest neighbours (H) defined as

$$H = -\frac{\sum_{c \in C} p(c)\, log_2\, p(c)}{log_2 |C|} \quad (1)$$

where C is the set of class labels encountered in the set of nearest neighbours (i.e., a subset of the five relations plus *none*), and $p(c)$ is the probability of class c, which is simply the proportion of instances with class label c in the set of nearest neighbours. Intuitively this means that the cost is zero if all nearest neighbours are of the same class, whereas the cost goes to 1 if the nearest neighbours are equally distributed over all possible classes.

Step 4: Matching The classification step will usually give rise to one-to-many alignment of nodes. In order to reduce this to just one-to-one alignments, we search for a node matching which minimizes the sum of costs over all alignments. This is a well-known problem in combinatorial optimization known as the *Assignment Problem*. The equivalent in graph-theoretical terms is a *minimum weighted bipartite graph matching*. This problem can be solved in polynomial time ($O(n^3)$) using e.g., the *Hungarian algorithm* (Kuhn, 1955). The output of the algorithm is the labeled tree matching obtained by removing all node alignments labeled with the special *none* relation.

6 Experiments

6.1 Experimental setup

Word alignment and full tree alignments are conceptually different tasks, which require partly different features and may have different practical applications. These are therefore addressed in separate experiments.

Table 2 summarizes the respective sizes of development and the held-out test set in terms of number of aligned graph pairs, number of aligned node pairs and number of tokens. The percentage of aligned nodes over all graphs is calculated relative to the number of nodes over all graphs. Since alignments involving non-terminal nodes are ignored in the task of word alignment, the number of aligned node pairs and the percentage of aligned nodes is lower in the word develop and word test sets. Table 3 gives the distribution of semantic relations in the development and test set, for word and tree alignment. It can be observed that the distribution if fairly skewed with Equals being the majority class, even more so for word alignments. Another thing to notice is that Intersects are much more frequent at the level of non-terminal alignments.

Data	**Graph pairs**	**Node pairs**	**Tokens**	**Aligned nodes (%)**
word develop	2 664	13 027	45 149	15.71
word test	547	2 858	10 005	14.96
tree develop	2 664	22 741	45 149	47.20
tree test	547	4 894	10 005	47.05

Table 2: Properties of develop and test data sets

Data	**Eq**	**Re**	**Spec**	**Gen**	**Int**
word develop	84.92	6.15	2.10	1.77	5.07
word test	85.62	6.09	2.17	1.99	4.13
tree develop	56.61	6.57	7.52	6.38	22.91
tree test	58.40	7.11	7.40	6.38	20.72

Table 3: Distribution of semantic similarity relations for word alignment and for full tree alignments in both develop and test data sets

Development was carried out using 10-fold cross validation on the development data and consequently reported scores on the development data are averages over 10 folds. Only two parameters were coarsely optimized on the development set. First, the amount of downsampling of the *none* class varied between 0.1 or 0.5. Second, the parameter k of the memory-based classifier – the number of nearest neighbours taken into account during classification – ranged from 1 to 15. Optimal settings were finally applied when testing on the held-out data.

A simple greedy alignment procedure served as baseline. For word alignment, identical words are aligned as Equals and identical roots as Restates. For full tree alignment, this is extended to the level of phrases so that phrases with identical words are aligned as Equals and phrases with identical roots as Restates. The baseline does not predict Specifies, Generalizes or Intersects relations, as that would require a more involved, knowledge-based approach.

All features used are described in Table 4. The word-based features rely on pure string processing and require no linguistic preprocessing. The morphology-based features exploit the limited amount of morphological analysis provided by the Alpino parser (Bouma et al., 2001). For instance, it provides word roots and decomposes compound words. Likewise the part-of-speech-based features use the coarse-grained part-of-speech tags assigned by the Alpino parser. The lexical-semantic features rely on the Cornetto database (Vossen et al., 2008), a recent extension to the Dutch WordNet, to look-up synonym and hypernym relations among source and target lemmas. Unfortunately there is no word sense disambiguation module to identify the correct senses. In addition, a background corpus of over 500M words of (mainly) news text provides the word counts required to calculate the Lin similarity measure (Lin, 1998). The syntax-based features use the syntactic structure, which is a mix of phrase-based and dependency-based analysis. The phrasal features express similarity between the terminal yields of source and target nodes. With the exception of *same-parent-lc-phrase*, these features are only used for full tree alignment, not for word alignment.

6.2 Results on word alignment

We evaluate our alignment model in two steps: first focussing on word alignment and then on full tree alignment. Table 5 summarizes the results for MBGM on word alignment (50% downsampling and $k = 3$), which we compare statistically to the baseline performance, and informally with the human scores reported in Table 1 in Section 4 (note that the human scores are only for a subset of the data used for automatic evaluation).

The first thing to observe is that the MBGM scores on the development and tests sets are very similar throughout. For predicting word alignments, the MBGM system performs significantly better than the baseline system ($t(18) = 17.72, p < .0001$). On the test set, MBGM obtains an F-score of nearly 89%, which is almost exactly halfway between the scores of the baseline system and the human scores. In a similar vein, the performance of the MBGM system on relation labeling is considerably better than that of the baseline system. For all semantic relations, MBGM performs significantly better than the baseline ($t(18) > 9.4138, p < .0001$ for each relation, trivially so for the Specifies, Generalizes and Intersects relations, which the baseline system never predicts).

The macro scores are plain averages over the 5 scores on each relation, whereas the micro scores are weighted averages. As the Equals is the majority class and at the same time easiest to predict, the micro scores are higher. The macro scores, however, better reflect performance on the real challenge, that is, correctly predicting the relations other than Equals. The MBGM macro average is 27.37% higher than the baseline (but still some 10% below the human top line), while the micro average is 5.83% higher and only 0.75% below the human top line. Macro scores on the test set are overall lower than those on the develop set, presumably because of tuning on the development data.

6.3 Results on tree alignment

Table 6 contains the results of full tree alignment (50% downsampling and $k = 5$); here both terminal and non-terminal nodes are aligned and classified in one pass. Again scores on the development and test set are very similar, the latter being slightly better. For full tree alignment, MBGM once again performs significantly better than the baseline, $t(18) = 25.68, p < .0001$. With an F-score on the test set of 86.65, MBGM scores almost 20 percent higher than the baseline system. This F-score is less than 2% lower than the average F-score obtained by our human annotators on full tree alignment, albeit not on exactly the same sample. The picture that emerges for semantic relation labeling is closely related to the one we saw for word alignments. MBGM significantly outperforms the baseline, for each semantic relation ($t(18) > 12.6636, p < .0001$). MBGM scores a macro average F-score of 52.24% (an increase of 30.05% over the baseline) and a micro average of 80.03% (12.68% above the base score). It is inter-

Feature	Type	Description
Word		
word-subsumption	string	indicate if source word equals, has as prefix, is a prefix of, has a suffix, is a suffix of, has as infix or is an infix of target word
shared-pre-/in-/suffix-len	int	length of shared prefix/infix/suffix in characters
source/target-stop-word	bool	test if source/target word is in a stop word list of frequent function words
source/target-word-len	int	length of source/target word in characters
word-len-diff	int	word length difference in characters
source/target-word-uniq	bool	test if source/target word is unique in source/target sentence
same-words-lhs/rhs	int	no. of identical preceding/following words in source and target word contexts
Morphology		
root-subsumption	string	indicate if source root equals, has as prefix, is a prefix of, has a suffix, is a suffix of, has as infix or is an infix of target root
roots-share-pre-/in-/suffix	bool	source and target root share a prefix/infix/suffix
Part-of-speech		
source/target-pos	string	source/target part-of-speech
same-pos	bool	test if source and target have same part-of-speech
source/target-content-word	bool	test if source/target word is a content word
both-content-word	bool	test if both source and target word are content words
Lexical-semantic using Cornetto		
cornet-restates	float	1.0 if source and target words are synonyms and 0.5 if they are near-synonyms, zero otherwise
cornet-specifies	float	Lin similarity score if source word is a hyponym of target word, zero otherwise
cornet-generalizes	float	Lin similarity score if source word is a hypernym of target word, zero otherwise
cornet-intersects	float	Lin similarity score if source word share a common hypernym, zero otherwise
Syntax		
source/target-cat	string	source/target syntactic category
same-cat	bool	test if source and target have same syntactic category
source/target-parent-cat	string	source/target syntactic category of parent node
source/target-deprel	string	source/target dependency relation
same-deprel	bool	test if source and target have same dependency relation
same-dephead-root	bool	test if the dependency heads of the source and target have same root
Phrasal		
word-prec/rec	float	precision/recall on the yields of source and target nodes
same-lc-phrase	bool	test if lower-cased yields of source and target nodes are identical
same-parent-lc-phrase	bool	test if lower-cased yields of parents of source and target nodes are identical
source/target-phrase-len	int	length of source/target phrase in words
phrase-len-diff	int	phrase length difference in words

Table 4: Features (where slashes indicate multiple versions of the same feature, e.g. *source/target-pos* represents the two features *source-pos* and *target-pos*)

esting to observe that MBGM obtains *higher* F-scores on Equals and on Intersects (the two most frequent relations) than the human annotators obtained. As a result of this, the micro F-score of the automatic full tree alignment is less than 2% lower than the human reference score.

Tree alignment can also be implemented as a two-step procedure, where in the first step alignments and semantic relation classifications at the word level are produced, while in the second step these are used to predict alignments and semantic relations for non-terminals. We experimented with such a two-step procedure as well, in one version using the actual word alignments and in the other the predicted word alignments. The scores of the two-step prediction are only marginally different from those of one step prediction, both for alignment and for relation classification, giving improvements in the order of about 1% for both subtasks. As is to be expected, the scores with true word alignments are much better than those with predicted word alignments. They are interesting though, because they suggest that a fairly good full tree alignment can be automatically ob-

		Alignment:	Labeling:						
			Eq:	Re:	Spec:	Gen:	Int:	Macro:	Micro:
Develop baseline:	Prec:	80.59	81.84	46.26	0.00	0.00	0.00	25.61	80.22
	Rec:	81.58	93.10	34.71	0.00	0.00	0.00	25.56	82.20
	F:	81.08	87.11	39.66	0.00	0.00	0.00	25.35	80.70
Develop MBGM:	Prec:	91.72	94.54	61.26	74.60	67.82	45.80	68.80	90.82
	Rec:	87.82	95.91	46.19	40.87	43.22	27.27	50.61	86.96
	F:	89.73	95.02	52.67	52.81	52.80	34.19	57.50	88.85
Test baseline:	Prec:	82.45	83.83	43.12	0.00	0.00	0.00	25.39	82.17
	Rec:	82.19	93.87	27.01	0.00	0.00	0.00	24.18	82.02
	F:	82.32	88.57	33.22	0.00	0.00	0.00	24.36	82.14
Test MBGM:	Prec:	90.92	94.20	53.33	59.87	54.21	42.47	60.84	89.90
	Rec:	87.09	95.41	40.21	32.75	43.28	20.31	46.39	86.11
	F:	88.96	94.80	45.85	42.34	48.17	27.48	51.73	87.97

Table 5: Scores (in percentages) on word alignment and semantic relation labeling

		Alignment:	Labeling:						
			Eq:	Re:	Spec:	Gen:	Int:	Macro:	Micro:
Develop baseline:	Prec:	82.50	83.76	46.72	0.00	0.00	0.00	26.10	82.18
	Rec:	54.54	93.66	20.01	0.00	0.00	0.00	22.74	54.34
	F:	65.67	88.43	28.02	0.00	0.00	0.00	23.29	65.42
Develop MBGM:	Prec:	92.23	96.15	55.90	54.40	56.15	70.33	66.59	84.99
	Rec:	81.04	94.03	26.64	21.71	29.34	70.27	48.40	74.68
	F:	86.27	95.08	36.08	31.03	38.54	70.30	54.21	79.50
Test baseline:	Prec:	84.23	85.68	42.24	0.00	0.00	0.00	25.58	84.14
	Rec:	56.21	94.44	14.08	0.00	0.00	0.00	21.70	56.15
	F:	67.43	89.85	21.12	0.00	0.00	0.00	22.19	67.35
Test MBGM:	Prec:	92.27	96.67	60.25	46.92	56.85	68.64	65.87	85.23
	Rec:	81.67	94.54	27.87	19.55	30.94	71.01	48.87	75.44
	F:	86.65	95.60	38.11	27.60	40.07	69.80	54.24	80.03

Table 6: Scores (in percentages) on full tree alignment and semantic relation labeling

tained given a manually checked word alignment.

7 Conclusions

We have proposed to analyse semantic similarity between comparable sentences by aligning their syntax trees, matching each node to the most similar node in the other tree (if any). In addition, alignments are labeled with a semantic similarity relation. We have presented a Memory-based Graph Matcher (MBGM) that performs both tasks simultaneously as a combination of exhaustive pairwise classification using a memory-based learning algorithm, and global optimization of alignments using a combinatorial optimization algorithm. It relies on a combination of morphological/syntactic analysis, lexical resources such as word nets, and machine learning using a parallel monolingual treebank. Results on aligning comparable news texts from a monolingual parallel treebank for Dutch show that MBGM consistently and significantly outperforms the baseline, both for alignment and labeling. This holds both for word alignment and tree alignment.

In future research we will test MBGM on other data, as the DAESO corpus contains sub-corpora with various degrees of semantic overlap. In addition, we intend to explore alternative features from word space models. Finally, we plan to evaluate MBGM in the context of NLP applications such as multi-document summarization. This includes work on how to define similarity at the sentence level in terms of the proportion of aligned constituents. Both MBGM and the annotated data set will be publicly released.[2]

Acknowledgments

This work was conducted within the DAESO project funded by the Stevin program (De Nederlandse Taalunie).

References

Barzilay, Regina and Kathleen R. McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Bouma, Gosse, Gertjan van Noord, and Robert Malouf. 2001. Alpino: Wide-coverage computational analysis of Dutch. In Daelemans, Walter, Khalil Sima'an, Jorn Veenstra, and Jakub Zavre, editors, *Computational Linguistics in the Netherlands 2000.*, pages 45–59. Rodopi, Amsterdam, New York.

Daelemans, W., J. Zavrel, K. Van der Sloot, and A. Van den Bosch. 2009. TiMBL: Tilburg Memory Based Learner, version 6.2, reference manual. Technical Report ILK 09-01, Induction of Linguistic Knowledge, Tilburg University.

Dagan, I., O. Glickman, and B. Magnini. 2005. The PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, Southampton, U.K.

Gildea, Daniel. 2003. Loosely tree-based alignment for machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 80–87, Sapporo, Japan.

Groves, D., M. Hearne, and A. Way. 2004. Robust subsentential alignment of phrase-structure trees. In *Proceedings of the 20th International Conference on Computational Linguistics (CoLing '04)*, pages 1072–1078.

Krahmer, Emiel, Erwin Marsi, and Paul van Pelt. 2008. Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion. In Moore, J., S. Teufel, J. Allan, and S. Furui, editors, *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 193–196, Columbus, Ohio, USA.

Kuhn, Harold W. 1955. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Lavie, A., A. Parlikar, and V. Ambati. 2008. Syntax-driven learning of sub-sentential translation equivalents and translation rules from parsed parallel corpora. In *Proceedings of the Second Workshop on Syntax and Structure in Statistical Translation*, pages 87–95.

Lin, D. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, pages 296–304.

MacCartney, B. and C.D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 521–528.

MacCartney, Bill, Michel Galley, and Christopher D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 802–811, Honolulu, Hawaii, October.

Marsi, Erwin and Emiel Krahmer. 2005a. Classification of semantic relations by humans and machines. In *Proceedings of the ACL 2005 workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 1–6, Ann Arbor, Michigan.

Marsi, Erwin and Emiel Krahmer. 2005b. Explorations in sentence fusion. In *Proceedings of the 10th European Workshop on Natural Language Generation*, Aberdeen, GB.

Radev, D.R. and K.R. McKeown. 1998. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):469–500.

Tiedemann, J. and G. Kotzé. 2009. Building a Large Machine-Aligned Parallel Treebank. In *Eighth International Workshop on Treebanks and Linguistic Theories*, page 197.

Tinsley, J., V. Zhechev, M. Hearne, and A. Way. 2007. Robust language-pair independent sub-tree alignment. *Machine Translation Summit XI*, pages 467–474.

Vossen, P., I. Maks, R. Segers, and H. van der Vliet. 2008. Integrating lexical units, synsets and ontology in the Cornetto Database. In *Proceedings of LREC 2008*, Marrakech, Morocco.

Controlling Listening-oriented Dialogue using Partially Observable Markov Decision Processes

Toyomi Meguro†**, Ryuichiro Higashinaka**‡**, Yasuhiro Minami**†**, Kohji Dohsaka**†
†NTT Communication Science Laboratories, NTT Corporation
‡NTT Cyber Space Laboratories, NTT Corporation
meguro@cslab.kecl.ntt.co.jp
higashinaka.ryuichiro@lab.ntt.co.jp
{minami,dohsaka}@cslab.kecl.ntt.co.jp

Abstract

This paper investigates how to automatically create a dialogue control component of a listening agent to reduce the current high cost of manually creating such components. We collected a large number of listening-oriented dialogues with their user satisfaction ratings and used them to create a dialogue control component using partially observable Markov decision processes (POMDPs), which can learn a policy to satisfy users by automatically finding a reasonable reward function. A comparison between our POMDP-based component and other similarly motivated systems using human subjects revealed that POMDPs can satisfactorily produce a dialogue control component that can achieve reasonable subjective assessment.

1 Introduction

Although task-oriented dialogue systems have been actively researched (Hirshman, 1989; Ferguson et al., 1996; Nakano et al., 1999; Walker et al., 2002), recently non-task-oriented functions are starting to attract attention, and systems without a specific task that deal with more casual dialogues, such as chats, are being actively investigated from their social and entertainment aspects (Bickmore and Cassell, 2001; Higashinaka et al., 2008; Higuchi et al., 2008).

In the same vein, we have been working on listening-oriented dialogues in which one conversational participant attentively listens to the other (hereafter, listening-oriented dialogue). Our aim is to build listening agents that can implement such a listening process so that users can satisfy their desire to speak and be heard. Figure 1 shows an excerpt from a typical listening-oriented dialogue. In the literature, dialogue control components for less (or non-) task-oriented dialogue systems, such as listening agents, have typically used hand-crafted rules for dialogue control, which can be problematic because completely covering all dialogue states by hand-crafted rules is difficult when the dialogue has fewer task restrictions (Wallace, 2004; Isomura et al., 2009).

To solve this problem, this paper aims to automatically build a dialogue control component of a listening agent using partially observable Markov decision processes (POMDPs). POMDPs, which make it possible to learn a policy that can maximize the averaged reward in partially observable environments (Pineau et al., 2003), have been successfully adopted in task-oriented dialogue systems for learning a dialogue control module from data (Williams and Young, 2007). However, no work has attempted to use POMDPs for less (or non-) task-oriented dialogue systems, such as listening agents, because user goals are not as well-defined as task-oriented ones, complicating the finding of a reasonable reward function.

We apply POMDPs to listening-oriented dialogues by having the system learn a policy that simultaneously maximizes how well users feel that they are being listened to (hereafter, user satisfaction) and how smoothly the system generates dialogues (hereafter, smoothness). This formulation is new; no work has considered both user satisfaction and smoothness using POMDPs. We collected a large amount of listening-oriented dialogues and annotated them with dialogue acts and also obtained subjective evaluation results for them. From them, we calculated the rewards and learned the POMDP policies. We evaluated the dialogue-act tag sequences of our POMDPs using human subjects.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 761–769,
Beijing, August 2010

Utterance	Dialogue act
S: Good evening.	Greeting
The topic is “food,” nice to meet you.	Greeting
L: Nice to meet you, too.	Greeting
S: I had curry for dinner.	S-Disc (sub: fact)
Do you like curry?	Question (sub: pref)
L: Yes, I do.	Sympathy
S: Really?	Repeat
Me, too.	Sympathy
L: Do you usually go out to eat?	Question (sub: habit)
S: No, I always cook at home.	S-Disc (sub: habit)
I don’t use any special spices, but I sometimes cook noodles using soup and curry.	S-Disc (sub: habit)
L: That sounds good!	S-Disc (sub: pref (positive))

Figure 1: Excerpt of a typical listening-oriented dialogue. Dialogue topic is “food.” Dialogue acts corresponding to utterances are shown in parentheses (See Table 1 for meanings): S-Disc stands for Self-disclosure; pref for preference; S for speaker; and L for listener. The dialogue was translated from Japanese by the authors.

The next section introduces related work. Section 3 describes our approach. Section 4 describes our collection of listening-oriented dialogues. This is followed in Section 5 by an evaluation experiment that compared our POMDP-based dialogue control with other similarly motivated systems. The last section summarizes the main points and mentions future work.

2 Related work

With increased attention on social dialogues and senior peer counseling, work continues to emerge on listening-oriented dialogues. One early work is (Maatman et al., 2005), which showed that virtual agents can give users the sense of being heard using such gestures as nodding and head shaking. Recently, Meguro et al. (2009a) analyzed the characteristics of listening-oriented dialogues. They compared listening-oriented dialogues and casual conversations between humans, revealing that the two types of dialogues have significantly different flows and that listeners actively question with frequently inserted self-disclosures; the speaker utterances were mostly concerned with self-disclosure.

Shitaoka et al. (2010) also investigated the functions of listening agents, focusing on their response generation components. Their system takes the confidence score of speech recognition into account and changes the system response accordingly; it repeats the user utterance or makes an empathic utterance for high-confidence user utterances and makes a backchannel when the confidence is low. The system’s empathic utterances can be “I’m happy” or “That’s too bad,” depending on whether a positive or negative expression is included in the user utterances. Their system’s response generation only uses the speech recognition confidence and the polarity of user utterances as cues to choose its actions. Currently, it does not consider the utterance content or the user intention.

In order for listening agents to achieve high smoothness, a switching mechanism between the “active listening mode,” in which the system is a listener, and the “topic presenting mode,” in which the system is a speaker, has been proposed (Yokoyama et al., 2010; Kobayashi et al., 2010). Here, the system uses a heuristic function to maintain a high user interest level and to keep the system in an active listening mode. Dialogue control is done by hand-crafted rules. Our motivation bears some similarity to theirs in that we want to build a listening agent that gives users a sense of being heard; however, we want to automatically make such an agent from dialogue data.

POMDPs have been introduced for robot action control (Pineau et al., 2003). Here, the system learns to make suitable movements for completing a certain task. Over the years, POMDPs have been actively studied for applications to spoken dialogue systems. Williams et al. (2007) successfully used a POMDP for dialogue control in a ticket-buying domain in which the objective was to fix the departure and arrival places for tickets. Recent work on POMDPs indicates that it is possible to train a dialogue control module in task-oriented dialogues when the user goal is obvious. In contrast, in this paper, we aim to verify whether POMDPs can be applied to less task-oriented dialogues (i.e., listening-oriented dialogues) where user goals are not as obvious.

In a recent study, Minami et al. (2009) applied POMDPs to non-task-oriented man-machine interaction. Their system learned suitable action control of agents that can act smoothly by obtaining rewards from the statistics of artificially generated data. Our work is different because we use real human-human dialogue data to train POMDPs for dialogue control in listening-oriented dialogues.

3 Approach

A typical dialogue system has utterance understanding, dialogue control, and utterance generation modules. The utterance understanding module comprehends user natural-language utterances, whose output (i.e., a user dialogue act) is passed to the dialogue control module. The dialogue control module chooses the best system dialogue act at every dialogue point using the user dialogue act as input. The utterance generation module generates natural-language utterances and says them to users by realizing the system dialogue acts as surface forms.

This paper focuses on the dialogue control module of a listening agent. Since a listening-oriented dialogue has a characteristic conversation flow (Meguro et al., 2009a), focusing on this module is crucial because it deals with the dialogue flow. Our objective is to train from data a dialogue control module that achieves a smooth dialogue flow that makes users feel that they are being listened to attentively.

3.1 Dialogue control using POMDPs

The purpose of our dialogue control is to simultaneously create situations in which users feel listened to (i.e., user satisfaction) and to generate smooth action sequences (i.e., smoothness). To do this, we automatically and statistically train the reward and the policy of the POMDP using a large amount of listening-oriented dialogue data. POMDP is a reinforcement learning framework that can learn a policy to select an action sequence that maximizes average future rewards. Setting a reward is crucial in POMDPs.

For our purpose, we introduce two different rewards: one for user satisfaction and the other for smoothness. Before creating a POMDP structure, we used the dynamic Bayesian network (DBN) structure (Fig. 2) to obtain the statistical structure of the data and the two rewards.

The random values in the DBN are as follows: s_o and s_a are the dialogue state and action state, o is a speaker observation, a is a listener action, and d is a random variable for an evaluation score that indicates the degree of the user being listened to. This evaluation score can be obtained by questionnaires, and the variable is used for calculating a user satisfaction reward for the POMDP. The DBN arcs in Fig. 2 define the emission and transition probabilities. $\Pr(o'|s'_o)$ is the emission probability of o' given s'_o. $\Pr(d|s_o)$ is the emission probability of d given s_o. $\Pr(s'_o|s_o, a)$ is a transition probability from s_o to s'_o given a. The DBN is trained using the EM algorithm. Using the obtained variables, we calculate the two reward functions as follows:

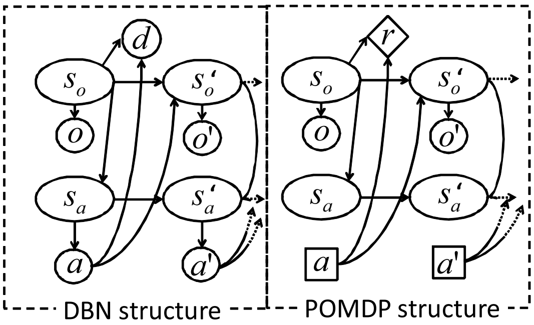


Figure 2: DBN and POMDP structures employed in this paper. Note that a in the POMDP is isolated from other states because it is decided by a learned policy.

(1) Reward for user satisfaction This reward is obtained from the d variable by

$$r_1((s_o, *), a) = \sum_{d=min}^{max} d \times \Pr(d|s_o, a),$$

where * is arbitrary s_a and min and max are minimum and maximum evaluation scores.

(2) Reward for smoothness For smoothness, we maximize the action predictive probability given the history of actions and observations. The probability is calculated from listening-oriented dialogue data. s_a is introduced for estimating the predictive probability of action a and for selecting a to maximize the predictive probability.

We set $\Pr(a|s_a) = 1$ when $a = s_a$ so that s_a corresponds one-on-one with a. Then, if $a_t = s_a$ at time t is given, we obtain

$$\begin{aligned}&\Pr(a_t|o_1, a_1, \ldots, a_{t-1}, o_t)\\&\quad= \sum_{s'_a} \Pr(a_t|s'_a) \Pr(s'_a|o_1, a_1, \ldots, a_{t-1}, o_t)\\&\quad= \Pr(s_a|o_1, a_1, \ldots, o_{t-1}, a_{t-1}, o_t)\end{aligned}$$

Consequently, maximizing the predictive probability of a equals maximizing that of s_a. If we set 1.0 to reward $r_2((*, s_a), a)$ when $s_a = a$, the POMDP will generate actions that maximize their predictive probabilities. We believe that this reward should increase the smoothness of a system action sequence since the sequence is generated according to the statistics of human-human dialogues.

Converting a DBN into a POMDP The DBN is converted into a POMDP (Fig. 2), while maintaining the transition and output probabilities. We convert d to r as described above.

The system is in a partially observed state. Since the state is not known exactly, we use a distribution called "belief state" b_t with which we obtain the average reward that will be gained in the future at time t by:

$$V_t = \sum_{\tau=0}^{\infty} \gamma^{\tau} \sum_{s} b_{\tau+t}((s_o, s_a)) r((s_o, s_a), a_{\tau+t}),$$

where τ is a discount factor; namely, the future reward is decreased by τ. A policy is learned by value iteration so that the action that maximizes V_t can be chosen. We define $r((s_o, s_a), a)$ as follows:

$$r((s_o, s_a), a) = r_1((s_o, *), a) + r_2((*, s_a), a).$$

By balancing these two rewards, we can choose an action that satisfies both user satisfaction and smoothness.

4 Data collection

We collected listening-oriented dialogues using human subjects who consisted of ten listeners (five males and five females) and 37 speakers (18 males and 19 females). The listeners and speakers ranged from 20 to 60 years old and were all native Japanese speakers. Listeners and speakers were matched to form a listener-speaker pair and communicated over the Internet with our chat interface. They used only text; they were not allowed to use voice, video, or facial expressions. The speakers chose their own listener and freely participated in dialogues from 7:00 pm to midnight for a period of 15 days. One conversation was restricted to about ten minutes. The subjects talked about a topic chosen by the speaker. There were 20 predefined topics: money, sports, TV and radio, news, fashion, pets, movies, music, housework and childcare, family, health, work, hobbies, food, human relationships, reading, shopping, beauty aids, travel, and miscellaneous. The listeners were instructed to make it easy for the speakers to say what the speakers wanted to say. We collected 1260 listening-oriented dialogues.

4.1 Dialogue-act annotation

We labeled the collected dialogues using the dialogue-act tag set shown in Table 1. We made these tags by selecting, extending, and modifying those from previous studies that concerned human listening behaviors in some way (Meguro et al., 2009a; Jurafsky et al., 1997; Ivey and Ivey, 2002). In our tag set, only question and self-disclosure tags have sub-category tags. Two annotators (not the authors) labeled each sentence of our collected dialogues using these 32 tags. In dialogue-act annotation, since there can be several sentences in one utterance, one annotator first split the utterances into sentences, and then both annotators labeled each sentence with a single dialogue act.

4.2 Obtaining evaluation scores

POMDPs need evaluation scores (i.e., d) for dialogue acts (i.e., a) for training a reward function. Therefore, we asked a third-party participant, who was neither a listener nor a speaker in our dialogue data collection, to evaluate the user satisfaction levels of the collected dialogues. She rated each dialogue in terms of how she would have felt "being heard" after the dialogue if she had been the speaker of the dialogue in question. She provided ratings on the 7-point Likert scale for each dialogue. Since she rated the whole dialogue with a single rating, we set the evaluation score of each action within a dialogue using the evaluation score for that dialogue.

We used a third-person's evaluation and not the original person's to avoid the fact that the evaluative criterion is too different between humans; identical evaluation scores from two people do not necessarily reflect identical user satisfaction levels. We highly valued the reliability and consistency of the third-person scores. This way, at least, we can train a policy that maximizes its average reward function for the rater, which we need to verify first before considering adaptation to two or more individuals.

5 Experiment

5.1 Experimental setup

The experiment followed three steps.

GREETING	Greeting and confirmation of dialogue theme. e.g., Hello. Let's talk about lunch.
INFORMATION	Delivery of objective information. e.g., My friend recommended a restaurant.
SELF-DISCLOSURE	Disclosure of preferences and feelings.
sub: fact	e.g., I live in Tokyo.
sub: experience	e.g., I had a hamburger for lunch.
sub: habit	e.g., I always go out for dinner.
sub: preference (positive)	e.g., I like hamburgers.
sub: preference (negative)	e.g., I don't really like hamburgers.
sub: preference (neutral)	e.g., Its taste is near my homemade taste.
sub: desire	e.g., I want to try it.
sub: plan	e.g., I'm going there next week.
sub: other	
ACKNOWLEDGMENT	Encourage the conversational partner to speak. e.g., Well. Aha.
QUESTION	Utterances that expect answers.
sub: information	e.g., Please tell me how to cook it.
sub: fact	e.g., What kind of curry?
sub: experience	e.g., What did you have for dinner?
sub: habit	e.g., Did you cook it yourself?
sub: preference	e.g., Do you like it?
sub: desire	e.g., Don't you want to eat rice?
sub: plan	e.g., What are you going to have for dinner?
sub: other	
SYMPATHY	Sympathetic utterances and praises. e.g., Me, too.
NON-SYMPATHY	Negative utterances. e.g., Not really.
CONFIRMATION	Confirm what the conversation partner said. e.g., Really?
PROPOSAL	Encourage the partner to act. e.g., Try it.
REPEAT	Repeat the partner's utterance.
PARAPHRASE	Paraphrase the partner's utterance.
APPROVAL	Broach or show goodwill toward the partner. e.g., Absolutely!
THANKS	Express thanks e.g., Thank you.
APOLOGY	Express regret e.g., I'm sorry.
FILLER	Filler between utterances. e.g., Uh. Let me see.
ADMIRATION	Express affection. e.g., Ha-ha.
OTHER	Other utterances.

Table 1: Definition and example of dialogue acts

	All	Food (subset of All)
# dialogues	1260	250
# words	479881	94867
# utterances per dialogue	28.2	29.1
# dialogues per listener	126	25
# dialogues per speaker	34	6.8
# dialogue acts	67801	13376
inter-annotator agreement	0.57	0.55

Table 2: Statistics of collected dialogues and dialogue-act annotation. Inter-annotator agreement means agreement of dialogue-act annotation using Cohen's κ.

In the first step, we created our POMDP system using our approach (See Section 3.1). We also made five other systems for comparison that we describe in Section 5.2. Each system outputs dialogue-act tag sequences for evaluation. The dialogue theme was "food" because it was the most frequent theme and accounted for 20% of our data (See Table 2 for the statistics); we trained our POMDP using the "food" dialogues. We restricted the dialogue topic to verify that our approach at least works with a small set. Since there is no established measure for automatically evaluating a dialogue-act tag sequence, we evaluated our dialogue control module using human subjective evaluations. However, this is very difficult to do because dialogue control modules only output dialogue acts, not natural language utterances.

In the second step, we recruited participants who created natural language utterances from dialogue-act tag sequences. In their creating dialogues, we provided them with situations to stimulate their imaginations. Table 3 shows the situations, which were deemed common in everyday Japanese life; we let the participants create utterances that fit the situations. These situations were necessary because, without restrictions, the evaluation scores could be influenced by dialogue content rather than by dialogue flow.

For this dialogue-imagining exercise, we recruited 16 participants (eight males and eight females) who ranged from 19 to 39 years old. Each participant made twelve dialogues using two situations. For assigning the situations, we first created four conditions: (1) a student and living with family, (2) working and living with family, (3) a student and living alone, and (4) working and living alone. Then the participants were categorized into one of these conditions on the basis of their actual lifestyle and assigned two of the situations matching the condition.

For each situation, each participant created six imaginary dialogues from the six dialogue-act sequences output by the six systems: our POMDP and the other five systems for comparison. This process produced such dialogues as shown in Figs. 5 and 6. The dialogue in Fig. 5 was made from a dialogue-act tag sequence of a human-human conversation using No. 1 of Table 3. The dialogue in Fig. 6 was made from the sequence of our POMDP using No. 2 of Table 3.

In the third step, we additionally recruited three judges (one male and two females) to evaluate the imagined 192 (16 × 2 × 6) dialogues. The judges were neither the participants who made dialogues nor those who rated the collected listening-oriented dialogues. Six dialogues made from one situation were randomly shown to the judges one-by-one, who then filled out questionnaires to indicate their user satisfaction levels by answering this question on a 7-point Likert scale: “If you had been the speaker, would you have felt that you were listened to?”

5.2 Systems for comparison

We created our POMDP-based dialogue control and five other systems for comparison.

POMDP We learned a policy based on our approach. We used “food” dialogues (See Section 4), and the evaluation scores were those described in Section 4.2. This system used the policy to generate sequences of dialogue-act tags by simulation; user observations were generated based on emission probability, and system actions were generated based on the policy.

In this paper, the total number of observations and actions was 33 because we have 32 dialogue-act tags (See Table 1) plus a “skip” tag. In learning the policy, an observation and an action must individually take turns, but our data can include multiple dialogue-act tags in one utterance. Therefore, if there is more than one dialogue-act tag in one utterance, a “skip” is inserted between the tags. The state numbers for S_o and S_a were 16 and 33, respectively. In this experiment, we set 10 to $r_2((*, s_a), a)$.

EvenPOMDP We arranged a POMDP using only the smoothness reward (hereafter, EvenPOMDP) by creating a POMDP system with a fixed evaluation score; hence user satisfaction is not incorporated in the reward. When using fixed (even) evaluation scores for all dialogues, the effect of the user satisfaction reward is denied, and the system only generates highly frequent sequences. We have EvenPOMDP to clarify whether user satisfaction is necessary. The other conditions are identical as in the POMDP system.

HMM We modeled our dialogue-act tag sequences using a Speaker HMM (SHMM) (Meguro et al., 2009a), which has been utilized to model two-party listening-oriented dialogues. In a SHMM, half the states emit listener dialogue acts, and the other half emit speaker dialogue acts. All states are connected to each other. We modeled the “food” dialogues using an SHMM, and made the model generate the most probable dialogue-act tag sequences. More specifically, first, a dialogue-act tag was generated randomly based on the initial state. If the state was that of a listener, we generated a maximum likelihood action and the state was randomly transited based on the transition probability. If the state was that of a speaker, we randomly generated an action based on the emission probability and the state was transited using the maximum likelihood transition probability.

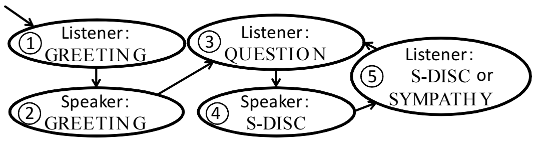


Figure 3: Structure of rule-based system

Rule-based system This system creates dialogue-act tag sequences using hand-crafted rules that are based on the findings in (Meguro et al., 2009a) and are realized as shown in Fig. 3. A sequence begins at state ① in Fig. 3, and one dialogue act is generated at each state. At state ③, a sub-category tag under QUESTION is chosen randomly, and at state ④, a matched sub-category tag under SELF-DISCLOSURE is chosen. At state ⑤, the listener’s SELF-DISCLOSURE or SYMPATHY is generated randomly.

Human dialogue sequence This system created dialogue-act tag sequences by randomly choosing dialogues between humans from the collected data and used their annotated tag sequences.

Random This system simply created dialogue-act tag sequences at random.

5.3 Experimental results

Figure 4 shows the average subjective evaluation scores. Except between HMM and EvenPOMDP, there was a significant difference (p<0.01) between all systems in a non-parametric multiple comparison test (Steel-Dwass test). The dialogues shown in Figs. 5 and 6 were generated by the systems. The dialogue in Fig. 5 was made from human dialogue sequences, and the one in Fig. 6 was made from POMDP.

	With whom	What day	What time	What	Where	Who made
1	family	weekday	around 6:00 pm	grilled salmon	home	mother
2	family	weekend	around 7:00 pm	potato and meat	home	mother
3	co-workers	weekday	at noon	boiled seaweed	lunch box	myself
⋮	⋮	⋮	⋮	⋮	⋮	⋮
32	friend	weekday	at noon	hamburger	school cafeteria	N/A

Table 3: Dialogue situations relating to everyday Japanese life

We qualitatively analyzed the dialogues of each system and observed the following characteristics:

POMDP At a dialogue's beginning, the system greets several times and shifts to a different phase in which listeners ask questions and self-disclose to encourage speakers to reciprocate.

Rule-based The output of this system seems very natural and easy to read. The dialogue-act tags followed reasonable rules, making it easier for the participants to create natural utterances from them.

Human conversation The dialogues between humans were obviously natural before they were changed to tags from the natural-language utterances. However, human dialogues have randomness, which makes it difficult for the participants to create natural-language utterances from the tags. Hence, the evaluation score for this system was lower than the "Rule-based."

HMM, EvenPOMDP Since these systems continually output the same action tags, their output was very unnatural. For example, greetings never stopped because GREETING is most frequently followed by GREETING in the data. These systems have no mechanism to stop this loop.

POMDP successfully avoided such continuation because its actions have more varied rewards. For example, GREETING is repeated in EvenPOMDP because its smoothness reward is high; however, in POMDP, although the smoothness reward remains high, its user satisfaction reward is not that high. This is because greetings appear in all dialogues and their user satisfaction reward converges to the average. Therefore, such actions as greetings do not get repeated in POMDP. In POMDP, some states have high user satisfaction rewards, and the POMDP policy generated actions to move to such states.

Random Since this system has more variety of tags than HMM, its evaluation scores outperformed HMM, but were outperformed by POMDP, which learned statistically from the data.

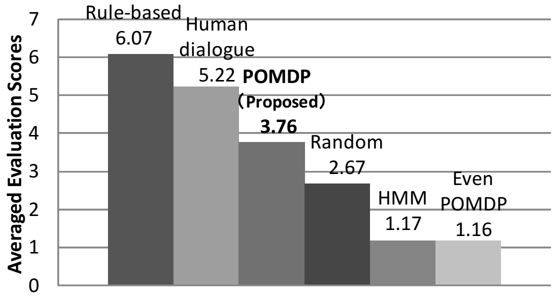


Figure 4: System scores. Except between POMDP and EvenPOMDP, significant differences exist among all systems (p<0.01).

From our qualitative analysis, we found that POMDP can generate more satisfying sequences than HMM/EvenPOMDP because it does not fall into the loop of frequent dialogue-act tag sequences. This suggests the usefulness of incorporating two kinds of rewards into the policy and that our approach for setting a reward is promising.

However, with the proposed POMDP, unnatural sequences remain; for example, the system suddenly output THANKS, as shown in Fig. 6. The number of states may have been too small. We plan to investigate what caused this in the future.

In our qualitative analysis, we observed that randomness in dialogues might hold a clue for improving evaluation scores. Therefore, we measured the perplexity of each system output using dialogue-act trigrams and obtained 72.8 for "Random," 27.4 for "Human dialogue," 7.4 for "POMDP," 3.2 for "HMM," 2.5 for "EvenPOMDP," and 1.7 for "Rule-based."

The perplexity of the human dialogues is less than that of the random system, but humans also exhibit a certain degree of freedom. On the other hand, POMDP's perplexity is less than the human dialogues; they still have some freedom, which probably led to their reasonable evaluation scores. Considering that HMM and EvenPOMDP, which continually output the same dialogue acts, had low

	Utterance	Dialogue act
S:	Hello.	Greeting
L:	Nice to meet you	Greeting
S:	I had dinner at home today.	S-Disc (sub: fact)
	Do you like grilled salmon?	Question, pref
L:	Yes, I think so.	Sympathy
	I sometimes want to have a fancy meal.	S-Disc (sub: desire)
S:	Deluxe.	Repeat
	Me too.	Sympathy
L:	Do you usually do your own cooking?	Question (sub: habit)
S:	No, I don't.	S-Disc, habit
	I always buy my meals at the convenience store.	S-Disc (sub: habit)
L:	I like the lunch boxes of convenience stores	S-Disc (sub: pref (positive))

Figure 5: Excerpt of listening-oriented dialogue that participant imagined from tag sequences of human conversations. Dialogue was translated from Japanese by the authors.

	Utterance	Dialogue act
L:	Nice to meet you.	Greeting
	Where and who did you have dinner with today?	Question (sub: fact)
S:	I had "niku-jaga" (meat and beef) with my family at home.	S-Disc (sub: fact)
L:	Oh.	Admiration
S:	I think it is normal to eat with your family at home.	S-Disc (sub: pref (neutral))
L:	Thanks.	Thanks
	Do you have any brothers or sisters?	Question (sub: fact)
	Soon, my brother and his wife will visit my home.	S-Disc (sub: plan)
S:	I see.	Sympathy
L:	I want to use expensive meat in my "niku-jaga."	S-Disc (sub: desire)
	Oh.	Admiration
	Please give me your recipe.	Question (sub: information)
S:	My friends claim that my "niku-jaga" is as good as a restaurant's.	Information
L:	I'd love to try it	S-Disc (sub: desire)

Figure 6: Excerpt of a listening-oriented dialogue made from tag sequences of POMDP

evaluation scores, we conclude that randomness is necessary in non-task-oriented dialogues and that some randomness can be included with our approach. We do not discuss "Rule-based" here because its tag sequence was meant to have small perplexity.

6 Conclusion and Future work

This paper investigated the possibility of automatically building a dialogue control module from dialogue data to create automated listening agents.

With a POMDP as a learning framework, a dialogue control module was learned from the listening-oriented dialogues we collected and compared with five different systems. Our POMDP system showed higher performance in subjective evaluations than other statistically motivated systems, such as an HMM-based one, that work by selecting the most likely subsequent action in the dialogue data. When we investigated the output sequences of our POMDP system, the system frequently chose to self-disclose and question, which corresponds to human listener behavior, as revealed in the literature (Meguro et al., 2009a). This suggests that learning dialogue control by POMDPs is achievable for listening-oriented dialogues.

The main contribution of this paper is that we successfully showed that POMDPs can be used to train dialogue control policies for less task-oriented dialogue systems, such as listening agents, where the user goals are not as clear as task-oriented ones. We also revealed that the reward function can be learned effectively by our formulation that simultaneously maximizes user satisfaction and smoothness. Finding an appropriate reward function is a real challenge for less task-oriented dialogue systems; this work has presented the first workable solution.

Much work still remains. Even though we conducted an evaluation experiment by simulation (i.e, offline evaluation), human dialogues obviously do not necessarily proceed as in simulations. Therefore, we plan to evaluate our system using online evaluation, which also forces us to implement utterance understanding and generation modules. We also want to incorporate the idea of topic shift into our policy learning because we observed in our data that listeners frequently change topics to keep speakers motivated. We are also considering adapting the system behavior to users. Specifically, we want to investigate dialogue control that adapts to the personality traits of users because it has been found that the flow of listening-oriented dialogues differs depending on the personality traits of users (Meguro et al., 2009b). Finally, although we only dealt with text, we also want to extend our approach to speech and other modalities, such as gestures and facial expressions.

References

Bickmore, Timothy and Justine Cassell. 2001. Relational agents: a model and implementation of building user trust. In *Proc. SIGCHI conference on human factors in computing systems (CHI)*, pages 396–403.

Ferguson, George, James F. Allen, and Brad Miller. 1996. TRAINS-95: towards a mixed-initiative planning assistant. In *Proc. Third Artificial Intelligence Planning Systems Conference (AIPS)*, pages 70–77.

Higashinaka, Ryuichiro, Kohji Dohsaka, and Hideki Isozaki. 2008. Effects of self-disclosure and empathy in human-computer dialogue. In *Proc. IEEE Workshop on Spoken Language Technology (SLT)*, pages 108–112.

Higuchi, Shinsuke, Rafal Rzepka, and Kenji Araki. 2008. A casual conversation system using modality and word associations retrieved from the web. In *Proc. 2008 conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 382–390.

Hirshman, Lynette. 1989. Overview of the DARPA speech and natural language workshop. In *Proc. DARPA Speech and Natural Language Workshop 1989*, pages 1–2.

Isomura, Naoki, Fujio Toriumi, and Kenichiro Ishii. 2009. Evaluation method of non-task-oriented dialogue system using HMM. *IEICE Transactions on Information and Systems*, J92-D(4):542–551.

Ivey, Allen E. and Mary Bradford Ivey. 2002. *Intentional Interviewing and Counseling: Facilitating Client Development in a Multicultural Society*. Brooks/Cole Publishing Company.

Jurafsky, Dan, Elizabeth Shriberg, and Debra Biasca, 1997. *Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual*.

Kobayashi, Yuka, Daisuke Yamamoto, Toshiyuki Koga, Sachie Yokoyama, and Miwako Doi. 2010. Design targeting voice interface robot capable of active listening. In *Proc. 5th ACM/IEEE international conference on Human-robot interaction (HRI)*, pages 161–162,

Maatman, R. M., Jonathan Gratch, and Stacy Marsella. 2005. Natural behavior of a listening agent. *Lecture Notes in Computer Science*, 3661:25–36.

Meguro, Toyomi, Ryuichiro Higashinaka, Kohji Dohsaka, Yasuhiro Minami, and Hideki Isozaki. 2009a. Analysis of listening-oriented dialogue for building listening agents. In *Proc. 10th Annual SIGDIAL Meeting on Discourse and Dialogue (SIGDIAL)*, pages 124–127.

Meguro, Toyomi, Ryuichiro Higashinaka, Kohji Dohsaka, Yasuhiro Minami, and Hideki Isozaki. 2009b. Effects of personality traits on listening-oriented dialogue. In *Proc. International Workshop on Spoken Dialogue Systems Technology (IWSDS)*, pages 104–107.

Minami, Yasuhiro, Akira Mori, Toyomi Meguro, Ryuichiro Higashinaka, Kohji Dohsaka, and Eisaku Maeda. 2009. Dialogue control algorithm for ambient intelligence based on partially observable markov decision processes. In *Proc. International Workshop on Spoken Dialogue Systems Technology (IWSDS)*, pages 254–263.

Nakano, Mikio, Noboru Miyazaki, Jun ichi Hirasawa, Kohji Dohsaka, and Takeshi Kawabata. 1999. Understanding unsegmented user utterances in real-time spoken dialogue systems. In *Proc. 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 200–207.

Pineau, Joelle., Geoff. Gordon, and Sebastian Thrun. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025–1032.

Shitaoka, Kazuya, Ryoko Tokuhisa, Takayoshi Yoshimura, Hiroyuki Hoshino, and Narimasa Watanabe. 2010. Active listening system for dialogue robot. In *JSAI SIG-SLUD Technical Report*, volume 58, pages 61–66. (in Japanese).

Walker, Marilyn, Alex Rudnicky, John Aberdeen, Elizabeth Owen Bratt, Rashmi Prasad, Salim Roukos, Greg S, and Seneff Dave Stallard. 2002. DARPA communicator evaluation: progress from 2000 to 2001. In *Proc. International Conference on Spoken Language Processing (ICSLP)*, pages 273–276.

Wallace, Richard S. 2004. *The Anatomy of A.L.I.C.E.* A.L.I.C.E. Artificial Intelligence Foundation, Inc.

Williams, Jason D. and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Yokoyama, Sachie, Daisuke Yamamoto, Yuka Kobayashi, and Miwako Doi. 2010. Development of dialogue interface for elderly people –switching the topic presenting mode and the attentive listening mode to keep chatting–. In *IPSJ SIG Technical Report*, volume 2010-SLP-80, pages 1–6. (in Japanese).

Recognising Entailment within Discourse

Shachar Mirkin[§], **Jonathan Berant**[†], **Ido Dagan**[§], **Eyal Shnarch**[§]

§ Computer Science Department, Bar-Ilan University
† The Blavatnik School of Computer Science, Tel-Aviv University

Abstract

Texts are commonly interpreted based on the entire discourse in which they are situated. Discourse processing has been shown useful for inference-based application; yet, most systems for textual entailment – a generic paradigm for applied inference – have only addressed discourse considerations via off-the-shelf coreference resolvers. In this paper we explore various discourse aspects in entailment inference, suggest initial solutions for them and investigate their impact on entailment performance. Our experiments suggest that discourse provides useful information, which significantly improves entailment inference, and should be better addressed by future entailment systems.

1 Introduction

This paper investigates the problem of recognising textual entailment within discourse. Textual Entailment (TE) is a generic framework for applied semantic inference (Dagan et al., 2009). Under TE, the relationship between a *text* (T) and a textual assertion (*hypothesis*, H) is defined such that *T entails H* if humans reading T would infer that H is most likely true (Dagan et al., 2006).

TE has been successfully applied to a variety of natural language processing applications, including information extraction (Romano et al., 2006) and question answering (Harabagiu and Hickl, 2006). Yet, most entailment systems have thus far paid little attention to discourse aspects of inference. In part, this is the result of the unavailability of adept tools for handling the kind of discourse processing required for inference. In addition in the main TE benchmarks, the Recognising Textual Entailment (RTE) challenges, discourse played little role. This state of affairs has started to change with the recent introduction of the RTE Pilot "Search" task (Bentivogli et al., 2009b), in which assessed texts are situated within complete documents. In this setting, texts need to be interpreted based on their entire discourse (Bentivogli et al., 2009a), hence attending to discourse issues becomes essential. Consider the following example from the task's dataset:

(T) *The seven men on board were said to have as little as 24 hours of air.*

For the interpretation of T, e.g. the identity and whereabouts of the seven men, one must consider T's discourse. The preceding sentence T', for instance, provides useful information to that aim:

(T') *The Russian navy worked desperately to save a small military submarine.*

This example demonstrates a common situation in texts, and is also applicable to the RTE Search task's setting. Still, little was done by the task's participants to consider discourse, and sentences were mostly processed independently.

Analyzing the Search task's development set, we identified several key discourse aspects that affect entailment in a discourse-dependent setting. First, we observed that the coverage of available coreference resolution tools is considerably limited. To partly address this problem, we extend the set of coreference relations to phrase pairs with a certain degree of lexical overlap, as long as no semantic incompatibility is found between them. Second, many bridging relations (Clark, 1975) are realized in the form of "global information" perceived as known for entire documents. As bridging falls completely out of the scope of available resolvers, we address this phenomenon by identifying and weighting prominent document terms and allowing their incorporation in inference even

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 770–778,
Beijing, August 2010

when they are not explicitly mentioned in a sentence. Finally, we observed a coherence-related discourse phenomenon, namely inter-relations between entailing sentences in the discourse, such as the tendency of entailing sentences to be adjacent to one another. To that end, we apply a two-phase classification scheme, where a second-phase meta-classifier is applied, extracting discourse and document-level features based on the classification of each sentence on its own.

Our results show that, even when simple solutions are employed, the reliance on discourse-based information is helpful and achieves a significant improvement of results. We analyze the contribution of each component and suggest some future work to better attend to discourse in entailment systems. To our knowledge, this is the most extensive effort thus far to empirically explore the effect of discourse on entailment systems.

2 Background

Discourse plays a key role in text understanding applications such as question answering or information extraction. Yet, such applications typically only handle a narrow aspect of discourse, addressing coreference by term substitution (Dali et al., 2009; Li et al., 2009). The limited coverage and scope of existing tools for coreference resolution and the unavailability of tools for addressing other discourse aspects also contribute to this situation. For instance, VP anaphora and bridging relations are usually not handled at all by such resolvers. A similar situation is seen in the TE research field.

The prominent benchmark for entailment systems evaluation is the series of RTE challenges. The main task in these challenges has traditionally been to determine, given a text-hypothesis pair (T,H), whether T entails H. Discourse played no role in the first two RTE challenges as T's were constructed of short simplified texts. In RTE-3 (Giampiccolo et al., 2007), where some paragraph-long texts were included, inter-sentential relations became relevant for correct inference. Yet the texts in the task were manually modified to ensure they are self-contained. Consequently, little effort was invested by the challenges' participants to address discourse issues beyond the standard substitution of coreferring nominal phrases, using publicly available tools such as JavaRap (Qiu et al., 2004) or OpenNLP[1], e.g. (Bar-Haim et al., 2008).

A major step in the RTE challenges towards a more practical setting of text processing applications occurred with the introduction of the Search task in the Fifth RTE challenge (RTE-5). In this task entailing sentences are situated within documents and depend on other sentences for their correct interpretation. Thus, discourse becomes a substantial factor impacting inference. Surprisingly, discourse hardly received any treatment in this task beyond the standard use of coreference resolution (Castillo, 2009; Litkowski, 2009), and an attempt to address globally-known information by removing from H words that appear in document headlines (Clark and Harrison, 2009).

3 The RTE Search Task

The RTE-5 Search task was derived from the TAC Summarization task[2]. The dataset consists of several corpora, each comprised of news articles concerning a specific *topic*, such as the impact of global warming on the Arctic or the London terrorist attacks in 2005. *Hypotheses* were manually generated based on Summary Content Units (Nenkova et al., 2007), clause-long statements taken from manual summaries of the corpora. *Texts* are unmodified sentences in the articles. Given a topic and a hypothesis, entailment systems are required to identify all sentences in the topic's corpus that entail the hypothesis.

Each sentence-hypothesis pair in both the development and test sets was annotated, judging whether the sentence entails the hypothesis. Out of 20,104 annotations in the development set, only 810 were judged as positive. This small ratio (4%) of positive examples, in comparison to 50% in traditional RTE tasks, better corresponds to the natural distribution of entailing texts in a corpus, thus better simulates practical settings.

The task may seem as a variant of information retrieval (IR), as it requires finding specific texts in a corpus. Yet, it is fundamentally different from IR for two reasons. First, the target output is a set

[1] http://opennlp.sourceforge.net

[2] http://www.nist.gov/tac/2009/Summarization/

of sentences, each evaluated independently, rather than a set of documents. Second, the decision criterion is *entailment* rather than *relevance*.

Despite the above, apparently, IR techniques provided hard-to-beat baselines for the RTE Search task (MacKinlay and Baldwin, 2009), outperforming every other system that relied on inference without IR-based pre-filtering. At the current state of performance of entailment systems, it seems that lexical coverage largely overshadows any other approach in this task. Still, most (6 out of 8) participants in the challenge applied their entailment systems to the entire dataset without a prior retrieval of candidate sentences. F_1 scores for such systems vary between 10% and 33%, in comparison to over 40% of the IR-based methods.

4 The Baseline RTE System

In this work we used BIUTEE, Bar-Ilan University Textual Entailment Engine (Bar-Haim et al., 2008; Bar-Haim et al., 2009), a state of the art RTE system, as a baseline and as a basis for our discourse-based enhancements. This section describes this system's architecture; the methods by which it was augmented to address discourse are presented in Section 5.

To determine entailment, BIUTEE performs the following main steps:

Preprocessing First, all documents are parsed and processed with standard tools for named entity recognition (Finkel et al., 2005) and coreference resolution. For the latter purpose, we use OpenNLP and enable the substitution of coreferring terms. This is the only way by which BIUTEE addresses discourse, representing the state of the art in entailment systems.

Entailment-based transformations Given a T-H pair (both represented as dependency parse trees), the system applies a sequence of knowledge-based entailment transformations over T, generating a set of texts which are entailed by it. The goal is to obtain consequent texts which are more similar to H. Based on preliminary results on the development set, in our experiments (Section 6) we use WordNet (Fellbaum, 1998) as the system's only knowledge resource, using its synonymy, hyponymy and derivation relations.

Classification A supervised classifier, trained on the development set, is applied to determine entailment of each pair based on a set of syntactic and lexical syntactic features assessing the degree by which T and its consequents cover H.

5 Addressing Discourse

In the following subsections we describe the prominent discourse phenomena that affect inference, which we have identified in an analysis of the development set and addressed in our implementation. As mentioned, these phenomena are poorly addressed by available reference resolvers or fall completely out of their scope.

5.1 Augmented coreference set

A large number of coreference relations are comprised of terms which share lexical elements, (e.g. *"airliners's first flight"* and *"Airbus A380's first flight"*). Although common in coreference relations, standard resolvers miss many of these cases. For the purpose of identifying additional coreferring terms, we consider two noun phrases in the same document as coreferring if: (i) their heads are identical and (ii) no semantic incompatibility is found between their modifiers. The types of incompatibility we handle are: (a) mismatching numbers, (b) antonymy and (c) co-hyponymy (coordinate terms), as specified by WordNet. For example, two nodes of the noun *distance* would be considered incompatible if one is modified by *short* and the second by its antonym *long*. Similarly, two modifier co-hyponyms of *distance*, such as *walking* and *running* would also result such an incompatibility. Adding more incompatibility types (e.g. *first* vs. *second flight*) may further improve the precision of this method.

5.2 Global information

Key terms or prominent pieces of information that appear in the document, typically at the title or the first few sentences, are many times perceived as "globally" known throughout the document. For example, the geographic location of the document theme, mentioned at the beginning of the document, is assumed to be known from that point on, and will often not be mentioned explicitly in further sentences. This is a bridging phenomenon that is typically not addressed by available discourse processing tools. To compensate for that, we identify key terms for each document based on *tf-idf* scores and consider them as global information for that document. For example, global terms for the topic discussing the ice melting in the Arctic, typically contain a location such as *Arctic* or *Antarctica* and terms referring to *ice*, like *permafrost* or *iceshelf*.

We use a variant of *tf-idf*, where term frequency is computed as follows: $tf(t_{i,j}) = n_{i,j} + \vec{\lambda}^{\top} \cdot \vec{f_{i,j}}$. Here, $n_{i,j}$ is the frequency of term i in document j ($t_{i,j}$), which is incremented by additional positive weights ($\vec{\lambda}$) for a set of features ($\vec{f_{i,j}}$) of the term. Based on our analysis, we defined the following features, which correlated mostly with global information: (i) does the term appear in the title? (ii) is it a proper name? (iii) is it a location? The weights for these features are set empirically.

The document's top-n global terms are added to each of its sentences. As a result, a global term that occurs in the hypothesis is matched in each sentence of the document, regardless of whether the term explicitly appears in the sentence.

Considering the previous sentence Another method for addressing missing coreference and bridging relations is based on the assumption that adjacent sentences often refer to the same entities and events. Thus, when extracting classification features for a given sentence, in addition to the features extracted from the parse tree of the sentence itself, we extract the same set of features from the current and previous sentences together. Recall the example presented in Section 1. T is annotated as entailing the hypothesis *"The AS-28 mini-submarine was trapped underwater"*, but the word *submarine*, e.g., appears only in its preceding sentence T'. Thus, considering both sentences together when classifying T increases its coverage of the hypothesis. Indeed, a bridging reference relates *on board* in T with *submarine* in T', justifying our assumption in this case.

5.3 Document-level classification

Beyond discourse references addressed above, further information concerning discourse and document structure is available in the Search setting and may contribute to entailment classification. We observed that entailing sentences tend to come in bulks. This reflects a common coherence aspect, where the discussion of a specific topic is typically continuous rather than scattered across the entire document. This *locality* phenomenon may be useful for entailment classification since knowing that a sentence entails the hypothesis increases the probability that adjacent sentences entail the hypothesis as well.

To capture this phenomenon, we use a two-phase meta-classification scheme, in which a *meta-classifier* utilizes entailment classifications of the first classification phase to extract *meta-features* and determine the final classification decision. This scheme also provides a convenient way to combine scores from multiple classifiers used in the first classification phase. We refer to these as *base-classifiers*. This scheme and the meta-features we used are detailed hereunder.

Let us write (s, h) for a sentence-hypothesis pair. We denote the set of pairs in the development (training) set as $\mathcal{D}$ and in the test set as $\mathcal{T}$. We split $\mathcal{D}$ into two halves, $\mathcal{D}_1$ and $\mathcal{D}_2$. We make use of n base-classifiers, $C_1, \ldots, C_n$, among which $C^\star$ is a designated classifier with additional roles in the process, as described below. Classifiers may differ, for example, in their classification algorithm. An additional meta-classifier is denoted C_M. The classification scheme is shown as Algorithm 1.

Algorithm 1 Meta-classification

```
Training
1: Extract features for every $(s, h)$ in $\mathcal{D}$
2: Train $C_1, \ldots, C_n$ on $\mathcal{D}_1$
3: Classify $\mathcal{D}_2$, using $C_1, \ldots, C_n$
4: Extract meta-features for $\mathcal{D}_2$ using the
   classification of $C_1, \ldots, C_n$
5: Train $C_M$ on $\mathcal{D}_2$

Classification
6: Extract features for every $(s, h)$ in $\mathcal{T}$
7: Classify $\mathcal{T}$ using $C_1, \ldots, C_n$
8: Extract meta-features for $\mathcal{T}$
9: Classify $\mathcal{T}$ using $C_M$
```

At Step 1, features are extracted for every (s, h) pair in the training set, as in the baseline system.

#	Ent?	Closest	*d*	2nd closest	*d*
1	NO	6	5	7	6
2	NO	6	4	7	5
3	NO	6	3	7	4
4	NO	6	2	7	3
5	NO	6	1	7	2
6	YES	7	1	7	1
7	YES	6 or 8	1	6 or 8	1
8	YES	7 or 9	1	7 or 9	1
9	YES	8	1	8	1
10	NO	8	1	8	2
11	NO	8	2	8	3

Figure 1: Comparison of the *closest* and *second-closest* schemes when applied to a bulk of entailing sentences (in white) situated within a non-entailing environment (in gray). Unlike the *closest* one, the *second-closest* scheme assigns larger distance values to non-entailing sentences located on the 'edge' of the bulk (5 and 10) than to entailing ones.

In Steps 2 and 3 we split the training set into two halves (taking half of each topic), train n different classifiers on the first half and then classify the second half using each of the n classifiers. Given the classification scores of the n base-classifiers to the (s, h) pairs in the second half of the training set, $\mathcal{D}_2$, we add in Step 4 the meta-features described in Section 5.3.1.

After adding the meta-features, we train (Step 5) a meta-classifier on this new set of features. Test sentences then go through the same process: features are extracted for them and they are classified by the already trained n classifiers (Steps 6 and 7), meta-features are extracted in Step 8, and a final classification decision is made by the meta-classifier in Step 9.

A retrieval step may precede the actual entailment classification, allowing the processing of fewer and potentially "better" candidates.

5.3.1 Meta-features

The following features are extracted in our meta-classification scheme:

Classification scores The classification score of each of the n base-classifiers.

Title entailment In many texts, and in news articles in particular, the title and the first few sentences often represent the entire document's content. Thus, knowing whether these sentences entail the hypothesis may be an indicator to the general potential of the document to include entailing sentences. Two binary features are added according to the classification of $C^\star$ indicating whether the title entails the hypothesis and whether the first sentence entails it.

Second-closest entailment Considering the locality phenomenon described above, we add a feature assigning higher scores to sentences in the vicinity of an entailment environment. This feature is computed as the distance to the second-closest entailing sentence in the document (counting the sentence itself as well), according to the classification of $C^\star$. Formally, let i be the index of the current sentence and $\mathcal{J}$ be the set of indices of entailing sentences in the document according to $C^\star$. For each $j \in \mathcal{J}$ we compute $d_{i,j} = |i-j|$, and choose the second smallest $d_{i,j}$ as d_i. The idea is that if entailing sentences indeed always come in bulks, then $d_i = 1$ for all entailing sentences, but $d_i > 1$ for all non-entailing ones. Figure 1 illustrates such a case, comparing the *second-closest* distance with the distance to the *closest* entailing sentence. In the *closest* scheme we do not count the sentence as closest to itself since it would disregard the environment of the sentence altogether, eliminating the desired effect. We scale the distance and add the feature score: $-\log(d_i)$.

Smoothed entailment This feature addressed the locality phenomenon by smoothing the classification score of sentence i with the scores of adjacent sentences, weighted by their distance from the current sentence i. Let $s(i)$ be the score assigned by $C^\star$ to sentence i. We add the Smoothed Entailment feature score:

$$\text{SE}(i) = \frac{\sum_w (b^{|w|} \cdot s(i+w))}{\sum_w (b^{|w|})}$$

where $0 < b < 1$ is the decay parameter and w is an integer bounded between $-N$ and N, denoting the distance from sentence *i*.

1$^{\text{st}}$ sentence entailing title Bensley and Hickl (2008) showed that the first sentence in a news article typically entails the article's title. We therefore assume that in each document, $s_1 \Rightarrow s_0$, where s_1 and s_0 are the document's first sentence and title respectively. Hence, under entailment transitivity, if $s_0 \Rightarrow h$ then $s_1 \Rightarrow h$. The corresponding binary feature states whether the sentence being classified is the document's first sentence *and* the title entails h according to $C^\star$.

	P (%)	R (%)	F_1 (%)
BIU-BL	14.53	55.25	23.00
BIU-DISC	20.82	57.25	30.53
BIU-BL^3	14.86	59.00	23.74
BIU-DISC$_{no-loc}$	22.35	57.12	32.13
All-yes baseline	4.6	100.0	8.9

Table 1: Micro-average results.

Note that the above locality-based features rely on high accuracy of the base classifier $C^{\star}$. Otherwise, it will provide misleading information to the features computation. We analyze the effect of this accuracy in Section 6.

6 Results and Analysis

Using the RTE-5 Search data, we compare BiuTee in its baseline configuration (cf. Section 4), denoted *BIU-BL*, with its discourse-aware enhancement (*BIU-DISC*) which uses all the components described in Section 5. To alleviate the strong IR effect described in Section 3, both systems are applied to the complete datasets (both training and test), without candidates pre-filtering.

BIU-DISC uses three base-classifiers ($n = 3$): *SVM*^{perf} (Joachims, 2006), and Naïve Bayes and Logistic Regression from the WEKA package (Witten and Frank, 2005). The first among these is set as our designated classifier $C^{\star}$, which is used for the computation of the document-level features. *SVM*^{perf} is also used for the meta-classifier. For the smoothed entailment score (cf. Section 5.3), we used $b = 0.9$ and $N = 3$. Global information is added by enriching each sentence with the highest-ranking term in the document, according to *tf-idf* scores (cf. Section 5.2), where document frequencies were computed based on about half a million documents from the TIPSTER corpus (Harman, 1992). The set of weights $\vec{\lambda}$ equals $\{2, 1, 4\}$ for title terms, proper names and locations, respectively. All parameters were tuned based on a 10-fold cross-validation on the development set, optimizing the micro-averaged F_1.

The results are presented in Table 1. As can be seen in the table, *BIU-DISC* outperforms *BIU-BL* in every measure, showing the impact of addressing discourse in this setting. To rule out the option that the improvement is simply due to the fact that we use three classifiers for *BIU-DISC* and a single one for *BIU-BL*, we show (*BIU-BL*^{3}) the results when the baseline system is applied in the same meta-classification configuration as *BIU-DISC*, with the same three classifiers. Apparently, without the discourse information this configuration's contribution is limited.

	P (%)	R (%)	F_1 (%)
	By Topic		
BIU-BL	16.54	55.62	25.50
BIU-DISC	22.69	57.96	32.62
All-yes baseline	4.85	100.00	9.25
	By Hypothesis		
BIU-BL	22.87	59.62	33.06
BIU-DISC	27.81	61.97	38.39
All-yes baseline	4.96	100.00	9.46

Table 2: Macro-average results.

As mentioned in Section 5.3, the benefit from the locality features rely directly on the performance of the base classifiers. Hence, considering the low precision scores obtained here, we applied *BIU-DISC* to the data in the meta-classification scheme, but with locality features removed. The results, shown as *BIU-DISC*$_{no-loc}$ in the Table, indicate that indeed performance increases without these features. The last line of the table shows the results obtained by a naïve baseline where all test-set pairs are considered entailing.

For completeness, Table 2 shows the macro-averaged results, when averaged over the topics or over the hypotheses. Although we tuned our system to maximize micro-averaged F_1, these figures comply with the ones shown in Table 1.

Analysis of locality As discussed in Section 5, determining whether a sentence entails a hypothesis should take into account whether adjacent sentences also entail the hypothesis. In the above experiment we were unable to show the contribution of our system's component that attempts to capture this information; on the contrary, the results show it had a negative impact on performance.

Still, we claim that this information can be useful when used within a more accurate system. We try to validate this conjecture by understanding how performance of the locality features varies as the systems becomes more accurate. We do so via the following simulation.

When classifying a certain sentence, the classi-

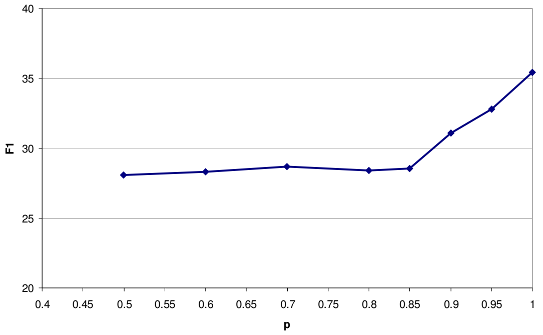


Figure 2: F_1 performance of *BIU-DISC* as a function of the accuracy in classifying adjacent sentences.

fications of its adjacent sentences are given by an *oracle classifier* that provides the correct answer with probability p. The system is applied using two locality features: the *1^{st} sentence entailing title* feature and a close variant of the *smoothed entailment* feature, which calculates the weighted average of adjacent sentences, but disregards the score of the currently evaluated sentence.[3] Thus we supply information about adjacent sentences and test whether overall performance increases with the accuracy of this information.

We performed this experiment for p in a range of [0.5-1.0]. Figure 2 shows the results of this simulation, based on the average F_1 of five runs for each p. Since performance, from a certain point, increases with the accuracy of the oracle classifier, we can conclude that indeed precise information about adjacent sentences improves performance on the current sentence, and that locality is a true phenomenon in the data. We note, however, that performance improves only when accuracy is very high, suggesting the currently limited practical potential of this information, at least in the way locality was represented in this work.

Ablation tests Table 3 presents the results of the ablation tests performed to evaluate the contribution of each component. Based on the result reported in Table 1 and the above discussion, the tests were performed relative to *BIU-DISC*$_{no-loc}$, the optimal configuration. As seen in the table, the removal of each component causes a drop in results. For global information we see a minor difference, which is not surprising considering the conservative approach we took, using a single global term for each sentence. Possibly, this information is also included in the other components, thus proving no marginal contribution relative to them. Under the conditions of an overwhelming majority of negative examples, this is a risky method to use, and should be considered when the ratio of positive examples is higher. For future work, we intend to use this information via classification features (e.g. the coverage obtained with and without global information), rather than the crude addition of the term to the sentence.

Component removed	F_1 (%)	ΔF_1 (%)
Previous sent. features	28.55	3.58
Augmented coref.	26.73	5.40
Global information	31.76	0.37

Table 3: Results of ablation tests relative to *BIU-DISC*$_{no-loc}$. The columns specify the component removed, the micro-averaged F_1 score achieved without it, and the marginal contribution of the component.

Analysis of augmented coreferences We analyzed the performance of the component for augmenting coreference relations relative to the OpenNLP resolver. Recall that our component works on top of the resolver's output and can add or remove coreference relations. As a complete annotation of coreference chains in the dataset is unavailable, we performed the following evaluation. Recall is computed based on the number of identified pairs from a sample of 100 intra-document coreference and bridging relations from the annotated dataset described in (Mirkin et al., 2010). Precision is computed based on 50 pairs sampled from the output of each method, equally distributed over topics. The results, shown in Table 4, indicate the much higher recall obtained by our component at some cost in precision. Although rather simple, the ablation test of this component shows its usefulness. Still, both methods achieve low absolute recall, suggesting the need for more robust tools for this task.

	P (%)	R (%)	F_1 (%)
OpenNLP	74	16	26.3
Augmented coref.	60	28	38.2

Table 4: Performance of coreference methods.

[3] The *second-closest entailment* feature was not used as it considers the oracle's decision for the current sentence, while we wish to use only information about adjacent sentences.

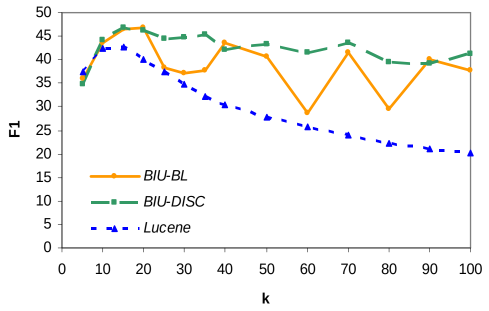


Figure 3: F_1 performance as a function of the number of retrieved candidates.

Candidate retrieval setting As mentioned in Section 3, best performance of RTE systems in the task was obtained when applying a first step of IR-based candidate filtering. We therefore compare the performance of *BIU-DISC* with that of *BIU-BL* under this setting as well.[4] For candidate retrieval we used Lucene, a state of the art search engine[5], in a range of top-k retrieved candidates. The results are shown in Figure 3. For reference, the figure also shows the performance along this range of Lucene as-is, when no further inference is applied to the retrieved candidates.

While *BIU-DISC* does not outperform *BIU-BL* at every point, the area under the curve is clearly larger for *BIU-DISC*. The figure also indicates that *BIU-DISC* is far more robust, maintaining a stable F_1 and enabling a stable tradeoff between recall and precision along the whole range (recall ranges between 42% and 55% for $k \in [15 - 100]$, with corresponding precision range of 51% to 33%).

Finally, Table 5 shows the results of the best systems as determined in our first experiment. We performed a single experiment to compare *BIU-DISC*$_{no-loc}$ and *BIU-BL*^{3} under a candidate retrieval setting, using $k = 20$, where both systems highly perform. We compare these results to the highest score obtained by Lucene, as well as to the two best submissions to the RTE-5 Search task[6]. *BIU-DISC*$_{no-loc}$ outperforms all other methods and its result is significantly better than *BIU-BL*^{3} with $p < 0.01$ according to McNemar's test.

	P (%)	R (%)	F_1 (%)
BIU-DISC$_{no-loc}$	50.77	45.12	47.78
BIU-BL^3	51.68	40.38	45.33
Lucene, top-15	35.93	52.50	42.66
RTE-5 best	40.98	51.38	45.59
RTE-5 second-best	42.94	38.00	40.32

Table 5: Performance of best configurations.

7 Conclusions

While it is generally assumed that discourse interacts with semantic entailment inference, the concrete impacts of discourse on such inference have been hardly explored. This paper presented a first empirical investigation of discourse processing aspects related to entailment. We argue that available discourse processing tools should be substantially improved towards this end, both in terms of the phenomena they address today, namely nominal coreference, and with respect to the covering of additional phenomena, such as bridging anaphora. Our experiments show that even rather simple methods for addressing discourse can have a substantial positive impact on the performance of entailment inference. Concerning the locality phenomenon stemming from discourse coherence, we learned that it does carry potentially useful information, which might become beneficial in the future when better-performing entailment systems become available. Until then, integrating this information with entailment confidence may be useful. Overall, we suggest that entailment systems should extensively incorporate discourse information, while developing sound algorithms for addressing various discourse phenomena, including the ones described in this paper.

Acknowledgements

The authors are thankful to Asher Stern and Ilya Kogan for their help in implementing and evaluating the augmented coreference component, and to Roy Bar-Haim for useful advice concerning this paper. This work was partially supported by the Israel Science Foundation grant 1112/08 and the PASCAL-2 Network of Excellence of the European Community FP7-ICT-2007-1-216886. Jonathan Berant is grateful to the Azrieli Foundation for the award of an Azrieli Fellowship.

[4] This time, for global information, the document's three highest ranking terms were added to each sentence.

[5] http://lucene.apache.org

[6] The best one is an earlier version of this work (Mirkin et al., 2009); the second is MacKinlay and Baldwin's (2009).

References

Bar-Haim, Roy, Jonathan Berant, Ido Dagan, Iddo Greental, Shachar Mirkin, Eyal Shnarch, and Idan Szpektor. 2008. Efficient semantic deduction and approximate matching over compact parse forests. In *Proc. of Text Analysis Conference (TAC)*.

Bar-Haim, Roy, Jonathan Berant, and Ido Dagan. 2009. A compact forest for scalable inference over entailment and paraphrase rules. In *Proc. of EMNLP*.

Bensley, Jeremy and Andrew Hickl. 2008. Unsupervised resource creation for textual inference applications. In *Proc. of LREC*.

Bentivogli, Luisa, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, Medea Lo Leggio, and Bernardo Magnini. 2009a. Considering discourse references in textual entailment annotation. In *Proc. of the 5th International Conference on Generative Approaches to the Lexicon (GL2009)*.

Bentivogli, Luisa, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009b. The fifth PASCAL recognizing textual entailment challenge. In *Proc. of TAC*.

Castillo, Julio J. 2009. Sagan in TAC2009: Using support vector machines in recognizing textual entailment and TE search pilot task. In *Proc. of TAC*.

Clark, Peter and Phil Harrison. 2009. An inference-based approach to recognizing entailment. In *Proc. of TAC*.

Clark, Herbert H. 1975. Bridging. In Schank, R. C. and B. L. Nash-Webber, editors, *Theoretical issues in natural language processing*, pages 169–174. Association of Computing Machinery.

Dagan, Ido, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer.

Dagan, Ido, Bill Dolan, Bernardo Magnini, and Dan Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, pages 15(4):1–17.

Dali, Lorand, Delia Rusu, Blaz Fortuna, Dunja Mladenic, and Marko Grobelnik. 2009. Question answering based on semantic graphs. In *Proc. of the Workshop on Semantic Search (SemSearch 2009)*.

Fellbaum, Christiane, editor. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press.

Finkel, Jenny Rose, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc. of ACL*.

Giampiccolo, Danilo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Harabagiu, Sanda and Andrew Hickl. 2006. Methods for using textual entailment in open-domain question answering. In *Proc. of ACL*.

Harman, Donna. 1992. The DARPA TIPSTER project. *SIGIR Forum*, 26(2):26–28.

Joachims, Thorsten. 2006. Training linear SVMs in linear time. In *Proc. of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*.

Li, Fangtao, Yang Tang, Minlie Huang, and Xiaoyan Zhu. 2009. Answering opinion questions with random walks on graphs. In *Proc. of ACL-IJCNLP*.

Litkowski, Ken. 2009. Overlap analysis in textual entailment recognition. In *Proc. of TAC*.

MacKinlay, Andrew and Timothy Baldwin. 2009. A baseline approach to the RTE5 search pilot. In *Proc. of TAC*.

Mirkin, Shachar, Roy Bar-Haim, Jonathan Berant, Ido Dagan Eyal Shnarch, Asher Stern, and Idan Szpektor. 2009. Addressing discourse and document structure in the RTE search task. In *Proc. of TAC*.

Mirkin, Shachar, Ido Dagan, and Sebastian Padó. 2010. Assessing the role of discourse references in entailment inference. In *Proc. of ACL*.

Nenkova, Ani, Rebecca Passonneau, and Kathleen Mckeown. 2007. The pyramid method: incorporating human content selection variation in summarization evaluation. In *ACM Transactions on Speech and Language Processing*.

Qiu, Long, Min-Yen Kan, and Tat-Seng Chua. 2004. A public reference implementation of the RAP anaphora resolution algorithm. In *Proc. of LREC*.

Romano, Lorenza, Milen Kouylekov, Idan Szpektor, and Ido Dagan. 2006. Investigating a generic paraphrase-based approach for relation extraction. In *Proc. of EACL*.

Witten, Ian H. and Eibe Frank. 2005. *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco.

Evaluating Dependency Representation for Event Extraction

Makoto Miwa[1] **Sampo Pyysalo**[1] **Tadayoshi Hara**[1] **Jun'ichi Tsujii**[1,2,3]
[1]Department of Computer Science, the University of Tokyo
[2]School of Computer Science, University of Manchester
[3]National Center for Text Mining
{mmiwa,smp,harasan,tsujii}@is.s.u-tokyo.ac.jp

Abstract

The detailed analyses of sentence structure provided by parsers have been applied to address several information extraction tasks. In a recent bio-molecular event extraction task, state-of-the-art performance was achieved by systems building specifically on dependency representations of parser output. While intrinsic evaluations have shown significant advances in both general and domain-specific parsing, the question of how these translate into practical advantage is seldom considered. In this paper, we analyze how event extraction performance is affected by parser and dependency representation, further considering the relation between intrinsic evaluation and performance at the extraction task. We find that good intrinsic evaluation results do not always imply good extraction performance, and that the types and structures of different dependency representations have specific advantages and disadvantages for the event extraction task.

1 Introduction

Advanced syntactic parsing methods have been shown to effective for many information extraction tasks. The BioNLP 2009 Shared Task, a recent bio-molecular event extraction task, is one such task: analysis showed that the application of a parser correlated with high rank in the task (Kim et al., 2009). The automatic extraction of bio-molecular events from text is important for a number of advanced domain applications such as pathway construction, and event extraction thus a key task in Biomedical Natural Language Processing (BioNLP).

Methods building feature representations and extraction rules around dependency representations of sentence syntax have been successfully applied to a number of tasks in BioNLP. Several parsers and representations have been applied in high-performing methods both in domain studies in general and in the BioNLP'09 shared task in particular, but no direct comparison of parsers or representations has been performed. Likewise, a number of evaluation of parser outputs against gold standard corpora have been performed in the domain, but the broader implications of the results of such intrinsic evaluations are rarely considered. The BioNLP'09 shared task involved documents contained also in the GENIA treebank (Tateisi et al., 2005), creating an opportunity for direct study of intrinsic and task-oriented evaluation results. As the treebank can be converted into various dependency formats using existing format conversion methods, evaluation can further be extended to cover the effects of different representations.

In this this paper, we consider three types of dependency representation and six parsers, evaluating their performance from two different aspects: dependency-based intrinsic evaluation, and effectiveness for bio-molecular event extraction with a state-of-the-art event extraction system. Comparison of intrinsic and task-oriented evaluation re-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 779–787,
Beijing, August 2010

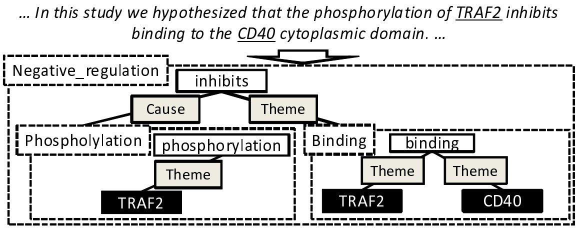

… In this study we hypothesized that the phosphorylation of TRAF2 inhibits binding to the CD40 cytoplasmic domain. …
Negative_regulation
inhibits
Cause
Theme
Phospholylation
phosphorylation
Theme
TRAF2
Binding
binding
Theme
Theme
TRAF2
CD40

Figure 1: Event Extraction.

sults shows that performance against gold standard annotations is not always correlated with event extraction performance. We further find that the dependency types and overall structures employed by the different dependency representations have specific advantages and disadvantages for the event extraction task.

2 Bio-molecular Event Extraction

In this study, we adopt the event extraction task defined in the BioNLP 2009 Shared Task (Kim et al., 2009) as a model information extraction task. Figure 1 shows an example illustrating the task of event extraction from a sentence. The shared task provided common and consistent task definitions, data sets for training and evaluation, and evaluation criteria. The shared task defined five simple events (Gene expression, Transcription, Protein catabolism, Phosphorylation, and Localization) that take one core argument, a multi-participant binding event (Binding), and three regulation events (Regulation, Positive regulation, and Negative regulation) used to capture both biological regulation and general causation. The participants of simple and Binding events were specified to be of the general Protein type, while regulation-type events could also take other events as arguments, creating complex event structures.

We consider two subtasks, Task 1 and Task 2, out of the three defined in the shared task. Task 1 focuses on core event extraction, and Task 2 involves augmenting extracted events with secondary arguments (Kim et al., 2009). Events are represented with a textual trigger, type, and arguments, where the trigger is a span of text that states the event in text. In Task 1 the event arguments that need to be extracted are restricted to the core Theme and Cause roles, with secondary arguments corresponding to locations and sites considered in Task 2.

2.1 Event Extraction System

For evaluation, we apply the system of Miwa et al. (2010b). The system was originally developed for finding core events (Task 1) using the native output of the Enju and GDep parsers. The system consists of three supervised classification-based modules: a trigger detector, an event edge detector, and a complex event detector. The trigger detector classifies each word into the appropriate event types, the event edge detector classifies each edge between an event and a candidate participant into an argument type, and the complex event detector classifies event candidates constructed by all edge combinations, deciding between event and non-event. The system uses one-vs-all support vector machines (SVMs) for classification.

The system operates on one sentence at a time, building features for classification based on the syntactic analyses for the sentence provided by the two parsers as well as the sequence of the words in the sentence, including the target candidate. The features include the constituents/words around entities (triggers and proteins), the dependencies, and the shortest paths among the entities. The feature generation is format-independent regarding the shared properties of different formats, but makes use also of format-specific information when available for extracting features, including the dependency tags, word-related information (e.g. a lexical entry in Enju format), and the constituents and their head information.

We apply here a variant of the base system incorporating a number of modifications. The applied system performs feature selection removing two classes of features that were found not to be beneficial for extraction performance, and applies a refinement of the trigger expressions of events. The system is further extended to find also secondary arguments (Task 2). For a detailed description of these improvements, we refer to Miwa et al. (2010a).

3 Parsers and Representations

Six publicly available parsers and three dependency formats are considered in this paper. The

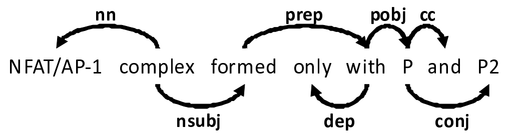


Figure 2: Stanford basic dependency tree

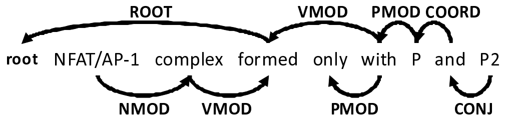


Figure 3: CoNLL-X dependency tree

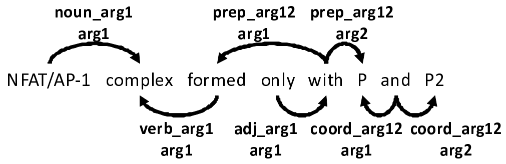


Figure 4: Predicate Argument Structure

parsers are GDep (Sagae and Tsujii, 2007), the Bikel parser (Bikel) (Bikel, 2004), the Stanford parser with two probabilistic context-free grammar (PCFG) models[1] (Wall Street Journal (WSJ) model (Stanford WSJ) and "augmented English" model (Stanford eng)) (Klein and Manning, 2003), the Charniak-Johnson reranking parser, using David McClosky's self-trained biomedical parsing model (MC) (McClosky, 2009), the C&C CCG parser, adapted to biomedical text (C&C) (Rimell and Clark, 2009), and the Enju parser with the GENIA model (Miyao et al., 2009). The formats are Stanford Dependencies (SD) (Figure 2), the CoNLL-X dependency format (CoNLL) (Figure 3) and the predicate-argument structure (PAS) format used by Enju (Figure 4). With the exception of Stanford and Enju, the analyses of these parsers were provided by the BioNLP 2009 Shared Task organizers.

The six parsers operate in a number of different frameworks, reflected in their analyses. GDep is a native dependency parser that produces CoNLL dependency trees, with dependency types similar to those of CoNLL 2007. Bikel, Stanford, and MC are phrase-structure parsers trained on Penn Treebank format (PTB) style treebanks, and they produce PTB trees. C&C is a deep parser based on Combinatory Categorial Grammar (CCG), and its native output is in a CCG-specific format. The output of C&C can be converted into SD by a rule-based conversion script (Rimell and Clark, 2009). Enju is deep parser based on Head-driven Phrase Structure Grammar (HPSG) and produces a format containing predicate argument structures along with a phrase structure tree in Enju format, which can be converted into PTB format (Miyao et al., 2009).

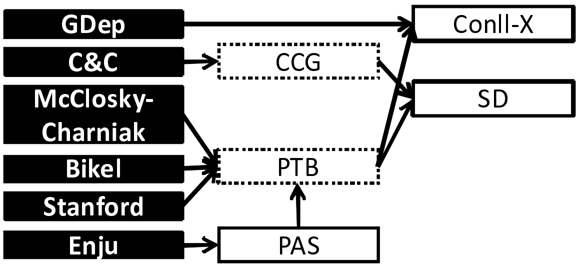


Figure 5: Format conversion dependencies in six parsers. Formats adopted for the evaluation are shown in solid boxes. SD: Stanford Dependency format, CCG: Combinatory Categorial Grammar output format, PTB: Penn Treebank format, and PAS: Predicate Argument Structure in Enju format.

For direct comparison and for the study of contribution of the formats in which the six parsers output their analyses to task performance, we apply a number of conversions between the outputs, shown in Figure 5. The Enju PAS output is converted into PTB using the method introduced by (Miyao et al., 2009). SD is generated from PTB by the Stanford tools (de Marneffe et al., 2006), and CoNLL generated from PTB by using Treebank Converter (Johansson and Nugues, 2007). With the exception of GDep, all CoNLL outputs are generated by the conversion and thus share dependency types. We note that all of these conversions can introduce some errors in the conversion process.

[1] Experiments showed no benefit from using the lexicalized models with the Stanford parser.

4 Evaluation Setting

4.1 Event Extraction Evaluation

Event extraction performance is evaluated using the evaluation script provided by the BioNLP'09 shared task organizers for the development data set, and the online evaluation system of the task for the test data set[2] . Results are reported under the official evaluation criterion of the task, i.e. the "Approximate Span Matching/Approximate Recursive Matching" criterion.

The event extraction system described in Section 2.1 is used with the default settings given in (Miwa et al., 2010b). The C-values of SVMs are set to 1.0, but the positive and negative examples are balanced by placing more weight on the positive examples. The examples predicted with confidence greater than 0.5, as well as the examples with the most confident labels, are extracted. Task 1 and Task 2 are solved at once for the evaluation.

Some of the parse results do not include word base forms or part-of-speech (POS) tags, which are required by the event extraction system. To apply these parsers, the GENIA Tagger (Tsuruoka et al., 2005) output is adopted to add this information to the results.

4.2 Dependency Representation Evaluation

The parsers are evaluated with precision, recall, and F-score for each dependency type. We note that the parsers may produce more fine-grained word segmentations than that of the gold standard: for example, two words "p70(S6)-kinase activation" in the gold standard tree (Figure 6 (a)) is segmented into five words by Enju (Figure 6 (b)). In the evaluation the word segmentations in the gold tree are used, and dependency transfer and word-based normalization are performed to match parser outputs to these. Dependencies related to the segmentations are transferred to the enclosing word as follows. If one word is segmented into several segments by a parser, all the dependencies between the segments are removed (Figure 6 (c)) and the dependency between another word and the segments is converted into the dependency between the two words (Figure 6 (d)).

The parser outputs in SD and CoNLL can be assumed to be trees, so each node in the tree have only one parent node. However, in the converted tree nodes can have more than one parent. We cannot simply apply accuracy, or (un)labeled attachment score[3]. Word-based normalization is performed to avoid negative impact by the word segmentations by parsers. When (a) and (d) in Figure 6 are compared, the counts of correct relations will be 1.0 (0.5 for upper NMOD and 0.5 for lower NMOD in Figure 6 (d)) for the parser (precision), and the counts of correct relations will be 1.0 (for NMOD in Figure 6 (a)) for the gold (recall). This F-score is a good approximation of accuracy.

4.3 GENIA treebank processing

For comparison and evaluation, the texts in the GENIA treebank (Tateisi et al., 2005) are converted to the various formats as follows. To create PAS, the treebank is converted with Enju, and for trees that fail conversion, parse results are used instead. The GENIA treebank is also converted into PTB[4], and then converted into SD and CoNLL as described in Section 3. While based on manually annotated gold data, the converted treebanks are not always correct due to conversion errors.

5 Evaluation

This section presents evaluation results. Intrinsic evaluation is first performed in Section 5.1. Section 5.2 considers the effect of different SD variants. Section 5.3 presents the results of experiments with different parsers. Section 5.4 shows the performance of different parsers. Finally, the performance of the event extraction system is discussed in context of other proposed methods for the task in Section 5.5.

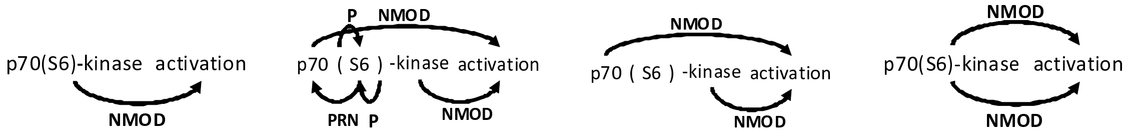
5.1 Intrinsic Evaluation

We initially briefly consider the results of an intrinsic evaluation comparing parser outputs to reference data automatically derived from the gold standard treebank. Table 1 shows results for the parsers whose outputs could be converted into the

[2] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/

[3] http://nextens.uvt.nl/~conll/

[4] http://categorizer.tmit.bme.hu/~illes/genia_ptb/


p70(S6)-kinase activation
NMOD
P NMOD
p70 (S6) -kinase activation
PRN P
NMOD
NMOD
p70 (S6) -kinase activation
NMOD
NMOD
p70(S6)-kinase activation
NMOD

(a) Gold Word Segmentations

(b) Parser Word Segmentations

(c) Inner Dependency Removal

(d) Dependency Transfer

Figure 6: Example of Word Segmentations of the words by gold and Enju and Dependency Transfer.

	Typed						Untyped					
	SD			CoNLL			SD			CoNLL		
	P	R	F	P	R	F	P	R	F	P	R	F
Bikel	70.31	70.37	70.34	77.81	77.56	77.69	80.54	80.60	80.57	82.43	82.18	82.31
SP WSJ	74.11	73.94	74.03	81.41	81.47	81.44	81.36	81.16	81.26	84.05	84.05	84.05
SP eng	79.08	78.89	78.98	84.92	84.82	84.87	84.16	83.96	84.06	86.54	86.47	86.51
C&C	80.31	78.04	79.16		-		84.91	82.28	83.57		-	
MC	79.56	79.63	79.60	88.13	87.87	88.00	87.43	87.50	87.47	89.81	89.42	89.62
Enju	85.59	85.62	85.60	88.59	89.51	89.05	88.28	88.30	88.29	90.24	90.77	90.50

Table 1: Comparison of precision, recall, and F-score results with five parsers (two models for Stanford) in two different formats on the development data set (SP abbreviates for Stanford Parser). Results shown separately for evaluation including dependency types and one eliminating them. Parser/model combinations above the line do not use in-domain data, others do.

SD and CoNLL dependency representations using the Stanford tools and Treebank Converter, respectively. For Stanford, both the Penn Treebank WSJ section and "augmented English" (eng) models were tested; the latter includes biomedical domain data. The Enju results for PAS are 91.48 with types and 93.39 without in F-score. GDep not shown as its output is not compatible with that of Treebank Converter.

Despite numerical differences, the two representations and two criteria (typed/untyped) all produce largely the same ranking of the parsers.[5] The evaluations also largely agree on the magnitude of the reduction in error afforded through the use of in-domain training data for the Stanford parser, with all estimates falling in the 15-19% range. Similarly, all show substantial differences between the parsers, indicating e.g. that the error rate of Enju is 50% or less of that of Bikel.

These results serve as a reference point for extrinsic evaluation results. However, it should be noted that as the parsers make use of annotated domain training data to different extents, this evaluation does not provide a sound basis for direct comparison of the parsers themselves.

	BD	CD	CDP	CTD
Task 1	**55.60**	54.35	54.59	54.42
Task 2	**53.94**	52.65	52.88	52.76

Table 2: Comparison of the F-score results with different SD variants on the development data set with the MC parser. The best score in each task is shown in bold.

5.2 Stanford Dependency Setting

SD have four different variants: basic dependencies (BD), collapsed dependencies (CD), collapsed dependencies with propagation of conjunct dependencies (CDP), and collapsed tree dependencies (CTD) (de Marneffe and Manning, 2008). Except for BD, these variants do not necessarily connect all the words in the sentence, and CD and CDP do not necessarily form a tree structure. Table 2 shows the comparison results with the MC parser. Dependencies are generalized by removing expressions after "_" of the dependencies (e.g.

[5] One larger divergence is between typed and untyped SD results for MC. Analysis suggest one cause is frequent errors in tagging hyphenated noun-modifiers such as *NF-kappaB* as adjectives.

"_with" in prep_with) for better performance. We find that basic dependencies give the best performance to event extraction, with little difference between the other variants. This result is surprising, as variants other than basic have features such as the resolution of conjunctions that are specifically designed for practical applications. However, basic dependencies were found to consistently provide best performance also for the other parsers[6]. Thus, in the following evaluation, the basic dependencies are adopted for all SD results.

5.3 Parser Comparison on Event Extraction

Results with different parsers and different formats on the development data set are summarized in Table 3. Baseline results are produced by removing dependency information from the parse results. The baseline results differ between the representations as the word base forms and POS tags produced by the GENIA tagger for use with SD and CoNLL are different from PAS, and because head word information in the Enju format is used. The evaluation finds best results for both tasks with Enju, using its native output format. However, as discussed in Section 2.1, the treatment of PAS and the other two formats are slightly different, this result does not necessarily indicate that PAS is the best alternative for event extraction.

The Bikel and Stanford WSJ parsers, lacking models adapted to the biomedical domain, performs mostly worse than the other parsers. The other parsers, even though trained on the treebank, do not provide performance as high as that for using the GENIA treebank, but, with the exception of Stanford eng with CoNLL, results with the parsers are only slightly worse than results with the treebank. The results with the data derived from the GENIA treebank can be considered as upper bounds for the parsers and formats at the task, although conversion errors are expected to lower these bounds to some extent. The results suggest that there is relative little remaining benefit to be gained from improving parser performance.

5.4 Effects of Dependency Representation

Intrinsic evaluation results (Section 5.1) cannot be used directly for comparing the parsers, since some of the parsers contain models trained on the GENIA treebank. To investigate the effects of the evaluation results to the event extraction, we performed event extraction with eliminating the dependency types. Table 4 summarizes the results with the dependency structures (without the dependency types) on the development data set. Interestingly, we find the performance increases in Bikel and Stanford by eliminating the dependency types. This implies that the inaccurate dependency types shown in Table 1 confused the event extraction system. SD and PAS drops more than CoNLL, and Enju with CoNLL structures perform best in total when the dependency types are removed. This result shows that the formats have their own strengths in finding events, and CoNLL structure with SD or PAS types can be a good representation for the event extraction.

By comparing Table 3, Table 1, and Table 4, we found that the better dependency performance does not always produce better event extraction performance especially when the difference of the dependency performance is small. MC and Enju results show that performance in dependency is important for event extraction. SD can be better than CoNLL for the event extraction (shown with the gold treebank data in Table 3), but the types and relations of CoNLL were well predicted, and MC and Enju performed better for CoNLL than for SD in total.

5.5 Performance of Event Extraction System

Several systems are compared by the extraction performance on the shared task test data in Table 5. GDep and Enju with PAS are used for the evaluation, which is the same evaluation setting with the original system by Miwa et al. (2010b). The performance of the best systems in the original shared task is shown for reference ((Björne et al., 2009) in Task 1 and (Riedel et al., 2009) in Task 2). The event extraction system performs significantly better than the best systems in the shared task, further outperforming the original system. This shows that the comparison of the parsers is performed with a state-of-the-art sys-

[6] Collapsed tree dependencies are not evaluated on the C&C parser since the conversion is not provided.

	Task 1			Task 2		
	SD	CoNLL	PAS	SD	CoNLL	PAS
Baseline	51.05	-	50.42	49.17	-	48.88
Bikel	53.29	53.22	-	51.40	51.27	-
Stanford WSJ	53.51	54.38	-	52.02	52.04	-
Stanford eng	55.02	53.66	-	53.41	52.74	-
GDep	-	55.70	-	-	54.37	-
MC	55.60	56.01	-	53.94	54.51	-
C&C	56.09	-	-	54.27	-	-
Enju	55.48	55.74	**56.57**	54.06	54.37	**55.31**
GENIA	56.34	56.09	57.94	55.04	54.57	56.40

Table 3: Comparison of F-score results with six parsers in three different formats on the development data set. Results without dependency information are shown as baselines. The results with the GENIA treebank (converted into PTB and PAS) are shown for comparison. The best score in each task is shown in bold, and the best score in each task and format is underlined.

	Task 1			Task 2		
	SD	CoNLL	PAS	SD	CoNLL	PAS
Bikel	53.41 (+0.12)	53.92 (+0.70)	-	51.59 (+0.19)	52.21 (+0.94)	-
Stanford WSJ	53.03 (-0.48)	54.52 (+0.14)	-	51.43 (-0.59)	52.60 (-0.14)	-
Stanford eng	54.48 (-0.54)	54.02 (+0.36)	-	52.88 (-0.53)	52.28 (+0.24)	-
GDep	-	54.97 (-0.73)	-	-	53.71 (-0.66)	-
MC	54.22 (-1.38)	55.24 (-0.77)	-	52.73 (-1.21)	53.42 (-1.09)	-
C&C	54.64 (-1.45)	-	-	52.98 (-1.29)	-	-
Enju	53.74 (-1.74)	**55.66** (-0.08)	55.23 (-1.34)	52.29 (-1.77)	**53.97** (-0.40)	53.69 (-1.62)
GENIA	55.79 (-0.55)	55.64 (-0.45)	56.42 (-1.52)	54.17 (-0.87)	53.83 (-0.74)	55.34 (-1.06)

Table 4: Comparison of F-score results with six parsers in three different dependency structures (without the dependency types) on the development data set. The changes from Table 3 are shown.

	Simple	Binding	Regulation	All
	Task 1			
Ours	**66.84** / **78.22** / **72.08**	48.70 / 52.65 / 50.60	**38.48** / **55.06** / **45.30**	**50.13** / **64.16** / **56.28**
Miwa	65.31 / 76.44 / 70.44	**52.16** / **53.08** / **52.62**	35.93 / 46.66 / 40.60	48.62 / 58.96 / 53.29
Björne	64.21 / 77.45 / 70.21	40.06 / 49.82 / 44.41	35.63 / 45.87 / 40.11	46.73 / 58.48 / 51.95
Riedel	N/A	23.05 / 48.19 / 31.19	26.32 / 41.81 / 32.30	36.90 / 55.59 / 44.35
Baseline	62.94 / 68.38 / 65.55	48.41 / 34.50 / 40.29	29.40 / 40.00 / 33.89	43.93 / 50.11 / 46.82
	Task 2			
Ours	**65.43** / **75.56** / **70.13**	**46.42** / **50.31** / **48.29**	**38.18** / **54.45** / **44.89**	**49.20** / **62.57** / **55.09**
Riedel	N/A	22.35 / 46.99 / 30.29	25.75 / 40.75 / 31.56	35.86 / 54.08 / 43.12
Baseline	60.88 / 63.78 / 62.30	44.99 / 31.78 / 37.25	29.07 / 39.52 / 33.50	42.62 / 47.84 / 45.08

Table 5: Comparison of Recall / Precision / F-score results on the test data set. Results on simple, binding, regulation, and all events are shown. GDep and Enju with PAS are used. Results by Miwa et al. (2010b), Björne et al. (2009), Riedel et al. (2009), and Baseline for Task 1 and Task 2 are shown for comparison. Baseline results are produced by removing dependency information from the parse results of GDep and Enju. The best score in each result is shown in bold.

tem.

6 Related Work

Many approaches for parser comparison have been proposed, and most comparisons have used gold treebanks with intermediate formats (Clegg and Shepherd, 2007; Pyysalo et al., 2007). Parser comparison has also been proposed on specific tasks such as unbounded dependencies (Rimell et al., 2009) and textual entailment (Önder Eker, 2009)[7]. Among them, application-oriented parser comparison across several formats was first introduced by Miyao et al. (2009), who compared eight parsers and five formats for the protein-protein interaction (PPI) extraction task. PPI extraction, the

[7] http://pete.yuret.com/

recognition of binary relations of between proteins, is one of the most basic information extraction tasks in the BioNLP field. Our findings do not conflict with those of Miyao et al. Event extraction can be viewed as an additional extrinsic evaluation task for syntactic parsers, providing more reliable and evaluation and a broader perspective into parser performance. An additional advantage of application-oriented evaluation on BioNLP shared task data is the availability of a manually annotated gold standard treebank, the GENIA treebank, that covers the same set of abstracts as the task data. This allows the gold treebank to be considered as an evaluation standard, in addition to comparison of performance in the primary task.

7 Conclusion

We compared six parsers and three formats on a bio-molecular event extraction task with a state-of-the-art event extraction system from two different aspects: dependency-based intrinsic evaluation and task-based extrinsic evaluation. The specific task considered was the BioNLP shared task, allowing the use of the GENIA treebank as a gold standard parse reference. Five of the six considered parsers were applied using biomedical models trained on the GENIA treebank, and they were found to produce similar performance. The comparison of the parsers from two aspects showed slightly different results, and and the dependency representations have advantages and disadvantages for the event extraction task.

The contributions of this paper are 1) the comparison of intrinsic and extrinsic evaluation on several commonly used parsers with a state-of-the-art system, and 2) demonstration of the limitation and possibility of the parser and system improvement on the task. One limitation of this study is that the comparison between the parsers is not perfect, as the parsers are used with the provided models, the format conversions miss some information from the original formats, and results with different formats depend on the ability of the event extraction system to take advantage of their strengths. To maximize comparability, the system was designed to extract features identically from similar parts of the dependency-based formats, further adding information provided by other formats, such as the lexical entries of the Enju format, from external resources. The results of this paper are expected to be useful as a guide not only for parser selection for biomedical information extraction but also for the development of event extraction systems.

The comparison in the present evaluation is limited to the dependency representation. As future work, it would be informative to extend the comparison to other syntactic representation, such as the PTB format. Finally, the evaluation showed that the system fails to recover approximately 40% of events even when provided with manually annotated treebank data, showing that other methods and resources need to be adopted to further improve bio-molecular event extraction systems. Such improvement is left as future work.

Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan), Genome Network Project (MEXT, Japan), and Scientific Research (C) (General) (MEXT, Japan).

References

Bikel, Daniel M. 2004. A distributional analysis of a lexicalized statistical parsing model. In *In EMNLP*, pages 182–189.

Björne, Jari, Juho Heimonen, Filip Ginter, Antti Airola, Tapio Pahikkala, and Tapio Salakoski. 2009. Extracting complex biological events with rich graph-based feature sets. In *Proceedings of the BioNLP'09 Shared Task on Event Extraction*, pages 10–18.

Clegg, Andrew B. and Adrian J. Shepherd. 2007. Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC Bioinformatics*, 8.

de Marneffe, Marie-Catherine and Christopher D. Manning. 2008. Stanford typed dependencies manual. Technical report, September.

de Marneffe, Marie-Catherine, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the IEEE / ACL 2006 Workshop on Spoken Language Technology*.

Johansson, Richard and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, Tartu, Estonia, May 25-26.

Kim, Jin-Dong, Tomoko Ohta, Sampo Pyysalo, Yoshinobu Kano, and Jun'ichi Tsujii. 2009. Overview of bionlp'09 shared task on event extraction. In *BioNLP '09: Proceedings of the Workshop on BioNLP*, pages 1–9.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 423–430, Morristown, NJ, USA. Association for Computational Linguistics.

McClosky, David. 2009. *Any Domain Parsing: Automatic Domain Adaptation for Natural Language Parsing*. Ph.D. thesis, Department of Computer Science, Brown University.

Miwa, Makoto, Sampo Pyysalo, Tadayoshi Hara, and Jun'ichi Tsujii. 2010a. A comparative study of syntactic parsers for event extraction. In *BioNLP2010: Proceedings of the Workshop on BioNLP*, Uppsala, Sweden, July.

Miwa, Makoto, Rune Sætre, Jin-Dong Kim, and Jun'ichi Tsujii. 2010b. Event extraction with complex event classification using rich features. *Journal of Bioinformatics and Computational Biology (JBCB)*, 8(1):131–146, February.

Miyao, Yusuke, Kenji Sagae, Rune Sætre, Takuya Matsuzaki, and Jun ichi Tsujii. 2009. Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics*, 25(3):394–400.

Önder Eker. 2009. Parser evaluation using textual entailments. Master's thesis, Boğaziçi Üniversitesi, August.

Pyysalo, Sampo, Filip Ginter, Veronika Laippala, Katri Haverinen, Juho Heimonen, and Tapio Salakoski. 2007. On the unification of syntactic annotations under the stanford dependency scheme: A case study on bioinfer and genia. In *Biological, translational, and clinical language processing*, pages 25–32, Prague, Czech Republic, June. Association for Computational Linguistics.

Riedel, Sebastian, Hong-Woo Chun, Toshihisa Takagi, and Jun'ichi Tsujii. 2009. A markov logic approach to bio-molecular event extraction. In *BioNLP '09: Proceedings of the Workshop on BioNLP*, pages 41–49, Morristown, NJ, USA. Association for Computational Linguistics.

Rimell, Laura and Stephen Clark. 2009. Porting a lexicalized-grammar parser to the biomedical domain. *J. of Biomedical Informatics*, 42(5):852–865.

Rimell, Laura, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 813–821, Singapore, August. Association for Computational Linguistics.

Sagae, Kenji and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *EMNLP-CoNLL 2007*.

Tateisi, Yuka, Akane Yakushiji, Tomoko Ohta, and Junfichi Tsujii. 2005. Syntax annotation for the genia corpus. In *Proceedings of the IJCNLP 2005, Companion volume*, pages 222–227, Jeju Island, Korea, October.

Tsuruoka, Yoshimasa, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. 2005. Developing a robust part-of-speech tagger for biomedical text. In Bozanis, Panayiotis and Elias N. Houstis, editors, *Panhellenic Conference on Informatics*, volume 3746 of *Lecture Notes in Computer Science*, pages 382–392. Springer.

Entity-Focused Sentence Simplification for Relation Extraction

Makoto Miwa[1] **Rune Sætre**[1] **Yusuke Miyao**[2] **Jun'ichi Tsujii**[1,3,4]
[1]Department of Computer Science, The University of Tokyo
[2]National Institute of Informatics
[3]School of Computer Science, University of Manchester
[4]National Center for Text Mining
mmiwa@is.s.u-tokyo.ac.jp,rune.saetre@is.s.u-tokyo.ac.jp,
yusuke@nii.ac.jp,tsujii@is.s.u-tokyo.ac.jp

Abstract

Relations between entities in text have been widely researched in the natural language processing and information-extraction communities. The region connecting a pair of entities (in a parsed sentence) is often used to construct kernels or feature vectors that can recognize and extract interesting relations. Such regions are useful, but they can also incorporate unnecessary distracting information. In this paper, we propose a rule-based method to remove the information that is unnecessary for relation extraction. Protein–protein interaction (PPI) is used as an example relation extraction problem. A dozen simple rules are defined on output from a deep parser. Each rule specifically examines the entities in one target interaction pair. These simple rules were tested using several PPI corpora. The PPI extraction performance was improved on all the PPI corpora.

1 Introduction

Relation extraction (RE) is the task of finding a relevant semantic relation between two given target entities in a sentence (Sarawagi, 2008). Some example relation types are person–organization relations (Doddington et al., 2004), protein–protein interactions (PPI), and disease–gene associations (DGA) (Chun et al., 2006). Among the possible RE tasks, we chose the PPI extraction problem. PPI extraction is a major RE task; around 10 corpora have been published for training and evaluation of PPI extraction systems.

Recently, machine-learning methods, boosted by NLP techniques, have proved to be effective for RE. These methods are usually intended to highlight or select the relation-related regions in parsed sentences using feature vectors or kernels. The shortest paths between a pair of entities (Bunescu and Mooney, 2005) or pair-enclosed trees (Zhang et al., 2006) are widely used as focus regions. These regions are useful, but they can include unnecessary sub-paths such as appositions, which cause noisy features.

In this paper, we propose a method to remove information that is deemed unnecessary for RE. Instead of selecting the whole region between a target pair, the target sentence is simplified into simpler, pair-related, sentences using general, task-independent, rules. By addressing particularly the target entities, the rules do not affect important relation-related expressions between the target entities. We show how rules of two groups can be easily defined using the analytical capability of a deep parser with specific examination of the target entities. Rules of the first group can replace a sentence with a simpler sentence, still including the two target entities. The other group of rules can replace a large region (phrase) representing one target entity, with just a simple mention of that target entity. With only a dozen simple rules, we show that we can solve several simple well-known problems in RE, and that we can improve the performance of RE on all corpora in our PPI test-set.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 788–796,
Beijing, August 2010

2 Related Works

The general paths, such as the shortest path or pair-enclosed trees (Section 1), can only cover a part of the necessary information for relation extraction. Recent machine-learning methods specifically examine how to extract the missing information without adding too much noise. To find more representative regions, some information from outside the original regions must be included. Several tree kernels have been proposed to extract such regions from the parse structure (Zhang et al., 2006). Also the graph kernel method emphasizes internal paths without ignoring outside information (Airola et al., 2008). Composite kernels have been used to combine original information with outside information (Zhang et al., 2006; Miwa et al., 2009).

The approaches described above are useful, but they can include unnecessary information that distracts learning. Jonnalagadda and Gonzalez (2009) applied bioSimplify to relation extraction. BioSimplify is developed to improve their link grammar parser by simplifying the target sentence in a general manner, so their method might remove important information for a given target relation. For example, they might accidentally simplify a noun phrase that is needed to extract the relation. Still, they improved overall PPI extraction recall using such simplifications.

To remove unnecessary information from a sentence, some works have addressed sentence simplification by iteratively removing unnecessary phrases. Most of this work is not task-specific; it is intended to compress all information in a target sentence into a few words (Dorr et al., 2003; Vanderwende et al., 2007). Among them, Vickrey and Koller (2008) applied sentence simplification to semantic role labeling. With retaining all arguments of a verb, Vickrey simplified the sentence by removing some information outside of the verb and arguments.

3 Entity-Focused Sentence Simplification

We simplify a target sentence using simple rules applicable to the output of a deep parser called Mogura (Matsuzaki et al., 2007), to remove noisy information for relation extraction. Our method relies on the deep parser; the rules depend on the Head-driven Phrase Structure Grammar (HPSG) used by Mogura, and all the rules are written for the parser Enju XML output format. The deep parser can produce deep syntactic and semantic information, so we can define generally applicable comprehensive rules on HPSG with specific examination of the entities.

For sentence simplification in relation extraction, the meaning of the target sentence itself is less important than maintaining the truth-value of the relation (interact or not). For that purpose, we define rules of two groups: clause-selection rules and entity-phrase rules. A clause-selection rule constructs a simpler sentence (still including both target entities) by removing noisy information before and after the relevant clause. An entity-phrase rule simplifies an entity-containing region without changing the truth-value of the relation. By addressing the target entities particularly, we can define rules for many applications, and we can simplify target sentences with less danger of losing relation-related mentions. The rules are summarized in Table 1.

Our method is different from the sentence simplification in other systems (ref. Section 2). First, our method relies on the parser, while bioSimplify by Jonnalagadda and Gonzalez (2009) is developed for the improvement of their parser. Second, our method tries to keep only the relation-related regions, unlike other general systems including bioSimplify which tried to keep all information in a sentence. Third, our entity-phrase rules modify only the entity-containing phrases, while Vickrey and Koller (2008) tries to remove all information outside of the target verb and arguments.

3.1 Clause-selection Rules

In compound or complex sentences, it is natural to assume that one clause includes both the target entities and the relation-related information. It can also be assumed that the remaining sentence parts, outside the clause, contain less related (or noisy) information. The clause-selection rules simplify a sentence by retaining only the clause that includes the target entities (and by discarding the remainder of the sentence). We define three types of

Rule Group	Rule Type	#	Example (original → simplified)
Clause Selection	Sentence Clause	1	We show that A interacts with B. → A interacts with B.
	Relative Clause	2	... A that interacts with B. → A interacts with B.
	Copula	1	A is a protein that interacts with B. → A interacts with B.
Entity Phrase	Apposition	2	a protein, A → A
	Exemplification	4	proteins, such as A → A
	Parentheses	2	a protein (A) → A
	Coordination	3	protein and A → A

Table 1: Rules for Sentence Simplification. (# is the rule count. A and B are the target entities.)

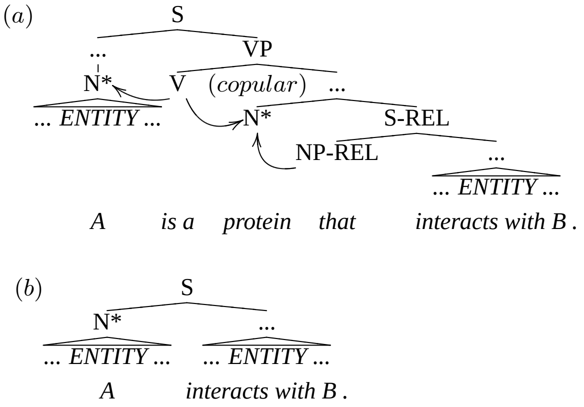


Figure 1: Copula Rule. (a) is simplified to (b). The arrows represent predicate–argument relations.

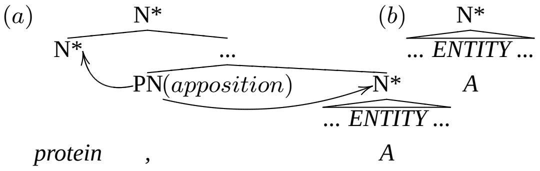


Figure 2: Apposition Rule.

clause-selection rules for sentence clauses, relative clauses, and copula. The *sentence clause rule* finds the (smallest) clause that includes both target entities. It then replaces the original sentence with the clause. The *relative clause rules* construct a simple sentence from a relative clause and the antecedent. If this simple sentence includes the target entities, it is used instead of the original sentence. We define two rules for the case where the antecedent is the subject of the relative clause. One rule is used when the relative clause includes both the target entities. The other rule is used when the antecedent includes one target entity and the relative clause includes the other target entity. The *copula rule* is for sentences that include copular verbs (e.g. be, is, become, etc). The rule constructs a simple sentence from a relative clause with the subject of the copular verb as the antecedent subject of the clause. The rule replaces the target sentence with the constructed sentence, if the relative clause includes one target entity and the subject of a copular verb includes the other target entity, as shown in Figure 1.

3.2 Entity-phrase Rules

Even the simple clauses (or paths between two target entities) include redundant or noisy expressions that can distract relation extraction. Some of these expressions are related to the target entities, but because they do not affect the truth-value of the relation, they can be deleted to make the path simple and clear. The target problem affects which expressions can be removed. We define four types of rules for appositions, exemplifications, parentheses, and coordinations. Two *apposition rules* are defined to select the correct element from an appositional expression. One element modifies or defines the other element in apposition, but the two elements represent the same information from the viewpoint of PPI. If the target entity is in one of these elements, removing the other element does not affect the truth-value of the interaction. Many of these apposition expressions are identified by the deep parser. The rule to select the last element is presented in Figure 2. Four *exemplification rules* are defined for the two major types of expressions using the phrases “including” or “such as”. Exemplification is represented by hyponymy or hypernymy. As for appositions, the truth-value of the interaction does not change whether we use the specific mention or the hyperclass that the mention represents. Two *parentheses rules* are defined. Parentheses are useful for synonyms, hyponyms, or hypernyms (ref. the two

```
 1: $S \leftarrow$ input sentence
 2: **repeat**
 3:     reset rules {apply all the rules again}
 4:     $P \leftarrow$ parse $S$
 5:     **repeat**
 6:         $r \leftarrow$ next rule {null if no more rules}
 7:         **if** $r$ is applicable to $P$ **then**
 8:             $P \leftarrow$ apply $r$ to $P$
 9:             $S \leftarrow$ sentence extracted from $P$
10:             break (Goto 3)
11:         **end if**
12:     **until** $r$ is null
13: **until** $r$ is null
14: **return** $S$
```

Figure 3: Pseudo-code for sentence simplification.

former rules). Three *coordination rules* are defined. Removing other phrases from coordinated expressions that include a target entity does not affect the truth-value of the target relation. Two rules are defined for simple coordination between two phrases (e.g. select left or right phrase), and one rule is defined to (recursively) remove one element from lists of more than two coordinated phrases (while maintaining the coordinating conjunction, e.g. “and”).

3.3 Sentence Simplification

To simplify a sentence, we apply rules repeatedly until no more applications are possible as presented in Figure 3. After one application of one rule, the simplified sentence is re-parsed before attempting to apply all the rules again. This is because we require a consistent parse tree as a starting point for additional applications of the rules, and because a parser can produce more reliable output for a partly simplified sentence than for the original sentence. Using this method, we can also backtrack and seek out conversion errors by examining the cascade of partly simplified sentences.

4 Evaluation

To elucidate the effect of the sentence simplification, we applied the rules to five PPI corpora and evaluated the PPI extraction performance. We then analyzed the errors. The evaluation settings will be explained in Section 4.1. The results of the PPI extraction will be explained in Section 4.2. Finally, the deeper analysis results will be presented in Section 4.3.

4.1 Experimental Settings

The state-of-the-art PPI extraction system AkaneRE by Miwa et al. (2009) was used to evaluate our approach. The system uses a combination of three feature vectors: bag-of-words (BOW), shortest path (SP), and graph features. Classification models are trained with a support vector machine (SVM), and AkaneRE (with Mogura) is used with default parameter settings. The following two systems are used for a state-of-the-art comparison: AkaneRE with multiple parsers and corpora (Miwa et al., 2009), and Airola et al. (2008) single-parser, single-corpus system.

The rules were evaluated on the BioInfer (Pyysalo et al., 2007), AIMed (Bunescu et al., 2005), IEPA (Ding et al., 2002), HPRD50 (Fundel et al., 2006), and LLL (Nédellec, 2005) corpora[1]. Table 2 shows the number of positive (interacting) vs. all pairs. One duplicated abstract in the AIMed corpus was removed.

These corpora have several differences in their definition of entities and relations (Pyysalo et al., 2008). In fact, BioInfer and AIMed target all occurring entities related to the corpora (proteins, genes, etc). On the other hand, IEPA, HPRD50, and LLL only use limited named entities, based either on a list of entity names or on a named entity recognizer. Only BioInfer is annotated for other event types in addition to PPI, including static relations such as protein family membership. The sentence lengths are also different. The duplicated pair-containing sentences contain the following numbers of words on average: 35.8 in BioInfer, 31.3 in AIMed, 31.8 in IEPA, 26.5 in HPRD50, and 33.4 in LLL.

For BioInfer, AIMed, and IEPA, each corpus is split into training, development, and test datasets[2]. The training dataset from AIMed was the only training dataset used for validating the rules. The development datasets are used for error analysis. The evaluation was done on the test dataset, with models trained using training and development

[1] `http://mars.cs.utu.fi/PPICorpora/GraphKernel.html`

[2] This split method will be made public later.

	BioInfer		AIMed		IEPA		HPRD50		LLL	
	pos	all	pos	all	pos	all	pos	all	pos	all
training	1,848	7,108	684	4,072	256	630	-	-	-	-
development	256	928	102	608	23	51	-	-	-	-
test	425	1,618	194	1,095	56	136	-	-	-	-
all	2,534	9,653	980	5,775	335	817	163	433	164	330

Table 2: Number of positive (pos) vs. all possible sentence pairs in used PPI corpora.

	BioInfer			AIMed			IEPA		
Rule	Applied	F	AUC	Applied	F	AUC	Applied	F	AUC
No Application	0	62.5	83.0	0	61.2	87.9	0	73.4	82.5
Clause Selection	4,313	**63.5**	**83.9**	2,569	**62.5**	**88.2**	307	75.0	83.7
Entity Phrase	22,066	60.5	80.9	7,784	61.2	86.1	1,031	72.7	83.3
ALL	26,281	62.9	82.1	10,783	60.2	85.7	1,343	**75.4**	**85.7**

Table 3: Performance of PPI Extraction on test datasets. "Applied" represents the number of times the rules are applied on the corpus. "No Application" means PPI extraction without sentence simplification. ALL is the case all rules are used. The top scores for each corpus are shown in bold.

datasets). Ten-fold cross-validation (CV) was done to facilitate comparison with other existing systems. For HPRD50 and LLL, there are insufficient examples to split the data, so we use these corpora only for comparing the scores and statistics. We split the corpora for the CV, and measured the F-score (%) and area under the receiver operating characteristic (ROC) curve (AUC) as recommended in (Airola et al., 2008). We count each occurrence as one example because the correct interactions must be extracted for each occurrence if the same protein name occurs multiple times in a sentence.

In the experiments, the rules are applied in the following order: sentence–clause, exemplification, apposition, parentheses, coordination, copula, and relative-clause rules. Furthermore, if the same rule is applicable in different parts of the parse tree, then the rule is first applied closest to the leaf-nodes (deepest first). The order of the rules is arbitrary; changing it does not affect the results much. We conducted five experiments using the training and development dataset in IEPA, each time with a random shuffling of the order of the rules; the results were 77.8±0.26 in F-score and 85.9±0.55 in AUC.

Rule	Applied	F	AUC
No Application	0	72.9	84.5
Sentence Clause	145	71.6	83.8
Relative Clause	7	73.3	84.1
Copula	0	72.9	84.5
Clause Selection	152	71.4	83.4
Apposition	64	73.2	84.6
Exemplification	33	72.9	84.7
Parentheses	90	72.9	85.1
Coordination	417	73.6	85.4
Entity Phrase	605	74.1	86.6
ALL	763	**75.0**	**86.6**

Table 4: Performance of PPI Extraction on HPRD50.

Rule	Applied	F	AUC
No Application	0	79.0	84.6
Sentence Clause	135	81.3	85.2
Relative Clause	42	78.8	84.6
Copula	0	79.0	84.6
Clause Selection	178	81.0	85.6
Apposition	197	79.6	83.9
Exemplification	0	79.0	84.6
Parentheses	56	79.5	85.8
Coordination	322	**84.2**	89.4
Entity Phrase	602	83.8	90.1
ALL	761	82.9	**90.5**

Table 5: Performance of PPI Extraction on LLL.

4.2 Performance of PPI Extraction

The performance after rule application was better than the baseline (no application) on all the corpora, and most rules could be frequently applied. We show the PPI extraction performance on BioInfer, AIMed, and IEPA with rules of different groups in Table 3. The effect of using rules of different types for PPI extraction from HPRD50 and LLL is reported in Table 4 and Table 5. Table 6 shows the number of times each rule was applied in an "apply all-rules" experiment. The usability of the rules depends on the corpus, and different combinations of rules produce different

Rule	B	AIMed	IEPA	H	LLL
S. Cl.	3,960	2,346	300	150	135
R. Cl.	287	212	17	5	24
Copula	60	57	1	0	0
Cl. Sel.	4,307	2,615	318	155	159
Appos.	3,845	1,100	99	69	198
Exempl.	383	127	11	33	0
Paren.	2,721	2,158	235	91	88
Coord.	15,025	4,783	680	415	316
E. Foc.	21,974	8,168	1,025	608	602
Sum	26,281	10,783	1,343	763	761

Table 6: Distribution of the number of rules applied when all rules are applied. B:BioInfer, and H:HPRD50 corpora.

	Rules		Miwa et al.		Airola et al.	
	F	AUC	F	AUC	F	AUC
B	60.0	79.8	68.3	86.4	61.3	81.9
A	54.9	83.7	65.2	89.3	56.4	84.8
I	77.8	88.7	76.6	87.8	75.1	85.1
H	75.0	86.6	74.9	87.9	63.4	79.7
L	82.9	90.5	86.7	90.8	76.8	83.4

Table 7: Comparison with the results by Miwa et al. (2009) and Airola et al. (2008). The results with all rules are reported.

results. For the clause-selection rules, the performance was as good as or better than the baseline for all corpora, except for HPRD50, which indicates that the pair-containing clauses also include most of the important information for PPI extraction. Clause selection rules alone could improve the overall performance for the BioInfer and AIMed corpora. Entity-phrase rules greatly improved the performance on the IEPA, HPRD50, and LLL corpora, although these rules degraded the performance on the BioInfer and AIMed corpora. These phenomena hold even if we use small parts of the two corpora, so this is not because of the size of the corpora.

We compare our results with the results by Miwa et al. (2009) and Airola et al. (2008) in Table 7. On three of five corpora, our method provides better results than the state-of-the-art system by Airola et al. (2008), and also provides comparable results to those obtained using multiple parsers and corpora (Miwa et al., 2009) despite the fact that our method uses one parser and one corpus at a time. We cannot directly compare our result with Jonnalagadda and Gonzalez (2009) because the evaluation scheme, the baseline system, and test parts differ.

[FP→TN][Sentence, Parenthesis, Coordination] To characterize the AAV functions mediating this effect, cloned AAV type 2 wild-type or mutant genomes were transfected into simian virus 40 (SV40)-transformed hamster cells together with the six HSV replication genes (encoding UL5, UL8, major DNA-binding protein, **DNA polymerase**, UL42 , and **UL52**) which together are necessary and sufficient for the induction of SV40 DNA amplification (R. Heilbronn and H. zur Hausen, J. Virol. 63:3683-3692, 1989). (BioInfer.d760.s0)

[TP→FN][Coordination] Both the **GT155-calnexin** and the GT155-CAP-60 interactions were dependent on the presence of a correctly modified oligosaccharide group on GT155, a characteristic of many calnexin interactions. (AIMed.d167.s1408)

[TN→TN][Coordination, Parenthesis] **Leptin** may act as a negative feedback signal to the hypothalamic control of appetite through suppression of **neuropeptide Y** (NPY) secretion and stimulation of cocaine and amphetamine regulated transcript (CART) . (IEPA.d190.s454)

Figure 4: A rule-related error, a critical error, and a parser-related error. Regions removed by the rules are underlined, and target proteins are shown in bold. Predictions, applied rules, and sentence IDs are shown.

[FN→TP][Sentence, Coordination] **WASp** contains a binding motif for the Rho GTPase CDC42Hs as well as **verprolin** / cofilin-like actin-regulatory domains , but no specific actin structure regulated by CDC42Hs-WASp has been identified. (BioInfer.d795.s0)

[FN→TP][Parenthesis, Apposition] The protein **Raf-1** , a key mediator of mitogenesis and differentiation, associates with **p21ras** (refs 1-3) . (AIMed.d124.s1055)

[FN→TP][Sentence, Parenthesis] On the basis of far-Western blot and plasmon resonance (BIAcore) experiments, we show here that recombinant **bovine prion protein** (bPrP) (25-242) strongly interacts with the catalytic alpha/alpha' subunits of **protein kinase CK2** (also termed 'casein kinase 2') (IEPA.d197.s479)

Figure 5: Correctly simplified cases. The first sentence is a difficult (not PPI) relation, which is typed as "Similar" in the BioInfer corpus.

4.3 Analysis

We trained models using the training datasets and classified the examples in the development datasets. Two types of analysis were performed based on these results: *simplification-based* and *classification-based analysis*.

For the *simplification-based analysis*, we compared positive (interacting) and negative pair sentences that produce the exact same (inconsistent) sentence after protein names normalization and

	BioInfer				AIMed				IEPA				
Before simplification	FN	FP	TP	TN	FN	FP	TP	TN	FN	FP	TP	TN	Not Affected
After simplification	TP	TN	FN	FP	TP	TN	FN	FP	TP	TN	FN	FP	
No Error	18	2	3	35	14	21	21	8	3	2	0	4	32
No Application	3	2	0	3	0	7	8	0	0	1	0	1	7
Number of Errors	0	2	0	32	4	2	1	4	0	0	0	0	1
Number of Pairs	21	6	3	70	18	30	30	12	3	3	0	5	40
Coordination	0	0	0	20	4	2	1	0	0	0	0	0	1
Sentence	0	2	0	4	0	0	0	4	0	0	0	0	0
Parenthesis	0	0	0	5	0	0	0	0	0	0	0	0	0
Exemplification	0	0	0	2	0	0	0	0	0	0	0	0	0
Apposition	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 8: Distribution of sentence simplification errors compared to unsimplified predictions with their types (on the three development datasets). TP, True Positive; TN, True Negative; FN, False Negative; FP, False Positive. “No Error” means that simplification was correct; “No Application” means that no rule could be applied; Other rule names mean that an error resulted from that rule application. “Not Affected” means that the prediction outcome did not change.

simplification in the training dataset. The numbers of such inconsistent sentences are 7 for BioInfer, 78 for AIMed, and 1 for IEPA. The few inconsistencies in BioInfer and IEPA are from errors by the rules, mainly triggered by parse errors. The frequent inconsistencies in AIMed are mostly from inconsistent annotations. For example, even if all coordinated proteins are either interacting or not, only the first protein mention is annotated as interacting.

For the *classification-based analysis*, we specifically examine simplified pairs that were predicted differently before and after the simplification. Pairs predicted differently before and after rule application were selected: 100 random pairs from BioInfer and all 90 pairs from AIMed. For IEPA, all 51 pairs are reported. Simplified results are classified as errors when the rules affect a region unrelated to the entities in the smallest sentence clause. The results of analysis are shown in Table 8. There were 34 errors in BioInfer, and 11 errors in AIMed. Among the errors, there were five *critical errors* (in two sentences, in AIMed). Critical errors mean that the pairs lost relation-related mentions, and the errors are the only errors which caused the changes in the truth-value of the relation. There was also a *rule-related error* (in BioInfer), which means that rules with correct parse results affect a region unrelated to the entities, and parse errors (*parser-related errors*). Figure 4 shows the rule-related error in BioInfer, one critical error in AIMed, and one parser-related error in IEPA.

5 Discussion

Our end goal is to provide consistent relation extraction for real tasks. Here we discuss the “safety” of applying our simplification rules, the difficulties in the BioInfer and AIMed corpora, the reduction of errors, and the requirements for such a general (PPI) extraction system.

Our rules are applicable to sentences, with little danger of changing the relation-related mentions. Figure 5 shows three successfully simplified cases (“No Error” cases from Table 8). The sentence simplification leaves sufficient information to determine the value of the relation in these examples. Relation-related mentions remained for most of the simplification error cases. There were only five critical errors, which changed the truth-value of the relation, out of 46 errors in 241 pairs shown in Table 8. Please note that some rules can be dangerous for other relation extraction tasks. For example, the *sentence clause rule* could remove modality information (negation, speculation, etc.) modifying the clause, but there are few such cases in the PPI corpora (see Table 8). Also, the task of hedge detection (Morante and Daelemans, 2009) can be solved separately, in the original sentences, after the interacting pairs have been found. For example, in the BioNLP shared task challenge and the BioInfer corpus, interaction detection and modality are treated as two different tasks. Once other NLP tasks, like static relation (Pyysalo et al., 2009) or coreference resolution, become good enough, they can supplement or even substitute some of the proposed rules.

There are different difficulties in the BioInfer and AIMed corpora. BioInfer includes more complicated sentences and problems than the other corpora do, because 1) the apposition, coordination, and exemplification rules are more frequently used in the BioInfer corpus than in the other corpora (shown in Table 6), 2) there were more errors in the BioInfer corpus than in other corpora among the simplified sentences (shown in Table 8), and 3) BioInfer has more words per sentence and more relation types than the other corpora. AIMed contains several annotation inconsistencies as explained in Section 4.3. These inconsistencies must be removed to properly evaluate the effect of our method.

Simplification errors are mostly caused by parse errors. Our rule specifically examines a part of parser output; a probability is attached to the part. The probability is useful for defining the order of rule applications, and the n-best results by the parser are useful to fix major errors such as coordination errors. By using these modifications of rule applications and by continuous improvement in parsing technology for the biomedical domain, the performance on the BioInfer and AIMed corpora will be improved also for the all rules case.

The PPI extraction system lost the ability to capture some of the relation-related expressions left by the simplification rules. This indicates that the system used to extract some relations (before simplification) by using back-off features like bag-of-words. The system can reduce bad effects caused by parse errors, but it also captures the annotation inconsistencies in AIMed. Our simplification (without errors) can capture more general expressions needed for relation extraction. To provide consistent PPI relation extraction in a general setting (e.g. for multiple corpora or for other public text collections), the parse errors must be dealt with, and a relation extraction system that can capture (only) general relation-related expressions is needed.

6 Conclusion

We proposed a method to simplify sentences, particularly addressing the target entities for relation extraction. Using a few simple rules applicable to the output of a deep parser called Mogura, we showed that sentence simplification is effective for relation extraction. Applying all the rules improved the performance on three of the five corpora, while applying only the clause-selection rules raised the performance for the remaining two corpora as well. We analyzed the simplification results, and showed that the simple rules are applicable with little danger of changing the truth-values of the interactions.

The main contributions of this paper are: 1) explanation of general sentence simplification rules using HPSG for relation extraction, 2) presenting evidence that application of the rules improve relation extraction performance, and 3) presentation of an error analysis from two viewpoints: simplification and classification results.

As future work, we are planning to refine and complete the current set of rules, and to cover the shortcomings of the deep parser. Using these rules, we can then make better use of the parser's capabilities. We will also attempt to apply our simplification rules to other relation extraction problems than those of PPI.

Acknowledgments

This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan), Genome Network Project (MEXT, Japan), and Scientific Research (C) (General) (MEXT, Japan).

References

Airola, Antti, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter, and Tapio Salakoski. 2008. A graph kernel for protein-protein interaction extraction. In *Proceedings of the BioNLP 2008 workshop*.

Bunescu, Razvan C. and Raymond J. Mooney. 2005. A shortest path dependency kernel for relation extraction. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731.

Bunescu, Razvan C., Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun K. Ramani, and Yuk Wah Wong. 2005. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2):139–155.

Chun, Hong-Woo, Yoshimasa Tsuruoka, Jin-Dong Kim, Rie Shiba, Naoki Nagata, Teruyoshi Hishiki, and Jun'ichi Tsujii. 2006. Extraction of gene-disease relations from medline using domain dictionaries and machine learning. In *The Pacific Symposium on Biocomputing (PSB)*, pages 4–15.

Ding, J., D. Berleant, D. Nettleton, and E. Wurtele. 2002. Mining medline: abstracts, sentences, or phrases? *Pacific Symposium on Biocomputing*, pages 326–337.

Doddington, George, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program: Tasks, data, and evaluation. In *Proceedings of LREC'04*, pages 837–840.

Dorr, Bonnie, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *in Proceedings of Workshop on Automatic Summarization*, pages 1–8.

Fundel, Katrin, Robert Küffner, and Ralf Zimmer. 2006. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Jonnalagadda, Siddhartha and Graciela Gonzalez. 2009. Sentence simplification aids protein-protein interaction extraction. In *Proceedings of the 3rd International Symposium on Languages in Biology and Medicine*, pages 109–114, November.

Matsuzaki, Takuya, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient HPSG parsing with supertagging and cfg-filtering. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1671–1676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Miwa, Makoto, Rune Sætre, Yusuke Miyao, and Jun'ichi Tsujii. 2009. Protein-protein interaction extraction by leveraging multiple kernels and parsers. *International Journal of Medical Informatics*, June.

Morante, Roser and Walter Daelemans. 2009. Learning the scope of hedge cues in biomedical texts. In *Proceedings of the BioNLP 2009 Workshop*, pages 28–36, Boulder, Colorado, June. Association for Computational Linguistics.

Nédellec, Claire. 2005. Learning language in logic - genic interaction extraction challenge. In *Proceedings of the LLL'05 Workshop*.

Pyysalo, Sampo, Filip Ginter, Juho Heimonen, Jari Björne, Jorma Boberg, Jouni Järvinen, and Tapio Salakoski. 2007. BioInfer: A corpus for information extraction in the biomedical domain. *BMC Bioinformatics*, 8:50.

Pyysalo, Sampo, Antti Airola, Juho Heimonen, Jari Björne, Filip Ginter, and Tapio Salakoski. 2008. Comparative analysis of five protein-protein interaction corpora. In *BMC Bioinformatics*, volume 9(Suppl 3), page S6.

Pyysalo, Sampo, Tomoko Ohta, Jin-Dong Kim, and Jun'ichi Tsujii. 2009. Static relations: a piece in the biomedical information extraction puzzle. In *BioNLP '09: Proceedings of the Workshop on BioNLP*, pages 1–9, Morristown, NJ, USA. Association for Computational Linguistics.

Sarawagi, Sunita. 2008. Information extraction. *Foundations and Trends in Databases*, 1(3):261–377.

Vanderwende, Lucy, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Inf. Process. Manage.*, 43(6):1606–1618.

Vickrey, David and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*, pages 344–352, Columbus, Ohio, June. Association for Computational Linguistics.

Zhang, Min, Jie Zhang, Jian Su, and Guodong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *ACL-44: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 825–832. Association for Computational Linguistics.

Using Cross-Lingual Projections to Generate Semantic Role Labeled Corpus for Urdu - A Resource Poor Language

Smruthi Mukund
CEDAR
University at Buffalo
smukund@buffalo.edu

Debanjan Ghosh
Thomson Reuters R&D
debanjan.ghosh@thomsonreuters.com

Rohini K. Srihari
CEDAR
University at Buffalo
rohini@cedar.buffalo.edu

Abstract

In this paper we explore the possibility of using cross lingual projections that help to automatically induce role-semantic annotations in the PropBank paradigm for Urdu, a resource poor language. This technique provides annotation projections based on word alignments. It is relatively inexpensive and has the potential to reduce human effort involved in creating semantic role resources. The projection model exploits lexical as well as syntactic information on an English-Urdu parallel corpus. We show that our method generates reasonably good annotations with an accuracy of 92% on short structured sentences. Using the automatically generated annotated corpus, we conduct preliminary experiments to create a semantic role labeler for Urdu. The results of the labeler though modest, are promising and indicate the potential of our technique to generate large scale annotations for Urdu.

1 Introduction

Semantic Roles (also known as thematic roles) help to understand the semantic structure of a document (Fillmore, 1968). At a fundamental level, they help to capture the similarities and differences in the meaning of verbs via the arguments they define by generalizing over surface syntactic configurations. In turn, these roles aid in domain independent understanding as the semantic frames and semantic understanding systems do not depend on the syntactic configuration for each new application domain. Identifying semantic roles benefit several language processing tasks - information extraction (Surdeanu *et al.*, 2003), text categorization (Moschitti, 2008) and finding relations in textual entailment (Burchardt and Frank 2006).

Automatically identifying semantic roles is often referred to as shallow semantic parsing (Gildea and Jurafsky, 2002). For English, this process is facilitated by the existence of two main SRL annotated corpora – FrameNet (Baker *et al.*, 1998) and PropBank (Palmer *et al.*, 2005). Both datasets mark almost all surface realizations of semantic roles. FrameNet has 800 semantic frames that cover 120,000 example sentences[1]. PropBank has annotations that cover over 113,000 predicate-argument structures. Clearly English is well supported with resources for semantic roles. However, there are other widely spoken resource poor languages that are not as privileged. The PropBank based resources available for languages like Chinese (Xue and Palmer, 2009), Korean (Palmer *et al.*, 2006) and Spanish (Taule, 2008) are only about two-thirds the size of the English PropBank.

Several alternative techniques have been explored in the literature to generate semantic role labeled corpora for resource poor languages as providing manually annotated data is time consuming and involves intense human labor. Ambati and Chen (2007) have conducted an extensive survey and outlined the benefits of using parallel corpora to transfer annotations. A wide range of annotations from part of speech (Hi and Hwa, 2005) and chunks (Yarowsky *et al.*, 2001) to word senses (Diab and Resnik, 2002), dependencies (Hwa *et al.*, 2002) and semantic roles (Pado and Lapata, 2009) have been successfully transferred between languages. FrameNet style annotations in Chinese is obtained by mapping English FrameNet entries directly to concepts listed in HowNet[2] (online ontology for Chinese) with an accuracy of 68% (Fung and Chen, 2004).

[1] Wikipedia - http://en.wikipedia.org/wiki/PropBank
[2] http://www.keenage.com/html/e_index.html

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 797–805,
Beijing, August 2010

Fung *et al.* (2007) analyze an automatically annotated English-Chinese parallel corpus and show high cross-lingual agreement for PropBank roles (range of 75%-95% based on the roles).

In this paper we explore the possibility of using English-Urdu parallel corpora to generate SRL annotations for Urdu, a less commonly taught language (LCTL). Earlier attempts to generate SRL corpora using annotation projections have been for languages such as German, French (Pado and Lapata, 2009) and Italian (Moschitti, 2009) that have high vocabulary overlap with English. Also, German belongs to the same language family as English (Germanic family). Urdu on the other hand is an Indic language that is grammatically very different and shares almost no vocabulary with English.

The technique of cross lingual projections warrants good BLEU score that ensures correct word alignments. According to NIST 2008 Open Machine Translation challenge [3], a 0.2280 best BLEU score was achieved for Urdu to English translation. This is comparable to the BLEU scores achieved for German to English – 0.253 and French to English – 0.3 (Koehn, 2005). But, for SRL transfer, perfect word alignment is not mandatory as SRL requires semantic correspondence only. According to Fillmore (1982) semantic frames are based on conceptual structures. They are generalizations over surface structures and hence less prone to syntactic variations. Since English and Urdu have a reasonable semantic correspondence (Example 3), we believe that the projections when capped with a post processing step will considerably reduce the noise induced by inaccurate alignments and produce acceptable mappings.

Hindi is syntactically similar to Urdu. These languages are standardized forms of Hindustani. They are free word order languages and follow a general SOV (Subject-Object-Verb) structure. Projection approach has been used by (Mukerjee *et al.*, 2006) and (Sinha, 2009) to transfer verb predicates from English onto Hindi. Sinha (2009) achieves a 90% F-Measure in verb predicate transfer from English to Hindi. This shows that using cross lingual transfer approach to obtain semantic annotations for Urdu from English is an idea worth exploring.

1.1 Approach

Our approach leverages existing English PropBank annotations provided via the SemLink[4] corpus. SemLink provides annotations for VerbNet using the **pb** (**P**rop**B**ank) attribute. By using English-Urdu parallel corpus we acquire verb predicates and their arguments. When we transfer verb predicates (lemmas), we also transfer **pb** attributes. We obtain annotation projections from the parallel corpora as follows:

1. Take a pair of sentences *E* (in English) and *U* (in Urdu) that are translations of each other.
2. Annotate *E* with semantic roles.
3. Project the annotations from *E* onto *U* using word alignment information, lexical information and linguistic rules that involve syntactic information.

There are several challenges to the annotation projection technique. Dorr (1994) presents some major lexical-semantic divergence problems applicable in this scenario:

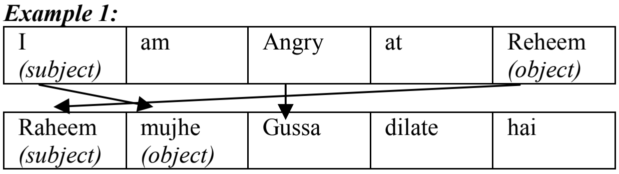
(a) Thematic Divergence - In some cases, although there exists semantic parallelism, the theme of the English sentence captured in the subject changes into an object in the Urdu sentence (Example 1).

(b) Conflatational Divergence - Sometimes target translations spans over a group of words (Example 1: *plays* is mapped to *kirdar ada).* Trying to ascertain this word span for semantic roles is difficult as the alignments can be incomplete and very noisy.

(c) Demotional divergence and Structural divergence - Despite semantic relatedness, in some sentence pairs, alignments obtained from simple projections generate random matchings as the usage is syntactically dissimilar (Example 2).

Handling all challenges adds complexity to our model. The heuristic rules that we implement are guided by linguistic knowledge of Urdu. This increases the effectiveness of the alignments.

Example 1:



I *(subject)*	am	Angry	at	Reheem *(object)*

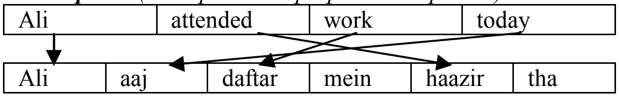
Raheem *(subject)*	mujhe *(object)*	Gussa	dilate	hai

(Raheem brings anger in me)

[3] http://www.itl.nist.gov/iaui/894.01/tests/mt/2008/doc/mt08_official_results_v0.html

[4] http://verbs.colorado.edu/semlink/

Example 2: *(noun phrase to prepositional pharse)*



(Ali was present at work today)

2 Generating Parallel Corpora

PropBank provides SRL annotated corpora for English. It uses predicate independent labels (ARG0, ARG1, etc.) which indicate how a verb relates to its arguments. The argument types are consistent across all uses of a single verb and do not consider the sense of the verb. We use the PropBank annotations provided for the Wall Street Journal (WSJ) part of the Penn Tree bank corpus (Marcus *et al.*, 2004). The arguments of a verb are labeled sequentially from ARG0 to ARG5 where ARG0 is the proto-typical Agent, ARG1 is the proto-typical patient, ARG2 is the recipient, and so on. There are other adjunct tags in the dataset that are indicated by ARGM that include tags for location (ARGM-LOC), temporal tags (ARGM-TMP) etc.

An Urdu corpus of 6000 sentences corresponding to 317 WSJ articles of Penn Tree Bank corpus is provided by CRULP[5] (used in the NIST 2008 machine translation task). We consider 2350 English sentences with PropBank annotations that have corresponding Urdu translations (CRULP corpus) for our experiments.

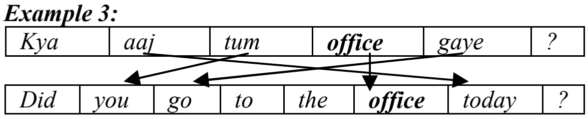
2.1 Sentence Alignment

Sentence alignment is a prerequisite for any parallel corpora processing. As the first step, we had to generate a perfect sentence aligned parallel corpus as the translated sentences, despite belonging to the same domain (WSJ – Penn tree bank), had several errors in demarcating the sentence boundaries.

Sentence alignment between English and Urdu is achieved over two iterations. In the first iteration, the length of each sentence is calculated based on the occurrence of words belonging to important part of speech categories such as proper nouns, adjectives and verbs. Considering main POS categories for length assessment helps overcome the conflatational divergence issue. For each English sentence, Urdu sentences with the same length are considered to be probable candidates for alignment. In the second iteration, an Urdu-English lexicon is used on the Urdu corpus and English translations are obtained. An English-Urdu sentence pair with maximum lexical match is considered to be sentence aligned.

Clearly this method is highly dependent on the existence of an exhaustive Urdu-English dictionary. The lexicons that we use to perform lookups are collected by mining Wikipedia and other online resources (Mukund *et al.,* 2010). However, lexicon lookups will fail for Out-Of-Vocabulary words. There could also be a collision if Urdu sentences have English transliterated words (Example 3, *"office"*). Such errors are manually verified for correctness.

Example 3:



2.2 Word Alignment

In the case of generating word alignments it is beneficial to calculate alignments in both translation directions (English – Urdu and Urdu - English). This nature of symmetry will help to reduce alignment errors. We use the Berkeley Aligner[6] word alignment package which implements a joint training model with posterior decoding (Liang *et al.*, 2006) to consider bidirectional alignments. Predictions are made based on the agreements obtained by two bidirectional models in the training phase. The intuitive objective function that incorporates data likelihood and a measure of agreement between the models is maximized using an EM-like algorithm. This alignment model is known to provide 29% reduction in AER over IBM model 4 predictions.

On our data set the word alignment accuracy is 71.3% (calculated over 200 sentence pairs). In order to augment the alignment accuracy, we added 3000 Urdu-English words and phrases obtained from the Urdu-English dictionary to our parallel corpus. The alignment accuracy improved by 3% as the lexicon affects the word cooccurrence count.

Word alignment in itself does not produce accurate semantic role projections from English to Urdu. This is because the verb predicates in Urdu can span more than one token. Semantic roles

[5] http://www.crulp.org/

[6] http://nlp.cs.berkeley.edu/Main.html

can cover sentential constituents of arbitrary length, and simply using word alignments for projection is likely to result in wrong role spans. Also, alignments are not obtained for all words. This could lead to missing projections.

One way to correct these alignment errors is to devise token based heuristic rules. This is not very beneficial as writing generic rules is difficult and different errors demand specific rules. We propose a method that considers POS, tense and chunk information along with word alignments to project annotations.

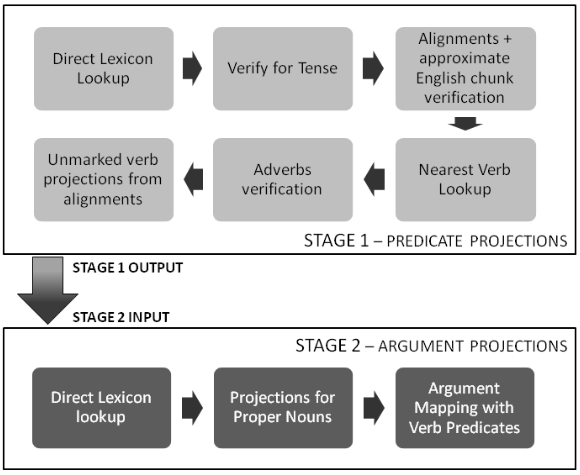

Direct Lexicon Lookup
Verify for Tense
Alignments + approximate English chunk verification
Nearest Verb Lookup
Adverbs verification
Unmarked verb projections from alignments
STAGE 1 – PREDICATE PROJECTIONS
STAGE 1 OUTPUT
STAGE 2 INPUT
STAGE 2 – ARGUMENT PROJECTIONS
Direct Lexicon lookup
Projections for Proper Nouns
Argument Mapping with Verb Predicates

Figure 1: Projection model

Our proposed approach can be explained in two stages as shown in figure 1. In Stage 1 only verb predicates are transferred from English to Urdu. Stage 2 involves transfer of arguments and depends on the output of Stage 1. Predicate transfer cannot rely entirely on word alignments (§3). Rules devised around the chunk boundaries boost the verb predicate recognition rate.

Any verb group sequence consisting of a main verb and its auxiliaries are marked as a verb chunk. Urdu data is tagged using the chunk tag set proposed exclusively for Indian languages by Bharati *et al.*, (2006). Table 1 shows the tags that are important for this task.

Verb Chunk	Description
VGF	Verb group is finite (decided by the auxiliaries)
VGNF	Verb group for non-finite adverbial and adjectival chunk
VGNN	Verb group has a gerund

Table 1: Verb chunk tags in Urdu

The sentence aligned parallel corpora that we feed as input to our model is POS tagged for both English and Urdu. Urdu data is also tagged for chunk boundaries and morphological features like tense, gender and number information. Named Entities are also marked on the Urdu data set as they help in tagging the ARGM arguments. All the NLP taggers (POS, NE, Chunker, and Morphological Analyzer) used in this work are detailed in Mukund *et al.,* (2010).

English data is not chunked using a conventional chunk tagger. Each English sentence is split into virtual phrases at boundaries determined by the following parts of speech – IN, TO, MD, POS, CC, DT, SYM,: (Penn Tree Bank tagset). These tags represent positions in a sentence that typically mark context transitions (they are mostly the closed class words). We show later how these approximate chunks assist in correcting predicate mappings.

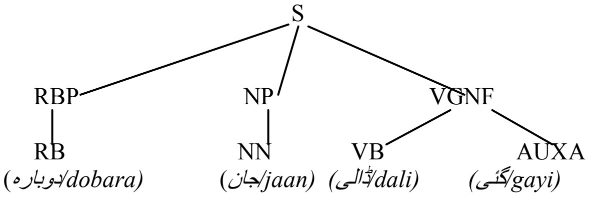
We use an Urdu-English dictionary (§2.1) that assigns English meanings to Urdu words in each sentence. Using translation information from a dictionary can help transfer verb predicates when the translation equivalent preserves the lexical meaning of the source language.

The first rule that gets applied for predicate transfer is based on lexicon lookup. If the English verb is found to be a synonym to an Urdu word that is part of a verb chunk, then the lemma associated with the English word is transferred to the entire verb chunk in Urdu. However not all translations' equivalents are lexically synonymous. Sometimes the word used in Urdu is different in meaning to that in English but relevant in the context (lexical divergence).

The word alignments considered in proximity to the approximate English chunks come to rescue in such scenarios. Here, for all the words occurring in each Urdu verb chunk, corresponding English aligned words are found from the word alignments. If the words that are found belong to the same approximate English chunk, then the verb predicate of that chunk (if present) is projected onto the verb chunk in Urdu. This heuristic technique increases the verb projection accuracy by about 15% as shown in §4.

The Penn tree bank tag set for English part of speech has different tags for verbs based on the tense information. VBD is used to indicate past tense, and VBP and VBZ for present tense. Urdu also has the tense information associated with the verbs in some cases. We exploit this similarity to project the verb predicates from English onto Urdu.

The adverbial chunk in Urdu includes pure adverbial phrases. These chunks also form part of the verb predicates.



[English meaning – Revitalized]

Figure 2: example for demotional divergence

E.g. consider the English word “revitalized” (figure 2). This is tagged VBD. However, the Urdu equivalent of this word is *“دوبارہ جان ڈالی گئی” (dobara jaan daali gayi ~ to put life in again)*. The POS tags are “RB, NN, VB, AUXA” *(adverb, noun, verb, aspectual auxiliary)*. The word *“dobara”* is a part of the adverbial chunk RBP and the infinite verb chunk VGNF spans across the last two words *“daali gayi”*. *“jaan”* is a noun chunk. This kind of demotional divergence is commonly observed in languages like Hindi and Urdu. In order to consider this entire phrase to be the Urdu equivalent representation of the English word *“revitalized”,* a rule for adverbial chunk is included as the last step to account for unaccommodated English verbs in the projections.

In the PropBank corpus, predicate argument relations are marked for almost all occurrences of non-copula verbs. We however do not have POS tags that help to identify non-copula words. Words that can be auxiliary verbs occur as non-copula verbs in Urdu. We maintain a list of such auxiliary verbs. When the verb chunk in Urdu contains only one word and belongs to the list, we simply ignore the verb chunk and proceed to the next chunk. This avoids several false positives in verb projections.

Stage 2 of the model includes the transfer of arguments. In order to see how well our method works, we project all argument annotations from English onto Urdu. We do not consider word alignments for arguments with proper nouns. The double metaphone algorithm (Philips 2000) is applied on both English NNP (proper noun) tagged words as well as English transliterated Urdu (NNP) tagged words. Arguments from English are mapped onto Urdu for word pairs with the same metaphone code.

For other arguments, we consider word alignments in proximity to verb predicates. The argument boundaries are determined based on chunk and POS information. We observe that our method projects the annotations associated with nouns fairly well. However, when the arguments contain adjectives, the boundaries are discontinuous. In such cases, we consider the entire chunk without the case marker as a probable candidate for the projected argument. We also have some issues with the ARGM-MOD arguments in that they overlap with the verb predicates. When the verb predicate that it overlaps with is a complex predicate, we consider the entire verb chunk to be the Urdu equivalent candidate argument. These rules along with word alignments yield fairly accurate projections.

The rules that we propose are dependent on the POS, chunk and tense information that are language specific. Hence our method is language independent only to the extent that the new language considered should have similar syntactic structure as Urdu. Indic languages fall in this category.

3 Verb Predicates

Detecting verb predicates can be a challenging task especially if very reliable and efficient tools such as POS tagger and chunkers are not available. We apply the POS tagger (CRULP tagset, 88% F-Score) and Chunker (Hindi tagset, 90% F-Score) provided by Mukund *et al.,* (2010) on the Urdu data set and show that syntactic information helps to compensate alignment errors. Stanford POS tagger[7] (Penn Tree bank tagset) is applied on the English data set.

Predicates can be simple predicates that lie within the chunk boundary or complex predicates when they span across chunk boundaries. When verbs in English are expressed in Urdu/Hindi, in several cases, more than one word is used to achieve perfect translation. In English the tense of the verb is mostly captured by the verb morpheme such as *“ask**ed**” “sa**id**” “say**ing**”*. In Urdu the tense is mostly captured by the auxiliary verbs. So a single word English verb such as *“talking”* would be translated into two words

[7] http://nlp.stanford.edu/software/tagger.shtml

"batein karna" where *"karna"~ do* is the auxiliary verb. However this cannot be generalized as there are instances when translations are word to word. E.g. *"said"* is mapped to a single word Urdu verb *"kaha"*.

Complex predicates in Urdu can occur in the following POS combinations. *Noun+Verb, Adjective+Verb, Verb+Verb, Adverb+Verb*. Table 2 lists the main verb tags present in the Urdu POS tagset. (refer Penn Tree bank POS tagset for English tags).

Urdu Tags	Description
VB	Verb
VBI	Infinitive Verb
VBL	Light Verb
VBLI	Infinitive Light Verb
VBT	Verb to be
AUXA	Aspectual Auxiliary
AUXT	Tense Auxiliary

Table 2: Verb tags

Auxiliary verbs in Urdu occur alongside VB, VBI, VBL or VBLI tags. Sinha (2009) defines complex predicates as a group of words consisting of a noun (NN/NNP), an adjective (JJ), a verb (VB) or an adverb (RB) followed by a light verb (VBL/VBLI). Light verbs are those which contribute to the tense and agreement of the verb (Butt and Geuder, 2001). However, despite the existence of a light verb tag, it is noticed that in several sentences, verbs followed by auxiliary verbs need to be grouped as a single predicate. Hence, we consider such combinations as belonging to the complex predicate category.

ENG- *According*_VBG *to*_TO *some*_DT *estimates*_NNS *the*_DT *rule*_NN *changes*_NNS *would*_MD ***cut_VB*** *insider*_NN *filings*_NNS *by*_IN *more*_JJR *than*_IN *a*_DT *third*_JJ
URD- [*Kuch*_QN *andaazon*_NN *ke*_CM *mutabiq*_NNCM]_NP [*kanoon*_NN *mein*_CM]_NP [*tabdeeliayan*_NN]_NP[*androni*_JJ *drjbndywn*_NN *ko*_CM]_NP [*ayk*_CD *thayiy*_FR *se*_CM]_NP [***zyada***_I ***kam***_JJ]_JJP [***karey***_VBL ***gi***_AUXT]_VGF

Example 4

Example 4 demonstrates the existence of a light verb in a complex predicate. The English verb *"cut"* is mapped to *"کم کریں گی" (kam karey gi)* belonging to the VBF chunk group.

ENG- *Rolls*_NNP *-*_*:* *Royce*_NNP *Motor*_NNP *Cars*_NNPS *Inc.*_NNP ***said***_VBD *it*_PRP ***expects***_VBZ *its*_PRP*$* *U.S.*_NNP *sales*_NNS *to*_TO ***remain***_VB *steady*_JJ *at*_IN *about*_IN *1 200*_CD *cars*_NNS *in*_IN *1990*_CD
URD - [*Rolls Royce motor car inc*_NNPC *ne*_CM]_NP [***kaha*_VB**]_VBNF [*wo*_PRP]_NP [*apney*_PRRFP$]_NP [*U.S.*_NNP *ki*_CM]_NP [*frwKt*_NN *ko*_CM]_NP [*1990*_CD *mein*_CM]_NP [*takreeban*_RB]_RBP [*1200_CD karon*_NN *par*_CM]_NP [*mtwazn*_JJ]_JJP [*rakhne*_VBI *ki*_CM]_VGNN [*tawaqqo*_NN]_NP [*karte*_VB *hai*_AUXT]_VGF

Example 5

In example 5, *"said"* corresponds to one Urdu word *"کہا"(kaha)* that also captures the tense information (past). However, consider the verb *"expects"*. This is a clear case of noun-verb complex predicate where *"expects"* is mapped to *"توقع کرتی ہے"(tawaqqo karte hai)*.

ENG- *Not*_RB *all*_PDT *those*_DT *who*_WP ***wrote***_VBD ***oppose***_VBP *the*_DT *changes*_NNS
URD -*wo tamaam jinhon ne* ***likha*** *tabdeeliyon ke* [***mukhalif***_JJ]_JJP [*nahi*_RB]_RBP [***hain***_VBT]_VGF

Example 6

In example 6, verb predicates are *"wrote"* and *"oppose"*. Consider the word *"oppose"*. There are two ways of representing this word in Urdu. As a verb chunk the translation would be ***"mukhalifat** nahi karte"* and as an adjectival chunk *"**mukhalif** nahi hai"*. The latter form of representation is used widely in the available translation corpus. The Urdu equivalent *of "oppose"* is *"مخالف ہیں"(mukhalif hai).*

Another interesting observation in example 6 is the existence of discontinuous predicates. Though *"oppose"* is one word in English, the Urdu representation has two words that do not occur together. The adverb *"nahi"* ~*"not"* occurs between the adjective and the verb. Statistically dealing with this issue is extremely challenging and affects the boundaries of other arguments. Generalizing the rules needed to identify discontinuous predicates requires more detailed analysis of the corpus – from the linguistic aspect – and has not been attempted in this paper. We however map *"مخالف نہیں ہیں"(mukhalif nahi hai)* to the predicate *"oppose"*. *"nahi"* is treated as an argument ARG_NEG in PropBank.

4 Projection Results

It is impossible for us to report our projection results on the entire data set as we do not have it manually annotated. For the purpose of evaluation, we manually annotated 100 long sentences (L) and 100 short sentences (S) from the full 2350 sentence set. All the results are reported on this 200 set of sentences. Set L has sentences that each has more than two verb predicates and several arguments. The number of words per sentence here is greater than 55. S; on the other hand has sentences with about 40 words each and no complex SOV structures.

The results shown in Table 3 are for all tags (verbs+args) that are projected from English onto Urdu. In order to understand why the performance over L dips, consider the results in Table 4 that are for verb projections only. Some long sentences in English have Urdu translations that do not maintain the same structure. For example an English phrase – *"... might **prompt** individuals to **get** out of stocks altogether"* is written in Urdu in a way that the English representation would be *"what makes individuals to **get** out of stocks is ..."*. The Urdu equivalent word for *"prompt"* is missing and the associated lemma gets assigned to the Urdu equivalent of *"get"* (the next lemma). This also affects the argument projections. Another reason is the effect of word alignments itself. Clearly longer sentences have greater alignment errors.

All tags[8]	100 long sentences	100 short sentences
Actual Tags	1267	372
Correct Tags	943	325
Found Tags	1212	353
L : Precision 77.8% Recall 74.4% F-Score 76%		
S: Precision 92% Recall 87.4% F-Score 89.7%		

Table 3: when all tags are considered

Comparing the results of Table 4 to Table 3, we see that argument projections affect the recall. This is because the projections of arguments depend not only on the word alignments but also on the verb predicates. Incorrect verb predicates affect the argument projections.

Only lemma	100 long sentences	100 short sentences
Actual Tags	670	240
Correct Tags	490	208
Overall Tags	720	257
L: Precision 68% Recall 73.1% F-Score 70.45%		
S : Precision 80.9% Recall 86.6% F-Score 83.65%		

Table 4: for verb projections only

Table 5 summarizes the results obtained when only the word alignments are considered to project all tags. But when virtual phrase boundaries in English are also considered, the F-score improves by 8% (Table 6). This is because virtual boundaries in a way mark context switch and when considered in proximity to the word alignments yield better predicate boundaries.

100 long sentences : only alignments	
Actual Tags	1267
Correct Tags	617
Overall Tags	782
Precision 78.9% Recall 48.7% F-Score 60.2%	

Table 5: with only word alignments

100 long sentences : alignments + virtual boundaries	
Actual Tags	1267
Correct Tags	792
Overall Tags	1044
Precision 75.8% Recall 62.5% F-Score 68.5%	

Table 6: with word alignments and virtual boundaries

100 Sentences	ARG 0	ARG 1	ARG 2	ARG 3	ARG M
Long	124	271	67	25	140
Found	111	203	36	12	114
P %	89.5	74.9	53.7	48	81.42
Short	34	47	4	2	19
Found	30	45	4	2	19
P %	88.2	95.7	100	100	100

Table 7: results of argument projections Precision (P) on arguments

Table 7 shows the results of argument projections over the first 4 arguments of PropBank – ARG0, ARG1, ARG2 and ARG3 (out of 24 arguments, majority are sparse in our test set) and the adjunct tag set ARGM.

5 Automatic Detection

The size of SRL annotated corpus generated for Urdu is limited with only 2350 sentences. To explore the possibilities of augmenting this data set, we train verb predicate and argument detection models. The results show great promise in generating large-scale automatic annotations.

5.1 Verb Predicate Detection

Verb predicate detection happens in two stages. In the first stage, the predicate boundaries are marked using a CRF (Lafferty *et al.*, 2001) based sequence labeling approach. The training data for the model is generated by annotating the automatically annotated Urdu SRL corpus using BI

[8] Tags - lemma (verb predicates) + arguments, Actual tags – number of tags in the English set, Found tags – number of tags transferred to Urdu, Correct Tags – number of tags correctly transferred

annotations. E.g. *kam* B-VG, k*arne par* I-VG. The non-verb predicates are labeled “-1”. The model uses POS, chunk and lexical information as features. We report the results on a set of 77 sentences containing a mix of short and long sentences.

Number of verb predicates correctly marked	377
Num of verb predicates found	484
Actual num of verb predicates	451
Precision 77.8% Recall 83.5% F-Score 80.54%	

Table 8: CRF results for verb boundaries

Every verb predicate is associated with a lemma mapped from the English VerbNet map file[9]. E.g. the Urdu verb *“کم کرنے پر” (kam karne par)* has the lemma *“lower”*. The second stage includes assigning these lemmas. Lemma assignment is based on lookups from a VerbNet like map file. We have compiled a large set of Urdu verb predicates by mapping translations found in the automatically annotated corpus to the VerbNet map file. This Urdu verb predicate list also accommodates complex predicates that occur along with verbs such as *“karna – to do”, “paana – to get”,* etc. (along with different variations of these verbs – *karte, kiya, paate etc.*). This verb predicate list (manually corrected) consists of 800 entries. Since our gold standard test set is very small, the lemma assignment for all verb predicates is 100% (no **pb** values and hence no senses). This list, however, has to be augmented further to meet the standards of the English VerbNet map file.

5.2 Argument Detection

Argument detection (SRL) is done in two steps: (1) argument boundary detection (2) argument label assignment. We perform tests for step 2 to show how well a standard SVM role detection model works on the automatically generated Urdu data set. For each pair of correct predicate *p* and an argument *i* we create a feature representation $F_{p,a}$ ~ set T of all arguments. To train a multi-class role-classifier, given the set T of all arguments, T can be rationalized as $T^{+}_{\arg i}$ (positive instances) and $T^{-}_{\arg i}$ (negative instances) for each argument *i*. In this way, individual ONE-vs-ALL (Gildea and Jurafsky, 2002) classifier for each argument *i* is trained. In the testing phrase, given an unseen sentence, for each argument $F_{p,q}$ is generated and classified by each individual classifier.

We created a set of *standard* SRL features as shown in table 9. The results (Tables 10 and 11), though not impressive, are promising. We believe that by increasing the number of samples (for each argument) in the training set and intelligently controlling the negative samples, the results can be improved significantly.

Training – 2270 sentences with 7315 argument instances. Test – 77 sentences with 496 argument instances. (22 different role types)

BaseLine Features (BL)	phrase-type (syntactic category; NP, PP etc.), predicate (in our case, verb group), path (syntactic path from the argument constituent to the predicate), head words (argument and the predicate respectively), position (whether the phrase is before or after the predicate)
Detailed Features	BL + POS (of the first word in the predicate), chunk tag of the predicate, POS (of the first word of the constituent argument), head word (of the verb group in a complex predicate), named entity (whether the argument contains any named entity, such as location, person, organization etc.)

Table 9: Features for SRL

Kernel/features	Precision	Recall	F-Score
LK – BL	71.88	48.25	57.74
LK – all	73.91	47.55	57.87
PK – BL	74.19	48.25	58.47
PK –all (best)	73.47	49.65	59.26

Table 10: Arg0 performance

Kernel/features	Precision	Recall	F-Score
LK – BL	69.35	22.87	34.40
LK – all	69.84	23.4	35.05
PK – BL	73.77	24.14	36.38
PK –all (best)	73.8	26.06	38.52

Table 11: Arg1 Performances
(PK - polynomial kernel LK – Linear kernel)

6 Conclusion

In this work, we develop an alignment system that is tailor made to fit the SRL problem scope for Urdu. Furthermore, we have shown that despite English being a totally different language, resources for Urdu can be generated if the subtle grammatical nuances of Urdu are accounted for while projecting the annotations. We plan to work on argument boundary detection and explore other features for argument detection. The lemma set generated for Urdu is being refined for finer granularity.

[9] http://verbs.colorado.edu/semlink/semlink1.1/vn-pb/README.TXT

References

Ambati, Vamshi and Chen, Wei, 2007. Cross Lingual Syntax Projection for Resource-Poor Languages. CMU.

Baker, Collin .F., Charles J. Fillmore, John B. Lowe. 1998. The Berkeley Frame Net project. *COLING-ACL*.

Bharati, Akshar, Dipti Misra Sharma, Lakshmi Bai and Rajeev Sangal. 2006. AnnCorra: Annotating Corpora Guidelines For POS And Chunk Annotation For Indian Language. *Technical Report*, Language Technologies Research Centre IIIT, Hyderabad.

Burchardt, Aljoscha and Anette Frank. 2006. Approaching textual entailment with LFG and FrameNet frames. *RTE-2 Workshop*. Venice, Italy.

Butt, Miriam and Wilhelm Geuder. 2001. On the (semi)lexical status of light verbs. *Norbert Corver and Henk van Riemsdijk, (Eds.), Semi-lexical Categories: On the content of function words and the function of content words*, Mouton de Gruyter, pp. 323–370, Berlin.

Diab, Mona and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. *40th Annual Meeting of ACL*, pp. 255-262, Philadelphia, PA.

Dorr, Bonnie, J. 1994. Machine Translation Divergences: A Formal Description and Proposed Solution. *ACL,* Vol. 20(4), pp. 597-631.

Fillmore, Charles J. 1968. The case for case. *Bach, & Harms(Eds.), Universals in Linguistic Theory*, pp. 1-88. Holt, Rinehart, and Winston, New York.

Fillmore, Charles J. 1982. Frame semantics. *Linguistics in the Morning Calm*, pp.111-137. Hanshin, Seoul, S. Korea.

Fung, Pascale and Benfeng Chen. 2004. BiFrameNet: Bilingual frame semantics resources construction by cross-lingual induction. *20th International Conference on Computational Linguistics*, pp. 931-935, Geneva, Switzerland.

Fung, Pascale, Zhaojun Wu, Yongsheng Yang and Dekai Wu. 2007. Learning bilingual semantic frames: Shallow semantic parsing vs. semantic role projection. *11th Conference on Theoretical and Methodological Issues in Machine Translation*, pp. 75-84, Skovde, Sweden.

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, Vol. 28(3), pp. 245-288.

Hi, Chenhai and Rebecca Hwa. 2005. A backoff model for bootstrapping resources for non-english languages. *Joint Human Language Technology Conference and Conference on EMNLP*, pp. 851-858, Vancouver, BC.

Hwa, Rebecca, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluation translational correspondance using annotation projection. *40th Annual Meeting of ACL*, pp. 392-399, Philadelphia, PA.

Koehn, Phillip. 2005. “Europarl: A parallel corpus for statistical machine translation,” MT summit, Citeseer.

Lafferty, John D., Andrew McCallum and C.N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *18th International Conference on Machine Learning*, pp. 282-289.

Liang, Percy, Ben Taskar, and Dan Klein. 2006. Alignment by Agreement, *NAACL*.

Marcus, Mitchell P., Beatrice Santorini and Mary Ann Marcinkiewicz. 2004. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, Vol. 19(2), pp. 313-330.

Moschitti, Alessandro. 2008. Kernel methods, syntax and semantics for relational text categorization. *17th ACM CIKM*, pp. 253-262, Napa Valley, CA.

Mukerjee, Amitabh , Ankit Soni and Achala M. Raina. 2006. Detecting Complex Predicates in Hindi using POS Projection across Parallel Corpora. *Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pp. 11–18. Sydney.

Mukund, S., Srihari, R. K., and Peterson, E. 2010. An Information Extraction System for Urdu – A Resource Poor Language. *Special Issue on Information Retrieval for Indian Languages*. TALIP.

Pado, Sebastian and Mirella Lapata. 2009. Cross-Lingual annotation Projection of Semantic Roles. *Journal of Artificial Intelligence Research*, Vol. 36, pp. 307-340.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, Vol. 31(1).

Palmer, Martha, Shijong Ryu, Jinyoung Choi, Sinwon Yoon, and Yeongmi Jeon. 2006. Korean Propbank. *Linguistic data consortium*, Philadelphia.

Philips, Lawrence. 2000. The Double Metaphone Search Algorithm. *C/C++ Users Journal.*

Sinha, R. Mahesh K. 2009. Mining Complex Predicates In Hindi Using A Parallel Hindi-English Corpus. *ACL International Joint Conference in Natural Language Processing*, pp 40.

Surdeanu, Mihai, Sanda Harabagiu, John Williams and Paul Aarseth. 2003. Using predicate-argument structures for information extraction. *41st Annual Meeting of the Association for Computational Linguistics*, pp. 8-15, Sapporo, Japan.

Taule, Mariona, M. Antonio Marti, and Marta Recasens. 2008. Ancora: Multi level annotated corpora for Catalan and Spanish. *6th International Conference on Language Resources and Evaluation*, Marrakesh, Morocco.

Xue, Nianwen and Martha Palmer. 2009. Adding semantic roles to the Chinese treebank. *Natural Language Engineering*, Vol. 15(1), pp. 143-172.

Yarowsky, David, Grace Ngai and Richard Wicentowski. 2001. Inducing multi lingual text analysis tools via robust projection across aligned corpora. *1st Human Language Technology Conference*, pp. 161-168, San Francisco, CA.

Recognition of Affect, Judgment, and Appreciation in Text

Alena Neviarouskaya
University of Tokyo
lena@mi.ci.i.u-tokyo.ac.jp

Helmut Prendinger
Nat. Institute of Informatics Tokyo
helmut@nii.ac.jp

Mitsuru Ishizuka
University of Tokyo
ishizuka@i.u-tokyo.ac.jp

Abstract

The main task we address in our research is classification of text using fine-grained attitude labels. The developed @AM system relies on the compositionality principle and a novel approach based on the rules elaborated for semantically distinct verb classes. The evaluation of our method on 1000 sentences, that describe personal experiences, showed promising results: average accuracy on the fine-grained level (14 labels) was 62%, on the middle level (7 labels) – 71%, and on the top level (3 labels) – 88%.

1 Introduction and Related Work

With rapidly growing online sources aimed at encouraging and stimulating people's discussions concerning personal, public or social issues (news, blogs, discussion forums, etc.), there is a great need in development of a computational tool for the analysis of people's attitudes. According to the Appraisal Theory (Martin and White, 2005), attitude types define the specifics of appraisal being expressed: affect (personal emotional state), judgment (social or ethical appraisal of other's behaviour), and appreciation (evaluation of phenomena).

To analyse contextual sentiment of a phrase or a sentence, rule-based approaches (Nasukawa and Yi, 2003; Moilanen and Pulman, 2007; Subrahmanian and Reforgiato, 2008), a machine-learning method using not only lexical but also syntactic features (Wilson et al., 2005), and a model of integration of machine learning approach with compositional semantics (Choi and Cardie, 2008) were proposed. With the aim to recognize fine-grained emotions from text on the level of distinct sentences, researchers have employed a keyword spotting technique (Chuang and Wu, 2004; Strapparava et al., 2007), a technique calculating emotion scores using Pointwise Mutual Information (PMI) (Kozareva et al., 2007), an approach inspired by common-sense knowledge (Liu et al., 2003), rule-based linguistic approaches (Boucouvalas, 2003; Chaumartin, 2007), machine-learning methods (Alm, 2008; Aman and Szpakowicz, 2008; Strapparava and Mihalcea, 2008), and an ensemble based multi-label classification technique (Bhowmick et al., 2009).

Early attempts to focus on distinct attitude types in the task of attitude analysis were made by Taboada and Grieve (2004), who determined a potential value of adjectives for affect, judgement and appreciation by calculating the PMI with the pronoun-copular pairs '*I was (affect)*', '*He was (judgement)*', and '*It was (appreciation)*', and Whitelaw et al. (2005), who used a machine learning technique (SVM) with fine-grained semantic distinctions in features (attitude type, orientation) in combination with "bag of words" to classify movie reviews. However, the concentration only on adjectives expressing appraisal and their modifiers greatly narrows the potential of the Whitelaw et al. (2005) approach.

In this paper we introduce our system @AM (**AT**titude **A**nalysis **M**odel), which (1) classifies sentences according to the fine-grained attitude labels (nine affect categories (Izard, 1971): 'anger', 'disgust', 'fear', 'guilt', 'interest', 'joy', 'sadness', 'shame', 'surprise'; four polarity labels for judgment and appreciation: 'POS jud', 'NEG jud', 'POS app', 'NEG app'; and 'neutral'); (2) assigns the strength of the attitude; and (3) determines the level of confidence, with which the attitude is expressed. @AM relies on a compositionality principle and a novel approach

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 806–814,
Beijing, August 2010

based on the rules elaborated for semantically distinct verb classes.

2 Lexicon for Attitide Analysis

We built a lexicon for attitude analysis that includes: (1) attitude-conveying terms; (2) modifiers; (3) “functional” words; and (4) modal operators.

2.1 The Core of Lexicon

As a core of lexicon for attitude analysis, we employ an Affect database and extended version of the SentiFul database developed by Neviarouskaya et al. (2009). The affective features of each emotion-related word are encoded using nine emotion labels (‘anger’, ‘disgust’, ‘fear’, ‘guilt’, ‘interest’, ‘joy’, ‘sadness’, ‘shame’, and ‘surprise’) and corresponding emotion intensities that range from 0.0 to 1.0. The original version of SentiFul database, which contains sentiment-conveying adjectives, adverbs, nouns, and verbs annotated by sentiment polarity, polarity scores and weights, was manually extended using attitude labels. Some examples of annotated attitude-conveying words are listed in Table 1. It is important to note here that some words may express different attitude types (affect, judgment, appreciation) depending on context; such lexical entries were annotated by all possible categories.

POS	Word	Category	Intensity
adjective	*honorable*	POS jud	0.3
	unfriendly	NEG aff (sadness)	0.5
		NEG jud	0.5
		NEG app	0.5
adverb	*gleefully*	POS aff (joy)	0.9
noun	*abnormality*	NEG app	0.25
verb	*frighten*	NEG aff (fear)	0.8
	desire	POS aff (interest)	1.0
		POS aff (joy)	0.5

Table 1. Examples of attitude-conveying words and their annotations.

2.2 Modifiers and Functional Words

We collected 138 modifiers that have an impact on contextual attitude features of related words, phrases, or clauses. They include:

1. Adverbs of degree (e.g., ‘*significantly*’, ‘*slightly*’ etc.) and affirmation (e.g., ‘*absolutely*’, ‘*seemingly*’) that have an influence on the strength of the attitude of related words. Two annotators gave coefficients for intensity degree strengthening or weakening (from 0.0 to 2.0) to each adverb, and the result was averaged (e.g., coeff(‘*slightly*’) = 0.2).
2. Negation words (e.g., ‘*never*’, ‘*nothing*’ etc.) reversing the polarity of related statement.
3. Adverbs of doubt (e.g., ‘*scarcely*’, ‘*hardly*’ etc.) and falseness (e.g., ‘*wrongly*’ etc.) reversing the polarity of related statement.
4. Prepositions (e.g., ‘*without*’, ‘*despite*’ etc.) neutralizing the attitude of related words.
5. Condition operators (e.g., ‘*if*’, ‘*even though*’ etc.) that neutralize the attitude of related words.

We distinguish two types of “functional” words that influence contextual attitude and its strength:

1. *Intensifying* adjectives (e.g., ‘*rising*’, ‘*rapidly-growing*’), nouns (e.g., ‘*increase*’), and verbs (e.g., ‘*to grow*’, ‘*to rocket*’) that increase the strength of attitude of related words.
2. *Reversing* adjectives (e.g., ‘*reduced*’), nouns (e.g., ‘*termination*), and verbs (e.g., ‘*to decrease*’, ‘*to limit*’, ‘*to diminish*’), which reverse the prior polarity of related words.

2.3 Modal Operators

Consideration of the modal operators in the tasks of opinion mining and attitude analysis is very important, as they indicate a degree of person’s belief in the truth of the proposition, which is subjective in nature (Hoye, 1997). Modals are distinguished by their *confidence level*. We collected modal operators of two categories: modal verbs (13 verbs) and modal adverbs (61 adverbs). Three human annotators assigned the *confidence level* ranging from 0.0 to 1.0 to each modal verb and adverb; these ratings were averaged (e.g., conf(‘*vaguely*’) = 0.17, conf(‘*arguably*’) = 0.63, conf(‘*would*’) = 0.8, conf(‘*veritably*’) = 1.0).

3 Compositionality Principle

Our algorithm for attitude classification is designed based on the *compositionality principle*, according to which we determine the attitudinal meaning of a sentence by composing the pieces that correspond to lexical units or other linguistic constituent types governed by the rules of *polarity reversal, aggregation (fusion)*, *propagation*, *domination*, *neutralization*, and *intensification*, at various grammatical levels.

Polarity reversal means that a phrase or statement containing an attitude-conveying term/phrase with prior positive polarity becomes negative, and vice versa. The rule of *polarity reversal* is applied in three cases: (1) negation word-modifier in relation with an attitude-conveying statement (e.g., '*never*' & POS('*succeed*') => NEG('*never succeed*')); (2) adverb of doubt in relation with attitude-conveying statement (e.g., '*scarcely*' & POS('*relax*') => NEG('*scarcely relax*')); (3) functional word of *reversing* type in relation with attitude-conveying statement (e.g., adjective '*reduced*' & POS('*enthusiasm*') => NEG('*reduced enthusiasm*')). In the case of judgment and appreciation, the use of the *polarity reversal* rule is straightforward ('POS jud' <=> 'NEG jud', 'POS app' <=> 'NEG app'). However, it is not trivial to find pairs of opposite emotions in the case of a fine-grained classification, except for 'joy' and 'sadness'. Therefore, we assume that (1) the opposite emotion for three positive emotions, i.e. 'interest', 'joy', and 'surprise', is 'sadness' ('POS aff' => 'sadness'); and (2) the opposite emotion for six negative emotions, i.e. 'anger', 'disgust', 'fear', 'guilt', 'sadness', and 'shame', is 'joy' ('NEG aff' => 'joy').

The rules of *aggregation (fusion)* are as follows: (1) if polarities of attitude-conveying terms in adjective-noun, noun-noun, adverb-adjective, adverb-verb phrases have opposite directions, mixed polarity with dominant polarity of a pre-modifier is assigned to the phrase (e.g., POS('*beautiful*') & NEG('*fight*') => POS-neg('*beautiful fight*'); NEG('*shamelessly*') & POS('*celebrate*') => NEG-pos('*shamelessly celebrate*')); otherwise (2) the resulting polarity is based on the equal polarities of terms, and the strength of attitude is measured as a maximum between polarity scores (intensities) of terms (*max*(score1,score2)).

The rule of *propagation* is useful, as proposed in (Nasukawa and Yi, 2003), for the task of the detection of local sentiments for given subjects. "Propagation" verbs propagate the sentiment towards the arguments; "transfer" verbs transmit sentiments among the arguments. The rule of *propagation* is applied when a verb of "propagation" or "transfer" type is used in a phrase/clause and sentiment of an argument that has prior neutral polarity needs to be investigated (e.g., PROP-POS('*to admire*') & '*his behaviour*' => POS('*his behaviour*'); '*Mr. X*' & TRANS('*supports*') & NEG('*crime business*') => NEG('*Mr. X*')).

The rules of *domination* are as follows: (1) if polarities of a verb (this rule is applied only for certain classes of verbs) and an object in a clause have opposite directions, the polarity of verb is prevailing (e.g., NEG('*to deceive*') & POS('*hopes*') => NEG('*to deceive hopes*')); (2) if compound sentence joints clauses using coordinate connector '*but*', the attitude features of a clause following after the connector are dominant (e.g., 'NEG(*It was hard to climb a mountain all night long*), *but* POS(*a magnificent view rewarded the traveler at the morning*).' => POS(whole sentence)).

The rule of *neutralization* is applied when preposition-modifier or condition operator relate to the attitude-conveying statement (e.g., '*despite*' & NEG('*worries*') => NEUT('*despite worries*')).

The rule of *intensification* means strengthening or weakening of the polarity score (intensity), and is applied when:

1. adverb of degree or affirmation relates to attitude-conveying term (e.g., Pos_score('*happy*') < Pos_score('*extremely happy*'));
2. adjective or adverb is used in a comparative or superlative form (e.g., Neg_score('*sad*') < Neg_score('*sadder*') < Neg_score ('*saddest*')).

Our method is capable of processing sentences of different complexity, including simple, compound, complex (with complement and relative clauses), and complex-compound sentences. We employ Connexor Machinese Syntax parser (`http://www.connexor.eu/`) that returns lemmas, parts of speech, dependency functions, syntactic function tags, and morphological tags. When handling the parser output, we represent the sentence as a set of primitive clauses. Each clause might include Subject formation, Verb formation and Object formation, each of which may consist of a main element (subject, verb, or object) and its attributives and complements. For the processing of complex or compound sentences, we build a so-called "relation matrix", which contains information about dependences (e.g., coordination, subordination, condition, contingency, etc.) between different clauses in a sentence. While applying the compositionality principle, we consecutively assign attitude features to words, phrases, formations, clauses, and finally, to the whole sentence.

4 Consideration of the Semantics of Verbs

All sentences must include a verb, because the verb tells us what action the subject is performing and object is receiving. In order to elaborate rules for attitude analysis based on the semantics of verbs, we investigated VerbNet (Kipper et al., 2007), the largest on-line verb lexicon that is organized into verb classes characterized by syntactic and semantic coherence among members of a class. Based on the thorough analysis of 270 first-level classes of VerbNet and their members, 73 verb classes (1) were found useful for the task of attitude analysis, and (2) were further classified into 22 classes differentiated by the role that members play in attitude analysis and by rules applied to them. Our classification is shown in Table 2.

For each of our verb classes, we developed set of rules that are applied to attitude analysis on the phrase/clause-level. Some verb classes (e.g., "*Psychological state or emotional reaction*", "*Judgment*", "*Bodily state and damage to the body*", "*Preservation*" etc.) include verbs annotated by attitude type, prior polarity orientation, and the strength of attitude. The attitude features of phrases that involve positively or negatively charged verbs from such classes are context-sensitive and are defined by means of rules designed for each of the class.

As an example, we provide short description and rules elaborated for the subclass "*Object-centered (oriented) emotional state*".

Features: subject experiences emotions towards some stimulus; verb prior polarity: positive or negative; context-sensitive.

Verb-Object rules (subject is ignored):

1. "Interior perspective" (subject's inner emotion state or attitude):

S & V+('*admires*') & O+('*his brave heart*') => (fusion, *max*(V_score,O_score)) => 'POS aff'.

S & V+('*admires*') & O-('*mafia leader*') => (verb valence dominance, V_score) => 'POS aff'.

S & V-('*disdains*') & O+('*his honesty*') => (verb valence dominance, V_score) => 'NEG aff'.

S & V-('*disdains*') & O-('*criminal activities*') => (fusion, *max*(V_score,O_score)) => 'NEG aff'.

2. "Exterior perspective" (social/ethical judgment):

S & V+('*admires*') & O+('*his brave heart*') => (fusion, *max*(V_score,O_score)) => 'POS jud'.

S & V+('*admires*') & O-('*mafia leader*') => (verb valence reversal, *max*(V_score,O_score)) => 'NEG jud'.

S & V-('*disdains*') & O+('*his honesty*') => (verb valence dominance, *max*(V_score,O_score)) => 'NEG jud'.

S & V-('*disdains*') & O-('*criminal activities*') => (verb valence reversal, *max*(V_score,O_score)) => 'POS jud'.

Verb class (verb samples)
1 Psychological state or emotional reaction
1.1 Object-centered (oriented) emotional state (*adore*)
1.2 Subject-driven change in emotional state (trans.) (*charm*, *inspire*, *bother*)
1.3 Subject-driven change in emotional state (intrans.) (*appeal to*, *grate on*)
2 Judgment
2.1 Positive judgment (*bless*, *honor*)
2.2 Negative judgment (*blame*, *punish*)
3 Favorable attitude (*accept*, *allow*, *tolerate*)
4 Adverse (unfavorable) attitude (*discourage*, *forbid*)
5 Favorable or adverse calibratable changes of state (*grow*, *decline*)
6 Verbs of removing
6.1 Verbs of removing with neutral charge (*delete*)
6.2 Verbs of removing with negative charge (*expel*)
6.3 Verbs of removing with positive charge (*evacuate*)
7 Negatively charged change of state (*break*, *crush*)
8 Bodily state and damage to the body (*sicken*, *injure*)
9 Aspectual verbs
9.1 Initiation, continuation of activity, and sustaining (*begin*, *continue*, *maintain*)
9.2 Termination of activity (*quit*, *finish*)
10 Preservation (*defend*, *insure*)
11 Verbs of destruction and killing (*damage*, *poison*)
12 Disappearance (*disappear*, *die*)
13 Limitation and subjugation (*confine*, *restrict*)
14 Assistance (*succor*, *help*)
15 Obtaining (*win*, *earn*)
16 Communication indicator/reinforcement of attitude (*guess*, *complain*, *deny*)
17 Verbs of leaving (*abandon*, *desert*)
18 Changes in social status or condition (*canonize*)
19 Success and failure
19.1 Success (*succeed*, *manage*)
19.2 Failure (*fail*, *flub*)
20 Emotional nonverbal expression (*smile*, *weep*)
21 Social interaction (*marry*, *divorce*)
22 Transmitting verbs (*supply*, *provide*)

Table 2. Verb classes for attitude analysis.

3. In case of neutral object => attitude type and prior polarity of verb, verb score (V_score).

Verb-PP (prepositional phrase) rules:

1. In case of negatively charged verb and PP starting with '*from*' => verb dominance:

S & V-('*suffers*') & PP-('*from illness*') => interior: 'NEG aff'; exterior: 'NEG jud'.

S & V-('*suffers*') & PP+ ('*from love*') => interior: 'NEG aff'; exterior: 'NEG jud'.

2. In case of positively charged verb and PP starting with '*in*'/'*for*' => treat PP the same way as object (see above):

S & V+('*believes*') & PP-('*in evil*') => interior: 'POS aff'; exterior: 'NEG jud'.

S & V+('*believes*') & PP+('*in kindness*') => interior: 'POS aff'; exterior: 'POS jud'.

In the majority of rules the strength of attitude is measured as a maximum between attitude scores (for example, the attitude conveyed by '*to suffer from grave illness*' is stronger than that of '*to suffer from slight illness*').

In contrast to the rules of "*Object-centered (oriented) emotional state*" subclass, which ignore attitude features of a subject in a sentence, the rules elaborated for the "*Subject-driven change in emotional state (trans.)*" disregard the attitude features of object, as in sentences involving members of this subclass object experiences emotion, and subject causes the emotional state. For example (due to limitation of space, here and below we provide only some cases):

S('*Classical music*') & V+('*calmed*') & O-('*disobedient child*') => interior: 'POS aff'; exterior: 'POS app'.

S-('*Fatal consequences of GM food intake*') & V-('*frighten*') & O('*me*') => interior: 'NEG aff'; exterior: 'NEG app'.

The Verb-Object rules for the "*Judgment*" subclasses, namely "*Positive judgment*" and "*Negative judgment*", are very close to those defined for the subclass "*Object-centered (oriented) emotional state*". However, Verb-PP rules have some specifics: for both positive and negative judgment verbs, we treat PP starting with '*for*'/'*of*'/'*as*' the same way as object in Verb-Object rules. For example:

S('*He*') & V-('*blamed*') & O+('*innocent person*') => interior: 'NEG jud'; exterior: 'NEG jud'.

S('*They*') & V-('*punished*') & O('*him*') & PP-('*for his misdeed*') => interior: 'NEG jud'; exterior: 'POS jud'.

Verbs from classes "*Favorable attitude*" and "*Adverse (unfavorable) attitude*" have prior neutral polarity and positive or negative reinforcement, correspondingly, that means that they only impact on the polarity and strength of non-neutral phrase (object in a sentence written in active voice, or subject in a sentence written in passive voice, or PP in case of some verbs). The rules are:

1. If verb belongs to the "*Favorable attitude*" class and the polarity of phrase is not neutral, then the attitude score of the phrase is intensified (symbol '^' means intensification):

S('*They*') & [V pos. reinforcement]('*elected*') & O+('*fair judge*') => 'POS app'; O_score^.

S('*They*') & [V pos. reinforcement]('*elected*') & O-('*corrupt candidate*') => 'NEG app'; O_score^.

2. If verb belongs to the "*Adverse (unfavorable) attitude*" class and the polarity of phrase is not neutral, then the polarity of phrase is reversed and score is intensified:

S('*They*') & [V neg. reinforcement]('*prevented*') & O-('*the spread of disease*') => 'POS app'; O_score^.

S+('*His achievements*') & [V neg. reinforcement]('*were overstated*') => 'NEG app'; S_score^.

Below are examples of processing the sentences with verbs from "*Verbs of removing*" class.

"*Verbs of removing with neutral charge*":

S('*The tape-recorder*') & [V neutral rem.]('*automatically ejects*') & O-neutral('*the tape*') => neutral.

S('*The safety invention*') & [V neutral rem.]('*ejected*') & O('*the pilot*') & PP-('*from burning plane*') => 'POS app'; PP_score^.

"*Verbs of removing with negative charge*":

S('*Manager*') & [V neg. rem.]('*fired*') & O-('*careless employee*') & PP('*from the company*') => 'POS app'; *max*(V_score,O_score).

"*Verbs of removing with positive charge*":

S('*They*') & [V pos. rem.]('*evacuated*') & O('*children*') & PP-('*from dangerous place*') => 'POS app'; *max*(V_score,PP_score).

Along with modal verbs and modal adverbs, members of the "*Communication indicator/reinforcement of attitude*" verb class also indicate the confidence level or degree of certainty concerning given opinion. Features are: subject (communicator) expresses statement with/without attitude; statement is PP starting with '*of*', '*on*', '*against*', '*about*', '*concerning*', '*regarding*', '*that*', '*how*' etc.; ground: positive or negative; reinforcement: positive or negative. The rules are:

1. If the polarity of expressed statement is neutral, then the attitude is neutral:

S('*Professor*') & [V pos. ground, pos. reinforcement, confidence:0.83]('*dwelled*') & PP-neutral('*on a question*') => neutral.

2. If the polarity of expressed statement is not neutral and the reinforcement is positive, then the score of the statement (PP) is intensified:

S('*Jane*') & [V neg. ground, pos. reinforcement, confidence:0.8]('*is complaining*') & PP-('*of a headache again*') => 'NEG app'; PP_score^; confidence:0.8.

3. If the polarity of expressed statement is not neutral and reinforcement is negative, then the polarity of the statement (PP) is reversed and score is intensified:

S('*Max*') & [V neg. ground, neg. reinforcement, confidence:0.2]('*doubt*') & PP-{'*that*' S+('*his good fortune*') & [V termination]('*will ever end*')} => 'POS app'; PP_score^; confidence:0.2.

In the last example, to measure the sentiment of PP, we apply rule for the verb '*end*' from the "*Termination of activity*" class, which reverses the non-neutral polarity of subject (in intransitive use of verb) or object (in transitive use of verb). For example, the polarity of both sentences '*My whole enthusiasm and excitement disappear like a bubble touching a hot needle*' and '*They discontinued helping children*' is negative.

5 Decision on Attitude Label

The decision on the most appropriate final label for the clause, in case @AM annotates it using different attitude types according to the words with multiple annotations (e.g., see word '*unfriendly*' in Table 1) or based on the availability of the words conveying different attitude types, is made based on the analysis of:

1) morphological tags of nominal heads and their premodifiers in the clause (e.g., first person pronoun, third person pronoun, demonstrative pronoun, nominative or genitive noun, etc.);

2) the sequence of hypernymic semantic relations of a particular noun in WordNet (Miller, 1990), which allows to determine its conceptual domain (e.g., "person, human being", "artifact", "event", etc.);

3) the annotations from the Stanford Named Entity Recognizer (Finkel et al. 2005) that labels PERSON, ORGANIZATION, and LOCATION entities.

For ex., '*I feel highly unfriendly attitude towards me*' conveys emotion ('NEG aff': 'sadness'), while '*The shop assistant's behavior was really unfriendly*' and '*Plastic bags are environment unfriendly*' express judgment ('NEG jud') and appreciation ('NEG app'), correspondingly.

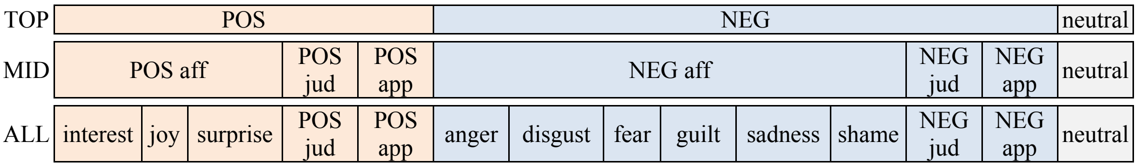
6 Evaluation

For the experiments, we used our own data set, as, to the best of our knowledge, there is no publicly available data set of sentences annotated by the fine-grained labels proposed in our work. In order to evaluate the performance of our algorithm, we created the data set of sentences extracted from personal stories about life experiences that were anonymously published on the *Experience Project* website (`www.experienceproject.com`), where people share personal experiences, thoughts, opinions, feelings, passions, and confessions through the network of personal stories. With over 4 million experiences accumulated (as of February 2010), *Experience Project* is a perfect source for researchers interested in studying different types of attitude expressed through text.

6.1 Data Set Description

For our experiment we extracted 1000 sentences[1] from various stories grouped by topics within 13 different categories, such as "Arts and entertainment", "Current events", "Education", "Family and friends", "Health and wellness", "Relationships and romance" and others, on the *Experience Project* website. Sentences were collected from 358 distinct topic groups, such as "I still remember September 11", "I am intelligent but airheaded", "I think bullfighting is cruel", "I quit smoking", "I am a fashion victim", "I was adopted" and others.

[1] This annotated data set is freely available upon request.



<table>
<tr><td>TOP</td><td colspan="5">POS</td><td colspan="8">NEG</td><td>neutral</td></tr>
<tr><td>MID</td><td colspan="3">POS aff</td><td>POS jud</td><td>POS app</td><td colspan="6">NEG aff</td><td>NEG jud</td><td>NEG app</td><td>neutral</td></tr>
<tr><td>ALL</td><td>interest</td><td>joy</td><td>surprise</td><td>POS jud</td><td>POS app</td><td>anger</td><td>disgust</td><td>fear</td><td>guilt</td><td>sadness</td><td>shame</td><td>NEG jud</td><td>NEG app</td><td>neutral</td></tr>
</table>

Figure 1. Hierarchy of attitude labels.

We considered three hierarchical levels of attitude labels in our experiment (see Figure 1). Three independent annotators labeled the sentences with one of 14 categories from the ALL level and a corresponding score (the strength or intensity value). These annotations were further interpreted using labels from the MID and the TOP levels. Fleiss' Kappa coefficient was used as a measure of reliability of human raters' annotations. The agreement coefficient on 1000 sentences was 0.53 on ALL level, 0.57 on MID level, and 0.73 on TOP level.

Only those sentences, on which at least two out of three human raters completely agreed, were included in the gold standards for our experiment. Three gold standards were created according to the hierarchy of attitude labels. Fleiss' Kappa coefficients are 0.62, 0.63, and 0.74 on ALL, MID, and TOP levels, correspondingly. Table 3 shows the distributions of labels in the gold standards.

ALL level		**MID level**	
Label	**Number**	**Label**	**Number**
anger	45	POS aff	233
disgust	21	NEG aff	332
fear	54	POS jud	66
guilt	22	NEG jud	78
interest	84	POS app	100
joy	95	NEG app	29
sadness	133	neutral	87
shame	18	**total**	925
surprise	36		
POS jud	66	**TOP level**	
NEG jud	78	**Label**	**Number**
POS app	100	POS	437
NEG app	29	NEG	473
neutral	87	neutral	87
total	868	**total**	997

Table 3. Label distributions in gold standards.

6.2 Results

The results of a simple method selecting the attitude label with the maximum intensity from the annotations of sentence tokens found in the database were considered as the baseline. After processing each sentence from the data set by the baseline method and our @AM system, we measured averaged accuracy, precision, recall, and F-score for each label in ALL, MID, and TOP levels. The results are shown in Table 4.

As seen from the obtained results, our algorithm performed with high accuracy significantly surpassing the baselines in all levels of attitude hierarchy, thus demonstrating the contribution of the sentence parsing and our hand-crafted rules to the reliable recognition of attitude from text. Two-tailed t-tests with significance level of 0.05 showed that the differences in accuracy between the baseline method and our @AM system are statistically significant ($p<0.001$) in fine-grained as well as coarse-grained classifications.

In the case of fine-grained attitude recognition (ALL level), the highest precision was obtained for 'shame' (0.923) and 'NEG jud' (0.889), while the highest recall was received for 'sadness' (0.917) and 'joy' (0.905) emotions at the cost of low precision (0.528 and 0.439, correspondingly). The algorithm performed with the worst results in recognition of 'NEG app' and 'neutral'.

The analysis of a confusion matrix for the ALL level revealed the following top confusions of our system: (1) 'anger', 'fear', 'guilt', 'shame', 'NEG jud', 'NEG app' and 'neutral' were predominantly incorrectly predicted as 'sadness' (for ex., @AM resulted in 'sadness' for the sentence '*I know we have several months left before the election, but I am already sick and tired of seeing the ads on TV*', while human annotations were 'anger'/'anger'/'disgust'); (2) 'interest', 'POS jud' and 'POS app' were mostly confused with 'joy' by our algorithm (e.g., @AM classified the sentence '*It's one of those life changing artifacts that we must have in order to have happier, healthier lives*' as 'joy'(-ful), while human annotations were 'POS app'/'POS app'/'interest').

Our system achieved high precision for all categories on the MID level (Table 4), with the exception of 'NEG app' and 'neutral', although

Level	Label	Baseline method				@AM			
		Accuracy	Precision	Recall	F-score	Accuracy	Precision	Recall	F-score
ALL	anger	0.437	0.742	0.511	0.605	**0.621**	0.818	0.600	0.692
	disgust		0.600	0.857	0.706		0.818	0.857	0.837
	fear		0.727	0.741	0.734		0.768	0.796	0.782
	guilt		0.667	0.364	0.471		0.833	0.455	0.588
	interest		0.380	0.357	0.368		0.772	0.524	0.624
	joy		0.266	0.579	0.364		0.439	0.905	0.591
	sadness		0.454	0.632	0.528		0.528	0.917	0.670
	shame		0.818	0.500	0.621		0.923	0.667	0.774
	surprise		0.625	0.694	0.658		0.750	0.833	0.789
	POS jud		0.429	0.227	0.297		0.824	0.424	0.560
	NEG jud		0.524	0.141	0.222		0.889	0.410	0.561
	POS app		0.349	0.150	0.210		0.755	0.400	0.523
	NEG app		0.250	0.138	0.178		0.529	0.310	0.391
	neutral		0.408	0.483	0.442		0.559	0.437	0.490
MID	POS aff	0.524	0.464	0.695	0.557	**0.709**	0.668	0.888	0.762
	NEG aff		0.692	0.711	0.701		0.765	0.910	0.831
	POS jud		0.405	0.227	0.291		0.800	0.424	0.554
	NEG jud		0.458	0.141	0.216		0.842	0.410	0.552
	POS app		0.333	0.150	0.207		0.741	0.400	0.519
	NEG app		0.222	0.138	0.170		0.474	0.310	0.375
	neutral		0.378	0.483	0.424		0.514	0.437	0.472
TOP	POS	0.732	0.745	0.796	0.770	**0.879**	0.918	0.920	0.919
	NEG		0.831	0.719	0.771		0.912	0.922	0.917
	neutral		0.347	0.483	0.404		0.469	0.437	0.452

Table 4. Results of the evaluation of performance of the baseline method and @AM system.

high recall was obtained only in the case of categories related to affect ('POS aff', 'NEG aff'). These results indicate that affect sensing is easier than recognition of judgment or appreciation from text. TOP level results (Table 4) show that our algorithm classifies sentences that convey positive or negative sentiment with high accuracy (92% and 91%, correspondingly). On the other hand, 'neutral' sentences still pose a challenge.

The analysis of errors revealed that system requires common sense or additional context to deal with sentences like '*All through my life I've felt like I'm second fiddle*' (gold standard: 'sadness'; @AM: 'neutral') or '*For me every minute on my horse is alike an hour in heaven!*' (gold standard: 'joy'; @AM: 'neutral').

We also evaluated the system performance with regard to attitude intensity estimation. The percentage of attitude-conveying sentences (not considering neutral ones), on which the result of our system conformed to the fine-grained gold standard (ALL level), according to the measured distance between intensities given by human raters (averaged values) and those obtained by our system is shown in Table 5. As seen from the table, our system achieved satisfactory results in estimation of the strength of attitude expressed through text.

Range of intensity difference	Percent of sentences, %
[0.0 – 0.2]	55.5
(0.2 – 0.4]	29.5
(0.4 – 0.6]	12.2
(0.6 – 0.8]	2.6
(0.8 – 1.0]	0.2

Table 5. Results on intensity.

7 Conclusions

In this paper we introduced @AM, which is so far, to the best of our knowledge, the only system classifying sentences using fine-grained attitude types, and extensively dealing with the semantics of verbs in attitude analysis. Our composition approach broadens the coverage of sentences with complex contextual attitude. The evaluation results indicate that @AM achieved reliable results in the task of textual attitude analysis. The limitations include dependency on lexicon and on accuracy of the parser. The primary objective for the future research is to develop a method for the extraction of reasons behind the expressed attitude.

References

Alm, Cecilia O. 2008. *Affect in Text and Speech*. PhD Dissertation. University of Illinois at Urbana-Champaign.

Aman, Saima, and Stan Szpakowicz. 2008. Using Roget's Thesaurus for Fine-Grained Emotion Recognition. *Proceedings of the Third International Joint Conference on Natural Language Processing*, Hyderabad, India, pp. 296-302.

Bhowmick, Plaban K., Anupam Basu, and Pabitra Mitra. 2009. Reader Perspective Emotion Analysis in Text through Ensemble based Multi-Label Classification Framework. *Computer and Information Science*, 2 (4): 64-74.

Boucouvalas, Anthony C. 2003. Real Time Text-to-Emotion Engine for Expressive Internet Communications. *Being There: Concepts, Effects and Measurement of User Presence in Synthetic Environments*, Ios Press, pp. 306-318.

Chaumartin, Francois-Regis. 2007. UPAR7: A Knowledge-based System for Headline Sentiment Tagging. *Proceedings of the SemEval-2007 International Workshop*, pp. 422-425.

Choi, Yejin, and Claire Cardie. 2008. Learning with Compositional Semantics as Structural Inference for Subsentential Sentiment Analysis. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 793-801.

Chuang, Ze-Jing, and Chung-Hsien Wu. 2004. Multimodal Emotion Recognition from Speech and Text. *Computational Linguistic and Chinese Language Processing*, 9(2): 45-62.

Finkel, Jenny R., Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *Proceedings of the 43nd Annual Meeting of the ACL*, pp. 363-370.

Hoye, Leo. 1997. *Adverbs and Modality in English*. New York: Addison Wesley Longman Inc.

Izard, Carroll E. 1971. *The Face of Emotion*. New York: Appleton-Century-Crofts.

Kipper, Karin, Anna Korhonen, Neville Ryant, and Martha Palmer. 2007. A Large-scale Classification of English Verbs. *Language Resources and Evaluation*, 42 (1): 21-40.

Kozareva, Zornitsa, Borja Navarro, Sonia Vazquez, and Andres Montoyo, A. 2007. UA-ZBSA: A Headline Emotion Classification through Web Information. *Proceedings of the SemEval-2007 International Workshop*, pp. 334-337.

Liu, Hugo, Henry Lieberman, and Ted Selker. 2003. A Model of Textual Affect Sensing Using Real-World Knowledge. *Proceedings of IUI-2003*, pp. 125-132.

Martin, James R., and Peter R.R. White. 2005. *The Language of Evaluation: Appraisal in English*. Palgrave, London, UK.

Miller, George A. 1990. WordNet: An On-line Lexical Database. *International Journal of Lexicography*, Special Issue, 3 (4): 235-312.

Moilanen, Karo, and Stephen Pulman. 2007. Sentiment Composition. *Proceedings of the Recent Advances in Natural Language Processing International Conference*, pp. 378-382.

Nasukawa, Tetsuya, and Jeonghee Yi. 2003. Sentiment Analysis: Capturing Favorability using Natural Language Processing. *Proceedings of the 2nd International Conference on Knowledge Capture*, pp. 70-77.

Neviarouskaya, Alena, Helmut Prendinger, and Mitsuru Ishizuka. 2009. SentiFul: Generating a Reliable Lexicon for Sentiment Analysis. *Proceedings of the International Conference on Affective Computing and Intelligent Interaction*, IEEE, Amsterdam, Netherlands, pp. 363-368.

Strapparava, Carlo, and Rada Mihalcea. 2008. Learning to Identify Emotions in Text. *Proceedings of the 2008 ACM Symposium on Applied Computing*, Fortaleza, Brazil, pp. 1556-1560.

Strapparava, Carlo, Alessandro Valitutti, and Oliviero Stock. 2007. Dances with Words. *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1719-1724.

Subrahmanian, V.S., and Diego Reforgiato. 2008. AVA: Adjective-Verb-Adverb Combinations for Sentiment Analysis. *Intelligent Systems*, IEEE, 23 (4): 43-50.

Taboada, Maite, and Jack Grieve. 2004. Analyzing Appraisal Automatically. *Proceedings of AAAI Spring Symposium on Exploring Attitude and Affect in Text*, pp.158-161.

Whitelaw, Casey, Navendu Garg, and Shlomo Argamon. 2005. Using Appraisal Groups for Sentiment Analysis. *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM, Bremen, Germany, pp. 625-631.

Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-level Sentiment Analysis. *Proceedings of HLT-EMNLP-2005*, ACL, pp. 347-354.

Nonparametric Word Segmentation for Machine Translation

ThuyLinh Nguyen Stephan Vogel Noah A. Smith
Language Technologies Institute
Carnegie Mellon University
{thuylinh,vogel,nasmith}@cs.cmu.edu

Abstract

We present an unsupervised word segmentation model for machine translation. The model uses existing monolingual segmentation techniques and models the joint distribution over source sentence segmentations and alignments to the target sentence. During inference, the monolingual segmentation model and the bilingual word alignment model are coupled so that the alignments to the target sentence guide the segmentation of the source sentence. The experiments show improvements on Arabic-English and Chinese-English translation tasks.

1 Introduction

In statistical machine translation, the smallest unit is usually the *word*, defined as a token delimited by spaces. Given a parallel corpus of source and target text, the training procedure first builds a word alignment, then extracts phrase pairs from this word alignment. However, in some languages (e.g., Chinese) there are no spaces between words.

The same problem arises when translating between two very different languages, such as from a language with rich morphology like Hungarian or Arabic to a language with poor morphology like English or Chinese. A single word in a morphologically rich language is often the composition of several morphemes, which correspond to separate words in English.[1]

Often some preprocessing is applied involving word segmentation or morphological analysis of the source and/or target text. Such preprocessing tokenizes the text into morphemes or words, which linguists consider the smallest meaning-bearing units of the language. Take as an example the Arabic word "fktbwha" and its English translation "so they wrote it". The preferred segmentation of "fktbwha" would be "f-ktb-w-ha (*so-wrote-they-it*)," which would allow for a one-to-one mapping between tokens in the two languages. However, the translation of the phrase in Hebrew is "wktbw ath". Now the best segmentation of the Arabic words would be "fktbw-ha," corresponding to the two Hebrew words. This example shows that there may not be one correct segmentation that can be established in a preprocessing step. Rather, tokenization depends on the language we want to translate into and needs to be tied in with the alignment process. In short, we want to find the tokenization yielding the best alignment, and thereby the best translation system.

We propose an unsupervised tokenization method for machine translation by formulating a generative Bayesian model to "explain" the bilingual training data. Generation of a sentence pair is described as follows: first a monolingual tokenization model generates the source sentence, then the alignment model generates the target sentence through the alignments with the source sentence. Breaking this generation process into two steps provides flexibility to incorporate existing monolingual morphological segmentation models such as those of Mochihashi et al. (2009) or Creutz and Lagus (2007). Using nonparametric

[1] We will use the terms *word segmentation*, *morphological analysis*, and *tokenization* more or less interchangeably.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 815–823,
Beijing, August 2010

models and the Bayesian framework makes it possible to incorporate linguistic knowledge as prior distributions and obtain the posterior distribution through inference techniques such as MCMC or variational inference.

As new test source sentences do not have translations which can help to infer the best segmentation, we *decode* the source string according to the posterior distribution from the inference step.

In summary, our segmentation technique consists of the following steps:

- A joint model of segmented source text and its target translation.
- Inference of the posterior distribution of the model given the training data.
- A decoding algorithm for segmenting source text.
- Experiments in translation using the preprocessed source text.

Our experiments show that the proposed segmentation method leads to improvements on Arabic-English and Chinese-English translation tasks.

In the next section we will discuss related work. Section 3 will describe our model in detail. The inference will be covered in Section 4, and decoding in Section 5. Experiments and results will be presented in Section 6.

2 Related Work

The problem of segmentation for machine translation has been studied extensively in recent literature. Most of the work used some linguistic knowledge about the source and the target languages (Nießen and Ney, 2004; Goldwater and McClosky, 2005). Sadat and Habash (2006) experimented with a wide range of tokenization schemes for Arabic-English translation. These experiments further show that even for a single language pair, different tokenizations are needed depending on the training corpus size. The experiments are very expensive to conduct and do not generalize to other language pairs. Recently, Dyer (2009) created manually crafted lattices for a subset of source words as references for segmentation when translating into English, and then learned the segmentation of the source words to optimize the translation with respect to these references. He showed that the parameters of the model can be applied to similar languages when translating into English. However, manually creating these lattices is time-consuming and requires a bilingual person with some knowledge of the underlying statistical machine translation system.

There have been some attempts to apply unsupervised methods for tokenization in machine translation (Chung and Gildea, 2009; Xu et al., 2008). The alignment model of Chung and Gildea (2009) forces every source word to align with a target word. Xu et al. (2008) modeled the source-to-null alignment as in the source word to target word model. Their models are special cases of our proposed model when the source model[2] is a unigram model. Like Xu et al. (2008), we use Gibbs sampling for inference. Chung and Gildea (2009) applied efficient dynamic programming-based variational inference algorithms.

We benefit from existing unsupervised monolingual segmentation. The source model uses the nested Pitman-Yor model as described by Mochihashi et al. (2009). When sampling each potential word boundary, our inference technique is a bilingual extension of what is described by Goldwater et al. (2006) for monolingual segmentation.

Nonparametric models have received attention in machine translation recently. For example, DeNero et al. (2008) proposed a hierarchical Dirichlet process model to learn the weights of phrase pairs to address the degeneration in phrase extraction. Teh (2006) used a hierarchical Pitman-Yor process as a smoothing method for language models.

Recent work on multilingual language learning successfully used nonparametric models for language induction tasks such as grammar induction (Snyder et al., 2009; Cohen et al., 2010), morphological segmentation (Goldwater et al., 2006; Snyder and Barzilay, 2008), and part-of-speech tagging (Goldwater and Griffiths, 2007; Snyder et al.,

[2] Note that "source model" here means a model of source text, not a source model in the noisy channel paradigm.

2008).

3 Models

We start with the generative process for a source sentence and its alignment with a target sentence. Then we describe individual models employed by this generation scheme.

3.1 Generative Story

A source sentence is a sequence of word tokens, and each word is either aligned or not aligned. We focus only on the segmentation problem and not reordering source words; therefore, the model will not generate the order of the target word tokens. A sentence pair and its alignment are captured by four components:

- a sequence of words in the source sentence,
- a set of null-aligned source tokens,
- a set of null-aligned target tokens, and
- a set of (source word to target word) alignment pairs.

We will start with a high-level story of how the segmentation of the source sentence and the alignment are generated.

1. A source language monolingual segmentation model generates the source sentence.
2. Generate alignments:
   (a) Given the sequence of words of the source sentence already generated in step 1, the alignment model marks each source word as either aligned or unaligned. If a source word is aligned, the model also generates the target word.
   (b) Unaligned target words are generated.

The model defines the joint probability of a segmented source language sentence and its alignment. During inference, the two parts are coupled, so that the alignment will influence which segmentation is selected. However, there are several advantages in breaking the generation process into two steps.

First of all, in principle the model can incorporate any existing probabilistic monolingual segmentation to generate the source sentence. For example, the source model can be the nested Pitman-Yor process as described by Mochihashi et al. (2009), the minimum description length model presented by Creutz and Lagus (2007), or something else. Also the source model can incorporate linguistic knowledge from a rule-based or statistical morphological disambiguator.

The model generates the alignment after the source sentence with word boundaries already generated. Therefore, the alignment model can be any existing word alignment model (Brown et al., 1993; Vogel et al., 1996). Even though the choices of source model or alignment model can lead to different inference methods, the model we propose here is highly extensible. Note that we assume that the alignment consists of at most one-to-one mappings between source and target words, with null alignments possible on both sides.

Another advantage of a separate source model lies in the segmentation of an unseen test set. In section 5 we will show how to apply the source model distribution learned from training data to find the best segmentation of an unseen test set.

Notation and Parameters

We will use bold font for a sequence or bags of words and regular font for an individual word. A source sentence $\mathbf{s}$ is a sequence of $|\mathbf{s}|$ words s_i: $\left(s_1, \ldots, s_{|\mathbf{s}|}\right)$; the translation of sentence $\mathbf{s}$ is the target sentence $\mathbf{t}$ of $|\mathbf{t}|$ words $\left(t_1, \ldots, t_{|\mathbf{t}|}\right)$. In sentence $\mathbf{s}$ the list of unaligned words is $\mathbf{s}_{\text{nal}}$ and the list of aligned source words is $\mathbf{s}_{\text{al}}$. In the target sentence $\mathbf{t}$ the list of unaligned words is $\mathbf{t}_{\text{nal}}$ and the list of target words having one-to-one alignment with source words $\mathbf{s}_{\text{al}}$ is $\mathbf{t}_{\text{al}}$. The alignment $\mathbf{a}$ of $\mathbf{s}$ and $\mathbf{t}$ is represented by $\{\langle s_i, \mathsf{null}\rangle \mid s_i \in \mathbf{s}_{\text{nal}}\} \cup \{\langle s_i, t_{a_i}\rangle \mid s_i \in \mathbf{s}_{\text{al}}; t_{a_i} \in \mathbf{t}_{\text{al}}\} \cup \{\langle \mathsf{null}, t_j\rangle \mid t_j \in \mathbf{t}_{\text{nal}}\}$ where a_i denotes the index in $\mathbf{t}$ of the word aligned to s_i.

The probability of a sequence or a set is denoted by $\mathbb{P}(.)$, probability at the word level is $\mathsf{p}(.)$. For example, the probability of sentence $\mathbf{s}$ is $\mathbb{P}(\mathbf{s})$, the probability of a word s is $\mathsf{p}(s)$, the probability that the target word t aligns to an aligned source word s is $\mathsf{p}\left(t \mid s\right)$.

A sentence pair and its alignment are generated from the following models:

- The *source* model generates sentence s with probability $\mathbb{P}\left(\mathbf{s}\right)$.

- The *source-to-null* alignment model decides independently for each word s whether it is unaligned with probability $\mathsf{p}\left(\mathsf{null} \mid s_i\right)$ or aligned with probability: $1 - \mathsf{p}\left(\mathsf{null} \mid s_i\right)$. The probability of this step, for all source words, is: $\mathbb{P}\left(\mathsf{s}_{\text{nal}}, \mathsf{s}_{\text{al}} \mid \mathsf{s}\right) = \prod_{s_i \in \mathbf{s}_{\text{nal}}} \mathsf{p}\left(\mathsf{null} \mid s_i\right) \times \prod_{s_i \in \mathbf{s}_{\text{al}}} \left(1 - \mathsf{p}\left(\mathsf{null} \mid s_i\right)\right)$.

  We will also refer to the source-to-null model as the *deletion model*, since words in s_{nal} are effectively deleted for the purposes of alignment.

- The *source-to-target* alignment model generates a bag of target words $\mathbf{t}_{\text{al}}$ aligned to the source words s_{al} with probability: $\mathbb{P}\left(\mathbf{t}_{\text{al}} \mid \mathsf{s}_{\text{al}}\right) = \prod_{s_i \in \mathbf{s}_{\text{al}}; t_{a_i} \in \mathbf{t}_{\text{al}}} \mathsf{p}\left(t_{a_i} \mid s_i\right)$. Note that we do not need to be concerned with generating $\mathbf{a}$ explicitly, since we do not model word order on the target side.

- The *null-to-target* alignment model generates the list of unaligned target words $\mathbf{t}_{\text{nal}}$ given aligned target words $\mathbf{t}_{\text{al}}$ with $\mathbb{P}\left(\mathbf{t}_{\text{nal}} \mid \mathbf{t}_{\text{al}}\right)$ as follows:

  - Generate the number of unaligned target words $|\mathbf{t}_{\text{nal}}|$ given the number of aligned target words $|\mathbf{t}_{\text{al}}|$ with probability $\mathbb{P}\left(|\mathbf{t}_{\text{nal}}| \mid |\mathbf{t}_{\text{al}}|\right)$.
  - Generate $|\mathbf{t}_{\text{nal}}|$ unaligned words $t \in \mathbf{t}_{\text{nal}}$ independently, each with probability $\mathsf{p}\left(t \mid \mathsf{null}\right)$.

  The resulting null-to-target probability is therefore: $\mathbb{P}\left(\mathbf{t}_{\text{nal}} \mid \mathbf{t}_{\text{al}}\right) = \mathbb{P}\left(|\mathbf{t}_{\text{nal}}| \mid |\mathbf{t}_{\text{al}}|\right) \prod_{t \in \mathbf{t}_{\text{nal}}} \mathsf{p}\left(t \mid \mathsf{null}\right)$.

  We also call the null-to-target model the *insertion* model.

The above generation process defines the joint probability of source sentence s and its alignment $\mathbf{a}$ as follows:

$$\mathbb{P}\left(\mathbf{s}, \mathbf{a}\right) = \underbrace{\mathbb{P}\left(\mathbf{s}\right)}_{\text{source model}} \times \underbrace{\mathbb{P}\left(\mathbf{a} \mid \mathbf{s}\right)}_{\text{alignment model}} \quad (1)$$

$$\mathbb{P}\left(\mathbf{a} \mid \mathbf{s}\right) = \mathbb{P}\left(\mathbf{t}_{\text{al}} \mid \mathbf{s}_{\text{al}}\right) \times \mathbb{P}\left(\mathbf{t}_{\text{nal}} \mid \mathbf{t}_{\text{al}}\right) \quad (2)$$
$$\times \prod_{s_i \in \mathbf{s}_{\text{nal}}} \mathsf{p}\left(\mathsf{null} \mid s_i\right) \times \prod_{s_i \in \mathbf{s}_{\text{al}}} \left(1 - \mathsf{p}\left(\mathsf{null} \mid s_i\right)\right)$$

3.2 Source Model

Our generative process provides the flexibility of incorporating different monolingual models into the probability distribution of a sentence pair. In particular we use the existing state-of-the-art nested Pitman-Yor n-gram language model as described by Mochihashi et al. (2009). The probability of s is given by

$$\mathbb{P}\left(\mathsf{s}\right) = \mathbb{P}\left(|\mathsf{s}|\right) \prod_{i=1}^{|\mathsf{s}|} \mathsf{p}\left(s_i \mid s_{i-n}, \ldots, s_{i-1}\right) \quad (3)$$

where the n-gram probability is a hierarchical Pitman-Yor language model using $(n-1)$-gram as the base distribution.

At the unigram level, the model uses the base distribution $\mathsf{p}\left(s\right)$ as the infinite-gram character-level Pitman-Yor language model.

3.3 Modeling Null-Aligned Source Words

The probability that a source word aligns to null $\mathsf{p}\left(\mathsf{null} \mid s\right)$ is defined by a binomial distribution with Beta prior $\mathrm{Beta}\left(\alpha p, \alpha\left(1-p\right)\right)$, where α and p are model parameters. When $p \to 0$ and $\alpha \to \infty$ the probability $\mathsf{p}\left(\mathsf{null} \mid s\right)$ converges to 0 forcing each source words align to a target word. We fixed $p = 0.1$ and $\alpha = 20$ in our experiment.

Xu et al. (2008) view the null word as another target word, hence in their model the probability that a source word aligns to null can only depend on itself.

By modeling the source-to-null alignment separately, our model lets the distribution depend on the word's n-gram context as in the source model. $\mathsf{p}\left(\mathsf{null} \mid s_{i-n}, \ldots, s_i\right)$ stands for the probability that the word s_i is not aligned given its context $\left(s_{i-n}, \ldots, s_{i-1}\right)$.

The n-gram source-to-null distribution $\mathsf{p}\left(\mathsf{null} \mid s_{i-n}, \ldots, s_i\right)$ is defined similarly to

$\mathsf{p}(\mathsf{null} \mid s_i)$ definition above in which the base distribution p now becomes the $(n-1)$-gram: $\mathsf{p}(\mathsf{null} \mid s_{i-n+1}, \ldots, s_i)$.[3]

3.4 Source-Target Alignment Model

The probability $\mathsf{p}(t \mid s)$ that a target word t aligns to a source word s is a Pitman-Yor process:

$$t \mid s \sim \mathrm{PY}(d, \alpha, \mathsf{p}_0(t \mid s))$$

here d and α are the input parameters, and $\mathsf{p}_0(t \mid s)$ is the base distribution.

Let $|s, \cdot|$ denote the number of times s is aligned to any t in the corpus and let $|s, t|$ denote the number of times s is aligned to t anywhere in the corpus. And let $\mathsf{ty}(s)$ denote the number of different target words t the word s is aligned to anywhere in the corpus. In the Chinese Restaurant Process metaphor, there is one restaurant for each source word s, the s restaurant has $\mathsf{ty}(s)$ tables and total $|s, \cdot|$ customers; table t has $|s, t|$ customers.

Then, at a given time in the generative process for the corpus, we can write the probability that t is generated by the word s as:

- if $|s, t| > 0$:

$$\mathsf{p}(t \mid s) = \frac{|s, t| - d + [\alpha + d\mathsf{ty}(s)]\mathsf{p}_0(t \mid s)}{|s, \cdot| + \alpha}$$

- if $|s, t| = 0$:

$$\mathsf{p}(t \mid s) = \frac{[\alpha + d\mathsf{ty}(s)]\mathsf{p}_0(t \mid s)}{|s, \cdot| + \alpha}$$

For language pairs with similar character sets such as English and French, words with similar surface form are often translations of each other. The base distribution can be defined based on the edit distance between two words (Snyder and Barzilay, 2008).

We are working with diverse language pairs (Arabic-English and Chinese-English), so we use the base distribution as the flat distribution $\mathsf{p}_0(t \mid s) = \frac{1}{T}$; T is the number of distinct target words in the training set. In our experiment, the model parameters are $\alpha = 20$ and $d = .5$.

3.5 Modeling Null-Aligned Target Words

The null-aligned target words are modeled conditioned on previously generated target words as:

$$\mathbb{P}(\mathbf{t}_{\mathrm{nal}} \mid \mathbf{t}_{\mathrm{al}}) = \mathbb{P}(|\mathbf{t}_{\mathrm{nal}}| \mid |\mathbf{t}_{\mathrm{al}}|) \prod_{t \in \mathbf{t}_{\mathrm{nal}}} \mathsf{p}(t \mid \mathsf{null})$$

This model uses two probability distributions:

- the number of unaligned target words: $\mathbb{P}(|\mathbf{t}_{\mathrm{nal}}| \mid |\mathbf{t}_{\mathrm{al}}|)$, and
- the probability that each word in $\mathbf{t}_{\mathrm{nal}}$ is generated by null: $\mathsf{p}(t \mid \mathsf{null})$.

We model the number of unaligned target words similarly to the distribution in the IBM3 word alignment model (Brown et al., 1993). IBM3 assumes that each aligned target words generates a null-aligned target word with probability p_0 and fails to generate a target word with probability $1 - p_0$. So the parameter p_0 can be used to control the number of unaligned target words. In our experiments, we fix $p_0 = .05$. Following this assumption, the probability of $|\mathbf{t}_{\mathrm{nal}}|$ unaligned target words generated from $|\mathbf{t}_{\mathrm{al}}|$ words is: $\mathbb{P}(|\mathbf{t}_{\mathrm{nal}}| \mid |\mathbf{t}_{\mathrm{al}}|) = \binom{|\mathbf{t}_{\mathrm{al}}|}{|\mathbf{t}_{\mathrm{nal}}|} p_0^{|\mathbf{t}_{\mathrm{nal}}|}(1 - p_0)^{|\mathbf{t}_{\mathrm{al}}| - |\mathbf{t}_{\mathrm{nal}}|}$.

The probability that a target word t aligns to null, $\mathsf{p}(t \mid \mathsf{null})$, also has a Pitman-Yor process prior. The base distribution of the model is similar to the source-to-target model's base distribution which is the flat distribution over target words.

4 Inference

We have defined a probabilistic generative model to describe how a corpus of alignments and segmentations can be generated jointly. In this section we discuss how to obtain the posterior distributions of the missing alignments and segmentations given the training corpus, using Gibbs sampling.

Suppose we are provided a morphological disambiguator for the source language such as MADA morphology tokenization toolkit (Sadat and Habash, 2006) for Arabic.[4] The morphological disambiguator segments a source word to

[3] We also might have conditioned this decision on words *following* s_i, since those have all been generated already at this stage.

[4] MADA provides several segmentation schemes; among them the MADA-D3 scheme seeks to separate all morphemes of each word.

morphemes of smallest meaning-bearing units of the source language. Therefore, a target word is equivalent to one or several morphemes. Given a morphological disambiguation toolkit, we use its output to bias our inference by not considering word boundaries after every character but only considering potential word boundaries as a subset of the morpheme boundaries set. In this way, the inference uses the morphological disambiguation toolkit to limit its search space.

The inference starts with an initial segmentation of the source corpus and also its alignment to the target corpus. The Gibbs sampler considers one potential word boundary at a time. There are two hypotheses at any given boundary position of a sentence pair $(\mathbf{s}, \mathbf{t})$: the *merge* hypothesis stands for no word boundary and the resulting source sentence $\mathbf{s}_{\text{merge}}$ has a word s spanning over the sample point; the *split* hypothesis indicates the resulting source sentence $\mathbf{s}_{\text{split}}$ has a word boundary at the sample point separating two words $s_1 s_2$. Similar to Goldwater et al. (2006) for monolingual segmentation, the sampler randomly chooses the boundary according to the relative probabilities of the merge hypothesis and the split hypothesis.

The model consists of source and alignment model variables; given the training corpora size of a machine translation system, the number of variables is large. So if the Gibbs sampler samples both source variables and alignment variables, the inference requires many iterations until the sampler mixes. Xu et al. (2008) fixed this by repeatedly applying GIZA++ word alignment after each sampling iteration through the training corpora.

Our inference technique is not precisely Gibbs sampling. Rather than sampling the alignment or attempting to collapse it out (by summing over all possible alignments when calculating the relative probabilities of the merge and split hypotheses), we seek the *best* alignment for each hypothesis. In other words, for each hypothesis, we perform a local search for a high-probability alignment of the merged word or split words, given the rest of alignment for the sentence. Up to one word may be displaced and realigned. This “local-best” alignment is used to score the hypothesis, and after sampling merge or split, we keep that best alignment.

This inference technique is motivated by runtime demands, but we do not yet know of a theoretical justification for combining random steps with maximization over some variables. A more complete analysis is left to future work.

5 Decoding for Unseen Test Sentences

Section 4 described how to get the model’s posterior distribution and the segmentation and alignment of the training data under the model. We are left with the problem of *decoding* or finding the segmentation of test sentences where the translations are not available. This is needed when we want to translate new sentences. Here, tokenization is performed as a preprocessing step, decoupled from the subsequent translation steps.

The decoding step uses the model’s posterior distribution for the training data to segment unseen source sentences. Because of the clear separation of the source model and the alignment model, the source model distribution learned from the Gibbs sampling directly represents the distribution over the source language and can therefore also handle the segmentation of unknown words in new test sentences. Only the source model is used in preprocessing.

The best segmentation $\mathbf{s}^*$ of a string of characters $\mathbf{c} = \left(c_1, \ldots, c_{|\mathbf{c}|}\right)$ according to the n-gram source model is:

$$\mathbf{s}^* = \underset{\mathbf{s}\text{ from }\mathbf{c}}{\operatorname{argmax}}\, \mathsf{p}\left(|\mathbf{s}|\right) \prod_{i=1}^{i=|\mathbf{s}|} \mathsf{p}\left(s_i \,|\, s_{i-n}, \ldots, s_{i-1}\right)$$

We use a stochastic finite-state machine for decoding. This is possible by composition of the following two finite state machines:

- Acceptor $\mathcal{A}_{\mathbf{c}}$. The string of characters $\mathbf{c}$ is represented as an finite state acceptor machine where any path through the machine represents an unweighted segmentation of $\mathbf{c}$.

- Source model weighted finite state transducer $\mathcal{L}_{\mathbf{c}}$. Knight and Al-Onaizan (1998) show how to build an n-gram language model by a weighted finite state machine. The states of the transducer are $(n-1)$-gram history, the edges are words from the language. The arc s_i coming from state

$(s_{i-n}, \ldots, s_{i-1})$ to state $(s_{i-n+1}, \ldots, s_i)$ has weight $\mathsf{p}\,(s_i \,|\, s_{i-n}, \ldots, s_{i-1})$.

The best segmentation s^* is given as $\mathrm{s}^* = \mathrm{BestPath}(\mathcal{A}_\mathbf{c} \circ \mathcal{L}_\mathbf{c})$.

6 Experiments

This section presents experimental results on Arabic-English and Chinese-English translation tasks using the proposed segmentation technique.

6.1 Arabic-English

As a training set we use the BTEC corpus distributed by the International Workshop on Spoken Language Translation (IWSLT) (Matthias and Chiori, 2005). The corpus is a collection of conversation transcripts from the travel domain. The "Supplied Data" track consists of nearly 20K Arabic-English sentence pairs. The development set consists of 506 sentences from the IWSLT04 evaluation test set and the unseen set consists of 500 sentences from the IWSLT05 evaluation test set. Both development set and test set have 16 references per Arabic sentence.

6.2 Chinese-English

The training set for Chinese-English translation task is also distributed by the IWSLT evaluation campaign. It consists of 67K Chinese-English sentence pairs. The development set and the test set each have 489 Chinese sentences and each sentence has 7 English references.

6.3 Results

We will report the translation results where the preprocessing of the source text are our unigram, bigram, and trigram source models and source-to-null model.

The MCMC inference algorithm starts with an initial segmentation of the source text into full word forms. For Chinese, we use the original word segmentation as distributed by IWSLT. To get an initial alignment, we generate the IBM4 Viterbi alignments in both directions using the GIZA++ toolkit (Och and Ney, 2003) and combine them using the "grow-diag-final-and" heuristic. The output of combining GIZA++ alignment for a sentence pair is a sequence of s_i-t_j entries where i is an index of the source sentence and j is an index of the target sentence. As our model allows only one-to-one mappings between the words in the source and target sentences, we remove s_i-t_j from the sequence if either the source word s_i or target word t_j is already in a previous entry of the combined alignment sequence. The resulting alignment is our initial alignment for the inference.

We also apply the MADA morphology segmentation toolkit (Habash and Rambow, 2005) to preprocess the Arabic corpus. We use the D3 scheme (each Arabic word is segmented into morphemes in sequence [CONJ+ [PART+ [Al+ BASE +PRON]]]), mark the morpheme boundaries, and then combine the morphemes again to have words in their original full word form. During inference, we only sample over these morpheme boundaries as potential word boundaries. In this way, we limit the search space, allowing only segmentations consistent with MADA-D3.

The inference samples 150 iterations through the whole training set and uses the posterior probability distribution from the last iteration for decoding. The decoding process is then applied to the entire training set as well as to the development and test sets to generate a consistent tokenization across all three data sets. We used the OpenFST toolkit (Allauzen et al., 2007) for finite-state machine implementation and operations. The output of the decoding is the preprocessed data for translation. We use the open source Moses phrase-based MT system (Koehn et al., 2007) to test the impact of the preprocessing technique on translation quality.[5]

6.3.1 Arabic-English Translation Results

We consider the Arabic-English setting. We use two baselines: original full word form and MADA-D3 tokenization scheme for Arabic-English translation. Table 1 compares the translation results of our segmentation methods with these baselines. Our segmentation method shows improvement over the two baselines on both the development and test sets. According to Sadat and Habash (2006), the MADA-D3 scheme per-

[5] The Moses translation alignment is the output of GIZA++, not from our MCMC inference.

	Dev.	Test
Original	59.21	54.00
MADA-D3	58.28	54.92
Unigram	**59.44**	56.18
Bigram	58.88	56.18
Trigram	58.76	**56.82**

Table 1: Arabic-English translation results (BLEU).

forms best for their Arabic-English translation especially for small and moderate data sizes. In our experiments, we see an improvement when using the MADA-D3 preprocessing over using the original Arabic corpus on the unseen test set, but not on the development set.

The Gibbs sampler only samples on the morphology boundary points of MADA-D3, so the improvement resulting from our segmentation technique does not come from removing unknown words. It is due to a better matching between the source and target sentences by integrating segmentation and alignment. We therefore expect the same impact on a larger training data set in future experiments.

6.3.2 Chinese-English Translation Results

	Dev.	Test
Whole word	23.75	**29.02**
Character	23.39	27.74
Unigram	**24.90**	**28.97**
Trigram	23.98	28.20

Table 2: Chinese-English translation result in BLEU score metric.

We next consider the Chinese-English setting. The translation performance using our word segmentation technique is shown in Table 2. There are two baselines for Chinese-English translation: (a) the source text in the full word form distributed by the IWSLT evaluation and (b) no segmentation of the source text, which is equivalent to interpreting each Chinese character as a single word.

Taking development and test sets into account, the best Chinese-English translation system results from our unigram model. It is significantly better than other systems on the development set and performs almost equally well with the IWSLT segmentation on the test set. Note that the segmentation distributed by IWSLT is a manual segmentation for the translation task.

Chung and Gildea (2009) and Xu et al. (2008) also showed improvement over a simple monolingual segmentation for Chinese-English translation. Our character-based translation result is comparable to their monolingual segmentations. Both trigram and unigram translation results outperform the character-based translation.

We also observe that there are no additional gains for Chinese-English translation when using a higher n-gram model. Our Gibbs sampler has the advantage that the samples are guaranteed to converge eventually to the model's posterior distributions, but in each step the modification to the current hypothesis is small and local. In iterations 100–150, the average number of boundary changes for the unigram model is 14K boundaries versus only 1.5K boundary changes for the trigram model. With 150 iterations, the inference output of trigram model might not yet represent its posterior distribution. We leave a more detailed investigation of convergence behavior to future work.

Conclusion and Future Work

We presented an unsupervised segmentation method for machine translation and presented experiments for Arabic-English and Chinese-English translation tasks. The model can incorporate existing monolingual segmentation models and seeks to learn a segmenter appropriate for a particular translation task (target language and dataset).

Acknowledgements

We thank Kevin Gimpel for interesting discussions and technical advice. We also thank the anonymous reviewers for useful feedback. This work was supported by DARPA Gale project, NSF grants 0844507 and 0915187.

References

Allauzen, C., M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. 2007. OpenFst: A General and Efficient Weighted Finite-State Transducer Library. In *Proceedings of the CIAA 2007*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer. http://www.openfst.org.

Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Comput. Linguist.*, 19(2):263–311.

Chung, T. and D. Gildea. 2009. Unsupervised Tokenization for Machine Translation. In *Proceedings of EMNLP 2009*, pages 718–726, Singapore, August. Association for Computational Linguistics.

Cohen, S. B., D. M. Blei, and N. A. Smith. 2010. Variational Inference for Adaptor Grammars. In *Proceedings of NAACL-HLT*, pages 564–572, June.

Creutz, Mathias and Krista Lagus. 2007. Unsupervised Models for Morpheme Segmentation and Morphology Learning. *ACM Trans. Speech Lang. Process.*, 4(1):1–34.

DeNero, J., A. Bouchard-Côté, and D. Klein. 2008. Sampling Alignment Structure under a Bayesian Translation Model. In *Proceedings of EMNLP 2008*, pages 314–323, Honolulu, Hawaii, October. Association for Computational Linguistics.

Dyer, C. 2009. Using a Maximum Entropy model to build segmentation lattices for MT. In *Proceedings of HLT 2009*, pages 406–414, Boulder, Colorado, June.

Goldwater, S. and T. L. Griffiths. 2007. A Fully Bayesian Approach to Unsupervised Part-of-Speech Tagging. In *Proceedings of ACL.*

Goldwater, S. and D. McClosky. 2005. Improving Statistical Machine Translation Through Morphological Analysis. In *Proc. of EMNLP.*

Goldwater, S., T. L. Griffiths, and M. Johnson. 2006. Contextual Dependencies in Unsupervised Word Segmentation. In *Proc. of COLING-ACL.*

Habash, N. and O. Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging, and Morphological Disambiguation in One Fell Swoop. In *Proc. of ACL.*

Knight, K. and Y. Al-Onaizan. 1998. Translation with Finite-State Devices. In *Proceedings of AMTA*, pages 421–437.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. of ACL (demo session).*

Matthias, E. and H. Chiori. 2005. Overview of the IWSLT 2005 Evaluation Campaign. In *Proceedings of IWSLT.*

Mochihashi, D., T. Yamada, and N. Ueda. 2009. Bayesian Unsupervised Word Segmentation with Nested Pitman-Yor Language Modeling. In *Proceedings of 47th ACL*, pages 100–108, Suntec, Singapore, August.

Nießen, S. and H. Ney. 2004. Statistical Machine Translation with Scarce Resources Using Morpho-Syntactic Information. *Computational Linguistics*, 30(2), June.

Och, F. and H. Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1).

Sadat, F. and N. Habash. 2006. Combination of Arabic Preprocessing Schemes for Statistical Machine Translation. In *Proceedings of the ACL*, pages 1–8.

Snyder, B. and R. Barzilay. 2008. Unsupervised Multilingual Learning for Morphological Segmentation. In *Proceedings of ACL-08: HLT*, pages 737–745, June.

Snyder, B., T. Naseem, J. Eisenstein, and R. Barzilay. 2008. Unsupervised Multilingual Learning for POS Tagging. In *Proceedings of EMNLP.*

Snyder, B., T. Naseem, and R. Barzilay. 2009. Unsupervised Multilingual Grammar Induction. In *Proceedings of ACL-09*, pages 73–81, August.

Teh, Y. W. 2006. A Hierarchical Bayesian Language Model Based On Pitman-Yor Processes. In *Proceedings of ACL*, pages 985–992, July.

Vogel, S., H. Ney, and C. Tillmann. 1996. HMM-Based Word Alignment in Statistical Translation. In *Proceedings of COLING*, pages 836–841.

Xu, J., J. Gao, K. Toutanova, and H. Ney. 2008. Bayesian Semi-Supervised Chinese Word Segmentation for Statistical Machine Translation. In *Proceedings of (Coling 2008)*, pages 1017–1024, Manchester, UK, August.

Automatic Discovery of Feature Sets for Dependency Parsing

Peter Nilsson **Pierre Nugues**
Department of Computer Science
Lund University
peter.nilsson.lund@telia.com Pierre.Nugues@cs.lth.se

Abstract

This paper describes a search procedure to discover optimal feature sets for dependency parsers. The search applies to the shift–reduce algorithm and the feature sets are extracted from the parser configuration. The initial feature is limited to the first word in the input queue. Then, the procedure uses a set of rules founded on the assumption that topological neighbors of significant features in the dependency graph may also have a significant contribution. The search can be fully automated and the level of greediness adjusted with the number of features examined at each iteration of the discovery procedure.

Using our automated feature discovery on two corpora, the Swedish corpus in CoNLL-X and the English corpus in CoNLL 2008, and a single parser system, we could reach results comparable or better than the best scores reported in these evaluations. The CoNLL 2008 test set contains, in addition to a *Wall Street Journal* (WSJ) section, an out-of-domain sample from the Brown corpus. With sets of 15 features, we obtained a labeled attachment score of 84.21 for Swedish, 88.11 on the WSJ test set, and 81.33 on the Brown test set.

1 Introduction

The selection of relevant feature sets is crucial to the performance of dependency parsers and this process is still in large part manual. Moreover, feature sets are specific to the languages being analyzed and a set optimal for, say, English can yield poor results in Chinese. With dependency parsers being applied today to dozens of languages, this makes the parametrization of a parser both a tedious and time-consuming operation. Incidentally, the advent of machine-learning methods seems to have shifted the tuning steps in parsing from polishing up grammar rules to the optimization of feature sets. And as with the writing of a grammar, the selection of features is a challenging task that often requires a good deal of effort and inspiration.

Most automatic procedures to build feature sets resort to greedy algorithms. Forward selection constructs a set by adding incrementally features from a predetermined superset while backward elimination removes them from the superset (Attardi et al., 2007). Both methods are sometimes combined (Nivre et al., 2006b). The selection procedures evaluate the relevance of a candidate feature in a set by its impact on the overall parsing score: Does this candidate improve or decrease the performance of the set?

Greedy search, although it simplifies the design of feature sets, shows a major drawback as it starts from a closed superset of what are believed to be the relevant features. There is a broad consensus on a common feature set including the words close to the top of the stack or the beginning of the queue, for the shift–reduce algorithm, but no clear idea on the limits of this set.

In this paper, we describe an automatic discovery procedure that is not bounded by any prior knowledge of a set of potentially relevant features. It applies to the shift–reduce algorithm and the ini-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 824–832,
Beijing, August 2010

tial feature consists solely of the first word of the queue. The search explores nodes along axes of the parser's data structures and the partially built graph using proximity rules to uncover sequences of relevant, efficient features. Using this procedure on the Swedish corpus in CoNLL-X and the English corpus in CoNLL 2008, we built feature sets that enabled us to reach a labeled attachment score of 84.21 for Swedish, 88.11 on the *Wall Street Journal* section of CoNLL 2008, and 81.33 on the Brown part of it with a set cardinality of 15.

2 Transition-based Parsing

Transition-based methods (Covington, 2001; Nivre, 2003; Yamada and Matsumoto, 2003; Zhang and Clark, 2009) have become a popular approach in multilingual dependency parsing because of their speed and performance. Transition-based methods share common properties and build a dependency graph from a sequence of actions, where each action is determined using a feature function. In a data-driven context, the function is typically implemented as a classifier and the features are extracted from the partially built graph and the parser's data structures, most often a queue and a stack.

2.1 Parser Implementation

In this study, we built a parser using Nivre's algorithm (Nivre, 2003). The parser complexity is linear and parsing completes in at most $2n+1$ operations, where n is the length of the sentence. Table 1 shows the transitions and actions to construct a dependency graph.

Given a sentence to parse, we used a classifier-based guide to predict the transition sequence to apply. At each step, the guide extracts features from the parser configuration and uses them as input to a classifier to predict the next transition. Before training the classification models, we projectivized the corpus sentences (Kunze, 1967; Nivre and Nilsson, 2005). We did not attempt to recover nonprojective sentences after parsing.

2.2 Training and Parsing Procedure

We extracted the features using a gold-standard parsing of the training set. We organized the classification, and hence the feature extraction, as a two-step process. The first step determines the action among *LeftArc*, *RightArc*, *Reduce*, and *Shift*; the second one, the grammatical function, if the action is either a left arc or a right arc.

Action	Parser configuration
Init.	$\langle nil, W, \emptyset \rangle$
LeftArc	$\langle n \mid S, n' \mid Q, G \rangle \rightarrow \langle S, n' \mid Q, G \cup \{\langle n', n \rangle\} \rangle$
RightArc	$\langle n \mid S, n' \mid Q, G \rangle \rightarrow \langle n' \mid n \mid S, Q, G \cup \{\langle n, n' \rangle\} \rangle$
Reduce	$\langle n \mid S, Q, G \rangle \rightarrow \langle S, Q, G \rangle$
Shift	$\langle S, n \mid Q, G \rangle \rightarrow \langle n \mid S, Q, G \rangle$

Table 1: Parser transitions (Nivre, 2003). W is the input, G, the graph, S, the stack, and Q, the queue. The triple $\langle S, Q, G \rangle$ represents the parser configuration and n, n', and n'' are lexical tokens. $\langle n', n \rangle$ represents an arc from n' to n.

Once the features are extracted, we train the corresponding models that we apply to the test corpus to predict the actions and the arc labels.

3 Feature Discovery

We designed an automatic procedure to discover and select features that is guided by the structure of the graph being constructed. The search algorithm is based on the assumption that if a feature makes a significant contribution to the parsing performance, then one or more of its topological neighbors in the dependency graph may also be significant. The initial state, from which we derive the initial feature, consists of the first word in the queue. There is no other prior knowledge on the features.

3.1 Node Attributes

In the discovery procedure, we considered the nodes of four data structures: the queue, the stack, the sentence, and the graph being constructed. We extracted three attributes (or fields) from each node: two static ones, the lexical value of the node and its part of speech, and a dynamic one evaluated at parse time: the dependency label of the arc linking the node to its head, if it exists. We denoted the attributes of node w, respectively,

$LEX(w)$, $POS(w)$, and $DEP(w)$. These attributes are used as input by most dependency parsers, whatever the language being parsed.

3.2 Search Axes

The feature search covers three different axes: the parser's data structures – the queue and the stack –, the graph being constructed, and the sentence. Given a feature set at step n of the discovery procedure, we defined a successor function that generates the set of topological neighbors of all the members in the feature set along these three axes. For a particular feature:

The data structure axis consists of the nodes in the stack and the queue. The immediate neighbors of a node in the stack are the adjacent nodes above and below. In the queue, these are the adjacent nodes before and after it. The top node on the stack and the next node in the queue have a special connection, since they are the ones used by the parser when creating an arc. Therefore, we considered them as immediate neighbors to each other. For a node that is neither in the stack, nor in the queue, there is no connection along this axis.

The graph axes traverse the partially constructed graph horizontally and vertically. The horizontal axis corresponds to the sibling nodes connected by a common head (Figure 1). The immediate neighbors of a node are its nearest siblings to the left and to the right. The vertical axis corresponds to the head and child nodes. The immediate neighbors are the head node as well as the leftmost and rightmost child nodes. There is no connection for nodes not yet part of the graph.

The sentence axis traverses the nodes in the order they occur in the original sentence. The immediate neighbors of a node are the previous and next words in the sentence.

4 Representing Features and Their Neighbors

We represented features with a parameter format partly inspired by MaltParser (Nivre et al., 2006a).

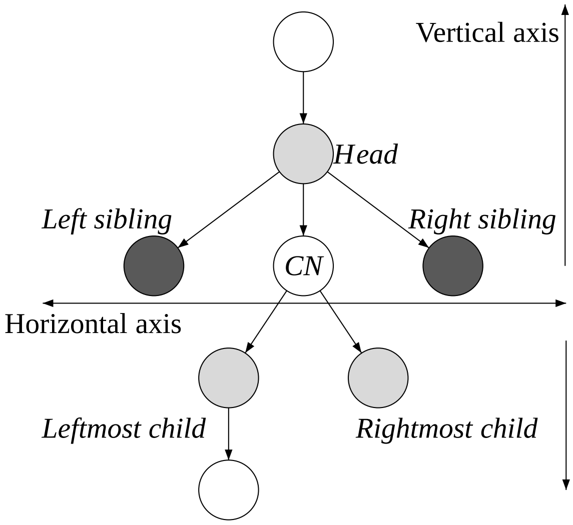


Figure 1: The vertical and horizontal axes, respectively in light and dark gray, relative to *CN*.

Each parameter consists of two parts. The first one represents a node in a data structure (*STACK* or *QUEUE*) and an attribute:

The nodes are identified using a zero-based index. Thus $STACK_1$ designates the second node on the stack.

The attribute of a node is one of part of speech (*POS*), lexical value (*LEX*), or dependency label (*DEP*), as for instance $LEX(QUEUE_0)$ that corresponds to the lexical value of the first token in the queue.

The second part of the parameter is an optional navigation path that allows to find other destination nodes in the graph. It consists of a sequence of instructions to move from the start node to the destination node. The list of possible instructions are:

- h: head of the current node;
- lc/rc: leftmost/rightmost child of the node;
- pw/fw: previous/following word of the node in the original sentence.

An example of a feature obtained using the navigation part is $POS(STACK_1\ lc\ pw)$, which is interpreted as: start from $STACK_1$. Then, using the instructions lc and pw, move to the left child of the start node and to the previous word of this child in the sentence. The requested feature is the part of speech of the destination node.

5 Initial State and Successor Function

The feature discovery is an iterative procedure that grows the feature set with one new feature at each iteration. We called *generation* such an iteration, where generation 1 consists of a single node. We denoted $FeatSet_i = \{f_1, f_2, ..., f_i\}$ the feature set obtained at generation i.

Although the features of a classifier can be viewed as a set, we also considered them as a tuple, where $Feat_i = \langle f_1, f_2, ..., f_i \rangle$ is the i-tuple at generation i and f_k, the individual feature discovered at generation k with $1 \leqslant k \leqslant i$. This enables us to keep the order in which the individual features are obtained during the search.

5.1 Initial State

We start the feature search with the empty set, $\emptyset$, that, by convention, has one neighbor: the first node in the queue $QUEUE_0$. We chose this node because this is the only one which is certain to exist all along the parsing process. Intuitively, this is also obvious that $QUEUE_0$ plays a significant role when deciding a parsing action. We defined the successor function of the empty set as: $SUCC(\emptyset) = \{POS(QUEUE_0), LEX(QUEUE_0)\}$.

5.2 Successors of a Node

The successors of a node consist of itself and all its topological neighbors along the three axes with their three possible attributes: part of speech, lexical value, and dependency label. For a particular feature in *FeatSet*, the generation of its successors is carried out through the following steps:

1. Interpret the feature with its possible navigation path and identify the destination node n.
2. Find all existing immediate neighboring nodes of n along the three search axes.
3. Assign the set of attributes – *POS*, *LEX*, and *DEP* – to n and its neighboring nodes.

If at any step the requested node does not exist, the feature evaluates to *NOTHING*.

5.3 Rules to Generate Neighbors

The generation of all the neighbors of the features in *FeatSet* may create duplicates as a same node can sometimes be reached from multiple paths. For instance, if we move to the leftmost child of a node and then to the head of this child, we return to the original node.

To compute the successor function, we built a set of rules shown in Table 2. It corresponds to a subset of the rules described in the axis search (Sect. 3.2) so that it omits the neighbors of a node that would unavoidably create redundancies. The third column in Table 2 shows the rules to generate the neighbors of $POS(QUEUE_0)$. They correspond to the rows:

PL. This stands for the *POS* and *LEX* attributes of the node. We only add $LEX(QUEUE_0)$ as we already have $POS(QUEUE_0)$.

PLD lc *and* PLD rc. *POS*, *LEX*, and *DEP* of the node's leftmost and rightmost children.

PLD pw. *POS*, *LEX*, and *DEP* of the previous word in the original string. The following word is the same as the next node in the queue, which is added in the next step. For that reason, *following word* is not added.

PL $QUEUE_1$. *POS* and *LEX* of $QUEUE_1$.

PLD $STACK_0$. *POS*, *LEX*, and *DEP* of $STACK_0$. This rule connects the queue to the top node of the stack.

Table 3 summarizes the results of the rule application and shows the complete list of successors of $POS(QUEUE_0)$. In this way, the search for a node's neighbors along the axes is reduced to one direction, either left or right, or up or down, that will depend on the topological relation that introduced the node in the feature set.

6 Feature Selection Algorithm

At each generation, we compute the Cartesian product of the current feature tuple $Feat_i$ and the set defined by its neighbors. We define the set of candidate tuples $CandFeat_{i+1}$ at generation $i+1$ as:

$$CandFeat_{i+1} = \{Feat_i\} \times SUCC(Feat_i),$$

where we have $Card(CandFeat_{i+1}) = Card(SUCC(Feat_i))$.

The members of $CandFeat_{i+1}$ are ranked according to their parsing score on the development

Data structures				Navigation paths					
$STACK_0$	$STACK_n, n > 0$	$QUEUE_0$	$QUEUE_n, n > 0$	*h*	*lc, rc*	*ls*	*rs*	*pw*	*fw*
PLD	PLD	PL	PL	h				h	h
PLD h	PLD h				lc	lc	lc	lc	lc
PLD lc	PLD lc	PLD lc			rc	rc	rc	rc	rc
PLD rc	PLD rc	PLD rc		ls	ls	ls		ls	ls
PLD ls	PLD ls			rs	rs		rs	rs	rs
PLD rs	PLD rs			pw	pw	pw	pw	pw	
PLD pw	PLD pw	PLD pw		fw	fw	fw	fw		fw
PLD fw	PLD fw								
PLD $STACK_1$	PLD $STACK_{n+1}$	PL $QUEUE_1$	PL $QUEUE_{n+1}$						
PL $QUEUE_0$		PLD $STACK_0$							

Table 2: Rules to compute the successors of a node. For each node category given in row 2, the procedure adds the features in the column headed by the category. PLD stands for the *POS*, *LEX*, and *DEP* attributes. In the right-hand side of the table, the category corresponds to the last instruction of the navigation path, if it exists, for instance *pw* in the feature $POS(STACK_1\ lc\ pw)$. We read the six successors of this node in the fifth column headed by *pw*: $STACK_1\ lc\ pw\ h$, $STACK_1\ lc\ pw\ lc$, $STACK_1\ lc\ pw\ rc$, $STACK_1\ lc\ pw\ ls$, $STACK_1\ lc\ pw\ rs$, and $STACK_1\ lc\ pw\ pw$. We then apply all the attributes to these destination nodes to generate the features.

Initial feature	POS	QUEUE	0	
Successors	LEX	QUEUE	0	
	PLD	QUEUE	0	lc
	PLD	QUEUE	0	rc
	PLD	QUEUE	0	pw
	PL	QUEUE	1	
	PLD	STACK	0	

Table 3: Features generated by the successor function $SUCC(\{POS(QUEUE_0)\})$. PLD stands for the three attributes *POS*, *LEX*, and *DEP* of the node; PL for *POS* and *LEX*.

set and when applying a greedy best-first search, $Feat_{i+1}$ is assigned with the tuple yielding the highest score:

$$Feat_{i+1} \leftarrow eval\_best(CandFeat_{i+1}).$$

The procedure is repeated with the immediate neighbors of $Feat_{i+1}$ until the improvement of the score is below a certain threshold.

We extended this greedy version of the discovery with a beam search that retains the *N*-best successors from the candidate set. In our experiments, we used beam widths of 4 and 8.

7 Experimental Setup

In a first experiment, we used the Swedish corpus of the CoNLL-X shared task (Buchholz and Marsi, 2006). In a second experiment, we applied the feature discovery procedure to the English corpus from CoNLL 2008 (Surdeanu et al., 2008), a dependency corpus converted from the Penn Treebank and the Brown corpus. In both experiments, we used the LIBSVM package (Chang and Lin, 2001) with a quadratic kernel, $\gamma = 0.2$, $C = 0.4$, and $\varepsilon = 0.1$. These parameters are identical to Nivre et al. (2006b) to enable a comparison of the scores.

We evaluated the feature candidates on a development set using the labeled and unlabeled attachment scores (LAS and UAS) that we computed with the `eval.pl` script from CoNLL-X. As there was no development set for the Swedish corpus, we created one by picking out every 10th sentence from the training set. The training was then carried out on the remaining part of the set.

8 Feature Discovery on a Swedish Corpus

In a first run, the search was optimized for the UAS. In a second one, we optimized the LAS. We also report the results we obtained subsequently on the CoNLL-X test set as an indicator of how well the training generalized.

8.1 The First and Second Generations

Table 4 shows the feature performance at the first generation sorted by UAS. The first row shows the two initial feature candidates, $\langle POS(QUEUE_0)\rangle$ and $\langle LEX(QUEUE_0)\rangle$. The third row shows the score produced by the initial features alone. The next rows show the unlabeled and labeled attachment scores with feature pairs combining one of the initial features and the one listed in the row. The combination of $POS(QUEUE_0)$ and $POS(STACK_0)$ yielded the best UAS: 74.02. The second feature improves the performance of $POS(QUEUE_0)$ by more than 30 points from 43.49.

For each generation, we applied a beam search. We kept the eight best pairs as starting states for the second generation and we added their neighboring nodes. Table 5 shows the eight best results out of 38 for the pair $\langle POS(QUEUE_0), POS(STACK_0)\rangle$.

Parent state: $\langle POS(QUEUE_0), POS(STACK_0)\rangle$				
Dev set		Test set		
UAS	LAS	UAS	LAS	Successors
79.50	65.34	79.07	65.86	P QUEUE 1
78.73	66.98	76.04	64.51	L STACK 0 fw
77.42	63.08	74.63	61.86	L QUEUE 1
77.06	64.54	75.28	62.90	L QUEUE 0 pw
76.83	66.01	73.61	63.77	L QUEUE 0
76.63	63.62	74.75	63.17	P STACK 0 fw
76.44	64.24	74.09	62.02	L STACK 0
76.39	63.12	73.99	61.16	L QUEUE 0 lc

Table 5: Ranking the successors of $\langle POS(QUEUE_0), POS(STACK_0)\rangle$ on the Swedish corpus. Out of the 38 successors, we show the eight that yielded the best results. P stands for *POS*, L for *LEX*, and D for *DEP*.

8.2 Optimizing the Unlabeled Attachement Score

We iterated the process over a total of 16 generations. Table 6, left-hand side, shows the list of the best scores for each generation. The scores on the development set increased steadily until generation 13, then reached a plateau, and declined around generation 15. The test set closely followed the development set with values about 1% lower. On this set, we reached a peak performance at generation 12, after which the results decreased.

Table 6, right-hand side, shows the features producing the final score in their order of inclusion in the feature set. As we applied a beam search, a feature listed at generation *i* does not necessary correspond to the highest score for this generation, but belongs to the feature tuple producing the best result at generation 16.

8.3 Optimizing the Labeled Attachement Score

We also applied the feature discovery with a search optimized for the labeled attachment score. This time, we reduced the beam width used in the search from 8 to 4 as we noticed that the candidates between ranks 5 and 8 never contributed to the best scoring feature set for any generation.

We observed a score curve similar to that of the UAS-optimized search. The train set followed the development set with increasing values for each generation but 1-2% lower. The optimal value was obtained at generation 15 with 84.21% for the test set. Then, the score for the test set decreased.

9 Feature Discovery on a Corpus of English

The training and development sets of the CoNLL 2008 corpus contain text from the *Wall Street Journal* exclusively. The test set contains text from the Brown corpus as well as from the *Wall Street Journal*. Table 7 shows the results after 16 generations. We used a beam width of 4 and the tests were optimized for the unlabeled attachment score. As for Swedish, we reached the best scores around generation 14-15. The results on the in-domain test set peaked at 90.89 and exceeded the results on the development set. As expected, the results for the out-of-domain corpus were lower, 87.50, however the drop was limited to 3.4.

10 Discussion and Conclusion

The results we attained with feature set sizes as small as 15 are competitive or better than figures

Parent state		$\langle POS(QUEUE_0)\rangle$	$\langle LEX(QUEUE_0)\rangle$		
UAS	LAS	Successors	UAS	LAS	Successors
43.49	26.45	None	42.76	23.56	None
74.02	59.67	POS STACK 0	**65.86**	52.18	POS STACK 0
67.77	54.50	LEX STACK 0	**58.59**	45.51	LEX STACK 0
58.37	41.83	POS QUEUE 0 pw	**51.98**	37.70	POS QUEUE 0 pw
55.28	38.49	LEX QUEUE 0 pw	50.44	29.71	POS QUEUE 1
51.53	30.43	POS QUEUE 1	50.38	35.24	LEX QUEUE 0 pw
51.05	32.66	LEX QUEUE 0 lc	49.37	32.27	POS QUEUE 0
49.71	31.54	POS QUEUE 0 lc	48.91	27.77	LEX QUEUE 1
49.49	29.18	LEX QUEUE 1	48.66	29.91	LEX QUEUE 0 lc
49.37	32.27	LEX QUEUE 0	47.25	28.92	LEX QUEUE 0 rc
48.68	29.34	DEP STACK 0	47.09	28.65	POS QUEUE 0 lc
48.47	30.84	LEX QUEUE 0 rc	46.68	27.08	DEP QUEUE 0 lc
46.77	26.86	DEP QUEUE 0 lc	45.69	27.83	POS QUEUE 0 rc
46.40	29.95	POS QUEUE 0 rc	44.77	26.17	DEP STACK 0
42.27	25.21	DEP QUEUE 0 pw	44.43	26.47	DEP QUEUE 0 rc
41.04	26.56	DEP QUEUE 0 rc	41.87	23.04	DEP QUEUE 0 pw

Table 4: Results of the beam search on the Swedish corpus at the first generation with the two initial feature candidates, $\langle POS(QUEUE_0)\rangle$ and $\langle LEX(QUEUE_0)\rangle$, respectively on the left- and right-hand side of the table. The third row shows the score produced by the initial features alone and the next rows, the figures for the candidate pairs combining the initial feature and the successor listed in the row. The eight best combinations shown in bold are selected for the next generation.

Generation	Dev set		Test set		Features
	UAS	LAS	UAS	LAS	
1	43.49	26.45	45.93	30.19	POS QUEUE 0
2	74.02	59.67	71.60	58.37	POS STACK 0
3	79.50	65.34	79.07	65.86	POS QUEUE 1
4	83.58	71.76	82.75	70.98	LEX STACK 0 fw
5	85.96	76.03	84.82	74.75	LEX STACK 0
6	87.23	77.32	86.34	76.52	LEX QUEUE 0 lc
7	88.42	80.00	87.67	78.99	POS STACK 1
8	89.43	81.56	88.09	80.26	LEX QUEUE 1
9	89.84	83.20	88.69	82.33	LEX QUEUE 0
10	90.23	83.89	89.17	83.31	DEP STACK 0 lc
11	90.49	84.31	89.58	**83.85**	POS STACK 0 fw
12	90.73	84.47	**89.66**	83.83	LEX STACK 0 fw ls
13	90.81	84.60	89.52	83.75	LEX STACK 0 fw ls lc
14	90.81	**84.70**	89.32	83.73	POS STACK 1 h
15	**90.85**	84.67	89.13	83.21	LEX STACK 1 rs
16	90.84	84.68	88.65	82.75	POS STACK 0 fw ls rc

Table 6: Best results for each generation on the Swedish corpus, optimized for UAS. Figures in bold designate the best scores. The right-hand side of the table shows the feature sequence producing the best result at generation 16.

Generation	Dev set		Test set WSJ		Test set Brown		Features
	UAS	LAS	UAS	LAS	UAS	LAS	
1	45.25	33.77	45.82	34.49	52.12	40.70	POS QUEUE 0
2	64.42	55.64	64.71	56.44	71.29	62.41	LEX STACK 0
3	78.62	68.77	78.99	70.30	78.67	65.17	POS QUEUE 1
4	81.83	76.67	82.46	77.82	80.57	72.95	LEX STACK 0 fw
5	84.43	79.78	84.89	80.88	84.03	76.99	POS STACK 0
6	85.95	81.60	86.61	82.93	84.55	77.80	DEP QUEUE 0 lc
7	86.95	82.73	87.73	84.09	85.26	78.48	LEX STACK 1
8	88.03	83.62	88.52	84.74	85.66	78.73	LEX QUEUE 1
9	88.61	84.97	89.15	86.20	86.29	79.86	LEX QUEUE 0
10	89.09	85.43	89.47	86.60	86.43	80.02	POS QUEUE 2
11	89.54	85.87	90.25	87.40	87.00	80.75	POS STACK 0 pw
12	89.95	86.21	90.63	87.77	86.87	80.46	POS QUEUE 3
13	90.26	86.56	90.64	87.80	87.35	80.86	POS STACK 1 pw
14	90.54	86.81	90.71	87.88	**87.50**	81.30	POS QUEUE 0 pw
15	90.61	86.94	**90.89**	**88.11**	87.47	**81.33**	LEX STACK 0 lc
16	**90.65**	**87.00**	90.88	88.09	87.42	81.28	POS STACK 0 pw ls

Table 7: Best results for each generation. English corpus. Selection optimized for UAS.

reported by state-of-the-art transition-based systems. We reached a UAS of 89.66 on the CoNLL-X Swedish corpus. On the same corpus, the top scores reported in the shared task were slightly lower: 89.54 and 89.50. Our best LAS was 84.21, and the two best scores in CoNLL-X were 84.58 and 82.55. Our results for the English corpus from CoNLL 2008 were optimized for an unlabeled attachment score and we obtained 90.89 for the in-domain test set and 87.50 for the out-of-domain one. Our best LAS were 88.11 and 81.33. Official results in CoNLL 2008 only reported the labeled attachment scores, respectively 90.13 and 82.81[1].

We believe these results remarkable. We used a single-parser system as opposed to ensemble systems and the results on the Brown corpus show an excellent resilience and robustness on out-of-domain data. The automatic discovery produced results matching or exceeding comparable systems, although no prior knowledge of the language being analyzed was used and no feature set was provided to the parser.

Although, a systematic search requires no intuitive guessing, it still consumes a considerable machine time. Due to the learning algorithm we use, SVM, training a model takes between 1 and 130 hours depending on the size of the corpus. The number of models to train at each generation corresponds to the number of feature candidates times the beam width. The first generation contains about 15 feature candidates per feature set and since features are only added, the number of candidates can grow to 100 at generation 10.

We believe there is a margin for improvement both in the parsing scores and in the time needed to determine the feature sets. Our scores in Swedish were obtained with models trained on 90% of the training set. They could probably be slightly improved if they had been trained on a complete set. In our experiments, we used three attributes: the part of speech, lexical value, and dependency label of the node. These attributes could be extended to lemmas and grammatical features. SVMs yield a high performance, but they are slow to train. Logistic regression with, for instance, the LIBLINEAR package (Fan et al., 2008) would certainly reduce the exploration time.

[1] Results are not exactly comparable as we used the CoNLL-X evaluation script that gives slightly higher figures.

Acknowledgments

The research leading to these results has received funding from the European community's seventh framework program FP7/2007-2013, challenge 2, cognitive systems, interaction, robotics, under grant agreement No 230902—ROSETTA.

References

Attardi, Giuseppe, Felice Dell'Orletta, Maria Simi, Atanas Chanev, and Massimiliano Ciaramita. 2007. Multilingual dependency parsing and domain adaptation using DeSR. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1112–1118, Prague, Czech Republic, June. Association for Computational Linguistics.

Buchholz, Sabine and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City, June. Association for Computational Linguistics.

Chang, Chih-Chung and Chih-Jen Lin. 2001. LIBSVM: a library for support vector machines. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Covington, Michael A. 2001. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, Athens, Georgia.

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Kunze, Jürgen. 1967. Die Behandlung nicht-projektiver Strukturen bei der syntaktischen Analyse und Synthese des englischen und des deutschen. In *MASPEREVOD-67: Internationales Symposium der Mitgliedsländer des RGW*, pages 2–15, Budapest, 10.–13. Oktober.

Nivre, Joakim and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, June.

Nivre, Joakim, Johan Hall, and Jens Nilsson. 2006a. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC2006)*, pages 2216–2219, Genoa, May 24-26.

Nivre, Joakim, Johan Hall, Jens Nilsson, Gülsen Eryigit, and Svetoslav Marinov. 2006b. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL)*, pages 221–225, New York, June, 8-9.

Nivre, Joakim. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 149–160, Nancy, 23-25 April.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the 12th Conference on Computational Natural Language Learning*, pages 159–177, Manchester, August.

Yamada, Hiroyasu and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 195–206, Nancy, 23-25 April.

Zhang, Yue and Stephen Clark. 2009. Transition-based parsing of the Chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT 09)*, pages 162–171, Paris, October.

Evaluation of Dependency Parsers on Unbounded Dependencies

Joakim Nivre
Uppsala University
joakim.nivre@lingfil.uu.se

Laura Rimell
Univ. of Cambridge
laura.rimell@cl.cam.ac.uk

Ryan McDonald
Google Inc.
ryanmcd@google.com

Carlos Gómez-Rodríguez
Universidade da Coruña
cgomezr@udc.es

Abstract

We evaluate two dependency parsers, MSTParser and MaltParser, with respect to their capacity to recover unbounded dependencies in English, a type of evaluation that has been applied to grammar-based parsers and statistical phrase structure parsers but not to dependency parsers. The evaluation shows that when combined with simple post-processing heuristics, the parsers correctly recall unbounded dependencies roughly 50% of the time, which is only slightly worse than two grammar-based parsers specifically designed to cope with such dependencies.

1 Introduction

Though syntactic parsers for English are reported to have accuracies over 90% on the Wall Street Journal (WSJ) section of the Penn Treebank (PTB) (McDonald et al., 2005; Sagae and Lavie, 2006; Huang, 2008; Carreras et al., 2008), broad-coverage parsing is still far from being a solved problem. In particular, metrics like attachment score for dependency parsers (Buchholz and Marsi, 2006) and Parseval for constituency parsers (Black et al., 1991) suffer from being an average over a highly skewed distribution of different grammatical constructions. As a result, infrequent yet semantically important construction types could be parsed with accuracies far below what one might expect.

This shortcoming of aggregate parsing metrics was highlighted in a recent study by Rimell et al. (2009), introducing a new parser evaluation corpus containing around 700 sentences annotated with unbounded dependencies in seven different grammatical constructions. This corpus was used to evaluate five state-of-the-art parsers for English, focusing on grammar-based and statistical phrase structure parsers. For example, in the sentence *By Monday, they hope to have a sheaf of* **documents** *both sides can* **trust**., parsers should recognize that there is a dependency between **trust** and **documents**, an instance of object extraction out of a (reduced) relative clause. In the evaluation, the recall of state-of-the-art parsers on this kind of dependency varies from a high of 65% to a low of 1%. When averaging over the seven constructions in the corpus, none of the parsers had an accuracy higher than 61%.

In this paper, we extend the evaluation of Rimell et al. (2009) to two dependency parsers, MSTParser (McDonald, 2006) and MaltParser (Nivre et al., 2006a), trained on data from the PTB, converted to Stanford typed dependencies (de Marneffe et al., 2006), and combined with a simple post-processor to extract unbounded dependencies from the basic dependency tree. Extending the evaluation to dependency parsers is of interest because it sheds light on whether highly tuned grammars or computationally expensive parsing formalisms are necessary for extracting complex linguistic phenomena in practice. Unlike the best performing grammar-based parsers studied in Rimell et al. (2009), neither MSTParser nor MaltParser was developed specifically as a parser for English, and neither has any special mechanism for dealing with unbounded dependencies. Dependency parsers are also often asymptotically faster than grammar-based or constituent parsers, e.g., MaltParser parses sentences in linear time.

Our evaluation ultimately shows that the recall of MSTParser and MaltParser on unbounded dependencies is much lower than the average (un)labeled attachment score for each system. Nevertheless, the two dependency parsers are found to perform only slightly worse than the best grammar-based parsers evaluated in Rimell et al.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 833–841,
Beijing, August 2010

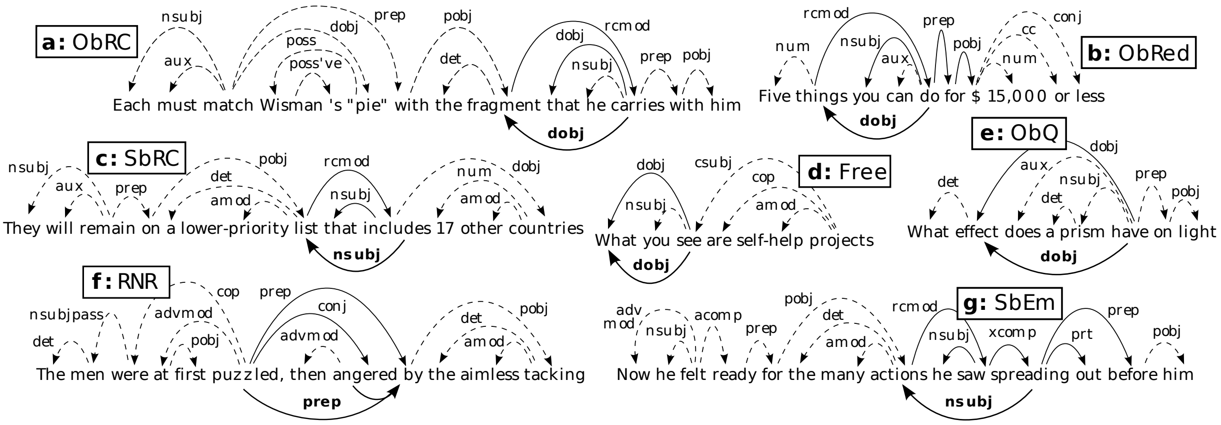

a: ObRC
nsubj
aux
poss
poss've
dobj
prep
pobj
det
dobj
rcmod
nsubj
prep
pobj
Each must match Wisman 's "pie" with the fragment that he carries with him
dobj
b: ObRed
rcmod
num
nsubj
aux
prep
pobj
cc
num
conj
Five things you can do for $ 15,000 or less
dobj
c: SbRC
nsubj
aux
prep
pobj
det
amod
rcmod
nsubj
num
amod
dobj
They will remain on a lower-priority list that includes 17 other countries
nsubj
d: Free
dobj
nsubj
csubj
cop
amod
What you see are self-help projects
dobj
e: ObQ
det
aux
nsubj
det
dobj
prep
pobj
What effect does a prism have on light
dobj
f: RNR
nsubjpass
det
cop
advmod
pobj
prep
conj
advmod
det
amod
pobj
The men were at first puzzled, then angered by the aimless tacking
prep
g: SbEm
advmod
nsubj
acomp
prep
pobj
det
amod
rcmod
nsubj
xcomp
prt
prep
pobj
Now he felt ready for the many actions he saw spreading out before him
nsubj

Figure 1: Examples of seven unbounded dependency constructions (**a**–**g**). Arcs drawn *below* each sentence represent the dependencies scored in the evaluation, while the tree *above* each sentence is the Stanford basic dependency representation, with solid arcs indicating crucial dependencies (cf. Section 4). All examples are from the development sets.

(2009) and considerably better than the other statistical parsers in that evaluation. Interestingly, though the two systems have similar accuracies overall, there is a clear distinction between the kinds of errors each system makes, which we argue is consistent with observations by McDonald and Nivre (2007).

2 Unbounded Dependency Evaluation

An unbounded dependency involves a word or phrase interpreted at a distance from its surface position, where an unlimited number of clause boundaries may in principle intervene. The unbounded dependency corpus of Rimell et al. (2009) includes seven grammatical constructions: object extraction from a relative clause (ObRC), object extraction from a reduced relative clause (ObRed), subject extraction from a relative clause (SbRC), free relatives (Free), object questions (ObQ), right node raising (RNR), and subject extraction from an embedded clause (SbEm), all chosen for being relatively frequent and easy to identify in PTB trees. Examples of the constructions can be seen in Figure 1. The evaluation set contains 80 sentences per construction (which may translate into more than 80 dependencies, since sentences containing coordinations may have more than one gold-standard dependency), while the development set contains between 13 and 37 sentences per construction. The data for ObQ sentences was obtained from various years of TREC, and for the rest of the constructions from the WSJ (0-1 and 22-24) and Brown sections of the PTB.

Each sentence is annotated with one or more gold-standard dependency relations representing the relevant unbounded dependency. The gold-standard dependencies are shown as arcs below the sentences in Figure 1. The format of the dependencies in the corpus is loosely based on the Stanford typed dependency scheme, although the evaluation procedure permits alternative representations and does not require that the parser output match the gold-standard exactly, as long as the "spirit" of the construction is correct.

The ability to recover unbounded dependencies is important because they frequently form part of the basic predicate-argument structure of a sentence. Subject and object dependencies in particular are crucial for a number of tasks, including information extraction and question answering. Moreover, Rimell et al. (2009) show that, although individual types of unbounded dependencies may be rare, the unbounded dependency types in the corpus, considered as a class, occur in as many as 10% of sentences in the PTB.

In Rimell et al. (2009), five state-of-the-art parsers were evaluated for their recall on the gold-standard dependencies. Three of the parsers were based on grammars automatically extracted from the PTB: the C&C CCG parser (Clark and Curran, 2007), the Enju HPSG parser (Miyao and Tsujii, 2005), and the Stanford parser (Klein and Manning, 2003). The two remaining systems were the

RASP parser (Briscoe et al., 2006), using a manually constructed grammar and a statistical parse selection component, and the DCU post-processor of PTB parsers (Cahill et al., 2004) using the output of the Charniak and Johnson reranking parser (Charniak and Johnson, 2005). Because of the wide variation in parser output representations, a mostly manual evaluation was performed to ensure that each parser got credit for the constructions it recovered correctly. The parsers were run essentially "out of the box", meaning that the development set was used to confirm input and output formats, but no real tuning was performed. In addition, since a separate question model is available for C&C, this was also evaluated on ObQ sentences. The best overall performers were C&C and Enju, which is unsurprising since they are deep parsers based on grammar formalisms designed to recover just such dependencies. The DCU post-processor performed somewhat worse than expected, often identifying the existence of an unbounded dependency but failing to identify the grammatical class (subject, object, etc.). RASP and Stanford, although not designed to recover such dependencies, nevertheless recovered a subset of them. Performance of the parsers also varied widely across the different constructions.

3 Dependency Parsers

In this paper we repeat the study of Rimell et al. (2009) for two dependency parsers, with the goal of evaluating how parsers based on dependency grammars perform on unbounded dependencies.

MSTParser[1] is a freely available implementation of the parsing models described in McDonald (2006). According to the categorization of parsers in Kübler et al. (2008) it is a *graph-based* parsing system in that core parsing algorithms can be equated to finding directed maximum spanning trees (either projective or non-projective) from a dense graph representation of the sentence. Graph-based parsers typically rely on global training and inference algorithms, where the goal is to learn models in which the weight/probability of correct trees is higher than that of incorrect trees. At inference time a global search is run to find the highest weighted dependency tree. Unfortunately, global inference and learning for graph-based dependency parsing is typically NP-hard (McDonald and Satta, 2007). As a result, graph-based parsers (including MSTParser) often limit the scope of their features to a small number of adjacent arcs (usually two) and/or resort to approximate inference (McDonald and Pereira, 2006).

MaltParser[2] is a freely available implementation of the parsing models described in Nivre et al. (2006a) and Nivre et al. (2006b). MaltParser is categorized as a *transition-based* parsing system, characterized by parsing algorithms that produce dependency trees by transitioning through abstract state machines (Kübler et al., 2008). Transition-based parsers learn models that predict the next state given the current state of the system as well as features over the history of parsing decisions and the input sentence. At inference time, the parser starts in an initial state, then greedily moves to subsequent states – based on the predictions of the model – until a termination state is reached. Transition-based parsing is highly efficient, with run-times often linear in sentence length. Furthermore, transition-based parsers can easily incorporate arbitrary non-local features, since the current parse structure is fixed by the state. However, the greedy nature of these systems can lead to error propagation if early predictions place the parser in incorrect states.

McDonald and Nivre (2007) compared the accuracy of MSTParser and MaltParser along a number of structural and linguistic dimensions. They observed that, though the two parsers exhibit indistinguishable accuracies overall, MSTParser tends to outperform MaltParser on longer dependencies as well as those dependencies closer to the root of the tree (e.g., verb, conjunction and preposition dependencies), whereas MaltParser performs better on short dependencies and those further from the root (e.g., pronouns and noun dependencies). Since long dependencies and those near to the root are typically the last constructed in transition-based parsing systems, it was concluded that MaltParser does suffer from some form of error propagation. On the other hand, the

[1] http://mstparser.sourceforge.net

[2] http://www.maltparser.org

richer feature representations of MaltParser led to improved performance in cases where error propagation has not occurred. However, that study did not investigate unbounded dependencies.

4 Methodology

In this section, we describe the methodological setup for the evaluation, including parser training, post-processing, and evaluation.[3]

4.1 Parser Training

One important difference between MSTParser and MaltParser, on the one hand, and the best performing parsers evaluated in Rimell et al. (2009), on the other, is that the former were never developed specifically as parsers for English. Instead, they are best understood as data-driven parser generators, that is, tools for generating a parser given a training set of sentences annotated with dependency structures. Over the years, both systems have been applied to a wide range of languages (see, e.g., McDonald et al. (2006), McDonald (2006), Nivre et al. (2006b), Hall et al. (2007), Nivre et al. (2007)), but they come with no language-specific enhancements and are not equipped specifically to deal with unbounded dependencies.

Since the dependency representation used in the evaluation corpus is based on the Stanford typed dependency scheme (de Marneffe et al., 2006), we opted for using the WSJ section of the PTB, converted to Stanford dependencies, as our primary source of training data. Thus, both parsers were trained on section 2–21 of the WSJ data, which we converted to Stanford dependencies using the Stanford parser (Klein and Manning, 2003). The Stanford scheme comes in several varieties, but because both parsers require the dependency structure for each sentence to be a tree, we had to use the so-called *basic* variety (de Marneffe et al., 2006).

It is well known that questions are very rare in the WSJ data, and Rimell et al. (2009) found that parsers trained only on WSJ data generally performed badly on the questions included in the evaluation corpus, while the C&C parser equipped with a model trained on a combination of WSJ and question data had much better performance. To investigate whether the performance of MSTParser and MaltParser on questions could also be improved by adding more questions to the training data, we trained one variant of each parser using data that was extended with 3924 questions taken from QuestionBank (QB) (Judge et al., 2006).[4] Since the QB sentences are annotated in PTB style, it was possible to use the same conversion procedure as for the WSJ data. However, it is clear that the conversion did not always produce adequate dependency structures for the questions, an observation that we will return to in the error analysis below.

In comparison to the five parsers evaluated in Rimell et al. (2009), it is worth noting that MSTParser and MaltParser were trained on the same basic data as four of the five, but with a different kind of syntactic representation – dependency trees instead of phrase structure trees or theory-specific representations from CCG and HPSG. It is especially interesting to compare MSTParser and MaltParser to the Stanford parser, which essentially produces the same kind of dependency structures as output but uses the original phrase structure trees from the PTB as input to training.

For our experiments we used MSTParser with the same parsing algorithms and features as reported in McDonald et al. (2006). However, unlike that work we used an atomic maximum entropy model as the second stage arc predictor as opposed to the more time consuming sequence labeler. McDonald et al. (2006) showed that there is negligible accuracy loss when using atomic rather than structured labeling. For MaltParser we used the projective Stack algorithm (Nivre, 2009) with default settings and a slightly enriched feature model. All parsing was projective because the Stanford dependency trees are strictly projective.

[3] To ensure replicability, we provide all experimental settings, post-processing scripts and additional information about the evaluation at http://stp.ling.uu.se/∼nivre/exp/.

[4] QB contains 4000 questions, but we removed all questions that also occurred in the test or development set of Rimell et al. (2009), who sampled their questions from the same TREC QA test sets.

4.2 Post-Processing

All the development and test sets in the corpus of Rimell et al. (2009) were parsed using MSTParser and MaltParser after part-of-speech tagging the input using SVMTool (Giménez and Màrquez, 2004) trained on section 2–21 of the WSJ data in Stanford basic dependency format. The Stanford parser has an internal module that converts the *basic* dependency representation to the *collapsed* representation, which explicitly represents additional dependencies, including unbounded dependencies, that can be inferred from the basic representation (de Marneffe et al., 2006). We performed a similar conversion using our own tool.

Broadly speaking, there are three ways in which unbounded dependencies can be inferred from the Stanford basic dependency trees, which we will refer to as *simple*, *complex*, and *indirect*. In the simple case, the dependency coincides with a single, direct dependency relation in the tree. This is the case, for example, in Figure 1**d**–**e**, where all that is required is that the parser identifies the dependency relation from a governor to an argument (`dobj(see, What)`, `dobj(have, effect)`), which we call the Arg relation; no post-processing is needed.

In the complex case, the dependency is represented by a *path* of direct dependencies in the tree, as exemplified in Figure 1**a**. In this case, it is not enough that the parser correctly identifies the Arg relation `dobj(carries, that)`; it must also find the dependency `rcmod(fragment, carries)`. We call this the Link relation, because it links the argument role inside the relative clause to an element outside the clause. Other examples of the complex case are found in Figure 1**c** and in Figure 1**f**.

In the indirect case, finally, the dependency cannot be defined by a path of labeled dependencies, whether simple or complex, but must be inferred from a larger context of the tree using heuristics. Consider Figure 1**b**, where there is a Link relation (`rcmod(things, do)`), but no corresponding Arg relation inside the relative clause (because there is no overt relative pronoun). However, given the other dependencies, we can infer with high probability that the implicit relation is `dobj`. Another example of the indirect case is in Figure 1**g**. Our post-processing tool performs more heuristic inference for the indirect case than the Stanford parser does (cf. Section 4.3).

In order to handle the complex and indirect cases, our post-processor is triggered by the occurrence of a Link relation (`rcmod` or `conj`) and first tries to add dependencies that are directly implied by a single Arg relation (relations involving relative pronouns for `rcmod`, shared heads and dependents for `conj`). If there is no overt relative pronoun, or the function of the relative pronoun is underspecified, the post-processor relies on the obliqueness hierarchy `subj` < `dobj` < `pobj` and simply picks the first "missing function", unless it finds a clausal complement (indicated by the labels `ccomp` and `xcomp`), in which case it descends to the lower clause and restarts the search there.

4.3 Parser Evaluation

The evaluation was performed using the same criteria as in Rimell et al. (2009). A dependency was considered correctly recovered if the gold-standard head and dependent were correct and the label was an "acceptable match" to the gold-standard label, indicating the grammatical function of the extracted element at least to the level of subject, passive subject, object, or adjunct.

The evaluation in Rimell et al. (2009) took into account a wide variety of parser output formats, some of which differed significantly from the gold-standard. Since MSTParser and MaltParser produced Stanford dependencies for this experiment, evaluation required less manual examination than for some of the other parsers, as was also the case for the output of the Stanford parser in the original evaluation. However, a manual evaluation was still performed in order to resolve questionable cases.

5 Results

The results are shown in Table 1, where the accuracy for each construction is the percentage of gold-standard dependencies recovered correctly. The *Avg* column represents a macroaverage, i.e. the average of the individual scores on the seven constructions, while the *WAvg* column represents

Parser	ObRC	ObRed	SbRC	Free	ObQ	RNR	SbEm	Avg	WAvg
MST	34.1	47.3	78.9	65.5	13.8	**45.4**	**37.6**	46.1	63.4
Malt	**40.7**	**50.5**	**84.2**	**70.2**	16.2	39.7	23.5	46.4	**66.9**
MST-Q					**41.2**			**50.0**	
Malt-Q					31.2			48.5	

Table 1: Parser accuracy on the unbounded dependency corpus.

Parser	ObRC	ObRed	SbRC	Free	ObQ	RNR	SbEm	Avg	WAvg
C&C	**59.3**	62.6	80.0	72.6	**81.2**	**49.4**	22.4	**61.1**	69.9
Enju	47.3	**65.9**	82.1	**76.2**	32.5	47.1	32.9	54.9	**70.9**
MST	34.1	47.3	78.9	65.5	41.2	45.4	**37.6**	50.0	63.4
Malt	40.7	50.5	**84.2**	70.2	31.2	39.7	23.5	48.5	66.9
DCU	23.1	41.8	56.8	46.4	27.5	40.8	5.9	34.6	47.0
RASP	16.5	1.1	53.7	17.9	27.5	34.5	15.3	23.8	34.1
Stanford	22.0	1.1	74.7	64.3	41.2	45.4	10.6	37.0	50.3

Table 2: Parser accuracy on the unbounded dependency corpus. The ObQ score for C&C, MSTParser, and MaltParser is for a model trained with additional questions (without this C&C scored 27.5; MSTParser and MaltParser as in Table 1).

a weighted macroaverage, where the constructions are weighted proportionally to their relative frequency in the PTB. WAvg excludes ObQ sentences, since frequency statistics were not available for this construction in Rimell et al. (2009).

Our first observation is that the accuracies for both systems are considerably below the ∼90% unlabeled and ∼88% labeled attachment scores for English that have been reported previously (McDonald and Pereira, 2006; Hall et al., 2006). Comparing the two parsers, we see that MaltParser is more accurate on dependencies in relative clause constructions (ObRC, ObRed, SbRC, and Free), where argument relations tend to be relatively local, while MSTParser is more accurate on dependencies in RNR and SbEm, which involve more distant relations. Without the additional QB training data, the average scores for the two parsers are indistinguishable, but MSTParser appears to have been better able to take advantage of the question training, since MST-Q performs better than Malt-Q on ObQ sentences. On the weighted average MaltParser scores 3.5 points higher, because the constructions on which it outperforms MSTParser are more frequent in the PTB, and because WAvg excludes ObQ, where MSTParser is more accurate.

Table 2 shows the results for MSTParser and MaltParser in the context of the other parsers evaluated in Rimell et al. (2009).[5] For the parsers which have a model trained on questions, namely C&C, MSTParser, and MaltParser, the figure shown for ObQ sentences is that of the question model. It can be seen that MSTParser and MaltParser perform below C&C and Enju, but above the other parsers, and that MSTParser achieves the highest score on SbEm sentences and MaltParser on SbRC sentences. It should be noted, however, that Table 2 does not represent a direct comparison across all parsers, since most of the other parsers would have benefited from heuristic post-processing of the kind implemented here for MSTParser and MaltParser. This is especially true for RASP, where the grammar explicitly leaves some types of attachment decisions for post-processing. For DCU, improved labeling heuristics would significantly improve performance. It is instructive to compare the dependency parsers to the Stanford parser, which uses the same output representation and has been used to prepare the training data for our experiments. Stanford has very low recall on ObRed and SbEm, the categories where heuristic inference plays the largest role, but mirrors MSTParser for most other categories.

6 Error Analysis

We now proceed to a more detailed error analysis, based on the development sets, and classify

[5] The average scores reported differ slightly from those in Rimell et al. (2009), where a microaverage (i.e., average over all dependencies in the corpus, regardless of construction) was reported.

the errors made by the parsers into three categories: A *global* error is one where the parser completely fails to build the relevant clausal structure – the relative clause in ObRC, ObRed, SbRC, Free, SbEmb; the interrogative clause in ObQ; and the clause headed by the higher conjunct in RNR – often as a result of surrounding parsing errors. When a global error occurs, it is usually meaningless to further classify the error, which means that this category excludes the other two. An Arg error is one where the parser has constructed the relevant clausal structure but fails to find the Arg relation – in the simple and complex cases – or the set of surrounding Arg relations needed to infer an implicit Arg relation – in the indirect case (cf. Section 4.2). A Link error is one where the parser fails to find the crucial Link relation – `rcmod` in ObRC, ObRed, SbRC, SbEmb; `conj` in RNR (cf. Section 4.2). Link errors are not relevant for Free and ObQ, where all the crucial relations are clause-internal.

Table 3 shows the frequency of different error types for MSTParser (first) and MaltParser (second) in the seven development sets. First of all, we can see that the overall error distribution is very similar for the two parsers, which is probably due to the fact that they have been trained on exactly the same data with exactly the same annotation (unlike the five parsers previously evaluated). However, there is a tendency for MSTParser to make fewer Link errors, especially in the relative clause categories ObRC, ObRed and SbRC, which is compatible with the observation from the test results that MSTParser does better on more global dependencies, while MaltParser has an advantage on more local dependencies, although this is not evident from the statistics from the relatively small development set.

Comparing the different grammatical constructions, we see that Link errors dominate for the relative clause categories ObRC, ObRed and SbRC, where the parsers make very few errors with respect to the internal structure of the relative clauses (in fact, no errors at all for MaltParser on SbRC). This is different for SbEm, where the analysis of the argument structure is more complex, both because there are (at least) two clauses involved and because the unbounded dependency can only be inferred indirectly from the basic dependency representation (cf. Section 4.2). Another category where Arg errors are frequent is RNR, where all such errors consist in attaching the relevant dependent to the second conjunct instead of to the first.[6] Thus, in the example in Figure 1**f**, both parsers found the `conj` relation between **puzzled** and **angered** but attached **by** to the second verb.

Type	Global	Arg	Link	A+L	Errors	# Deps
ObRC	0/1	1/1	7/11	5/3	13/16	20
ObRed	0/1	0/1	6/7	3/4	9/13	23
SbRC	2/1	1/0	7/13	0/0	10/14	43
Free	2/1	3/5	–	–	5/6	22
ObQ	4/7	13/13	–	–	17/20	25
RNR	6/4	4/6	0/0	4/5	14/15	28
SbEm	3/4	3/2	0/0	3/3	9/9	13

Table 3: Distribution of error types in the development sets; frequencies for MSTParser listed first and MaltParser second. The columns Arg and Link give frequencies for Arg/Link errors occurring without the other error type, while A+L give frequencies for joint Arg and Link errors.

Global errors are most frequent for RNR, probably indicating that coordinate structures are difficult to parse in general, and for ObQ (especially for MaltParser), probably indicating that questions are not well represented in the training set even after the addition of QB data.[7] As noted in Section 4.1, this may be partly due to the fact that conversion to Stanford dependencies did not seem to work as well for QB as for the WSJ data. Another problem is that the part-of-speech tagger used was trained on WSJ data only and did not perform as well on the ObQ data. Uses of *What* as a determiner were consistently mistagged as pronouns, which led to errors in parsing. Thus, for the example in Figure 1**e**, both parsers produced the correct analysis except that, because of the tagging error, they treated **What** rather than **effect** as the head of the *wh*-phrase, which counts as an error in the evaluation.

In order to get a closer look specifically at the Arg errors, Table 4 gives the confusion matrix

[6] In the Stanford scheme, an argument or adjunct must be attached to the first conjunct in a coordination to indicate that it belongs to both conjuncts.

[7] Parsers trained without QB had twice as many global errors.

	Sb	Ob	POb	EmSb	EmOb	Other	Total
Sb	–	0/0	0/0	0/0	0/0	2/1	2/1
Ob	2/3	–	0/0	0/1	0/0	4/2	6/6
POb	2/0	7/5	–	0/0	0/0	5/8	14/13
EmSb	1/1	4/2	0/0	–	0/0	1/2	6/5
EmOb	0/0	3/1	0/0	0/0	–	1/6	4/7
Total	5/4	14/8	0/0	0/1	0/0	13/19	32/32

Table 4: Confusion matrix for Arg errors (excluding RNR and using parsers trained on QB for ObQ); frequencies for MSTParser listed first and MaltParser second. The column Other covers errors where the function is left unspecified or the argument is attached to the wrong head.

for such errors, showing which grammatical functions are mistaken for each other, with an extra category Other for cases where the function is left unspecified by the parser or the error is an attachment error rather than a labeling error (and excluding the RNR category because of the special nature of the Arg errors in this category). The results again confirm that the two parsers make very few errors on subjects and objects clause-internally. The few cases where an object is mistaken as a subject occur in ObQ, where both parsers perform rather poorly in general. By contrast, there are many more errors on prepositional objects and on embedded subjects and objects. We believe an important part of the explanation for this pattern is to be found in the Stanford dependency representation, where subjects and objects are marked as such but all other functions realized by *wh* elements are left unspecified (using the generic *rel* dependency), which means that the recovery of these functions currently has to rely on heuristic rules as described in Section 4.2. Finally, we think it is possible to observe the tendency for MaltParser to be more accurate at local labeling decisions – reflected in fewer cross-label confusions – and for MSTParser to perform better on more distant attachment decisions – reflected in fewer errors in the Other category (and in fewer Link errors).

7 Conclusion

In conclusion, the capacity of MSTParser and MaltParser to recover unbounded dependencies is very similar on the macro and weighted macro level, but there is a clear distinction in their strengths – constructions involving more distant dependencies such as ObQ, RNR and SbEm for MSTParser and constructions with more locally defined configurations such as ObRC, ObRed, SbRC and Free for MaltParser. This is a pattern that has been observed in previous evaluations of the parsers and can be explained by the global learning and inference strategy of MSTParser and the richer feature space of MaltParser (McDonald and Nivre, 2007).

Perhaps more interestingly, the accuracies of MSTParser and MaltParser are only slightly below the best performing systems in Rimell et al. (2009) – C&C and Enju. This is true even though MSTParser and MaltParser have not been engineered specifically for English and lack special mechanisms for handling unbounded dependencies, beyond the simple post-processing heuristic used to extract them from the output trees. Thus, it is reasonable to speculate that the addition of such mechanisms could lead to computationally lightweight parsers with the ability to extract unbounded dependencies with high accuracy.

Acknowledgments

We thank Marie-Catherine de Marneffe for great help with the Stanford parser and dependency scheme, Lluís Màrquez and Jesús Giménez for great support with SVMTool, Josef van Genabith for sharing the QuestionBank data, and Stephen Clark and Mark Steedman for helpful comments on the evaluation process and the paper. Laura Rimell was supported by EPSRC grant EP/E035698/1 and Carlos Gómez-Rodríguez by MEC/FEDER (HUM2007-66607-C04) and Xunta de Galicia (PGIDIT07SIN005206PR, Redes Galegas de PL e RI e de Ling. de Corpus, Bolsas Estadas INCITE/FSE cofinanced).

References

Black, E., S. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of 4th DARPA Workshop*, 306–311.

Briscoe, T., J. Carroll, and R. Watson. 2006. The second release of the RASP system. In *Proceedings* *of the COLING/ACL 2006 Interactive Presentation Sessions*, 77–80.

Buchholz, S. and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, 149–164.

Cahill, A., M. Burke, R. O'Donovan, J. Van Genabith, and A. Way. 2004. Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based LFG approximations. In *Proceedings of ACL*, 320–327.

Carreras, X., M. Collins, and T. Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of CoNLL*, 9–16.

Charniak, E. and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL*, 173–180.

Clark, S. and J. R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33:493–552.

de Marneffe, M.-C., B. MacCartney, and C. D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

Giménez, J. and L. Màrquez. 2004. SVMTool: A general POS tagger generator based on support vector machines. In *Proceedings of LREC*.

Hall, J., J. Nivre, and J. Nilsson. 2006. Discriminative classifiers for deterministic dependency parsing. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, 316–323.

Hall, J., J. Nilsson, J. Nivre, G. Eryiğit, B. Megyesi, M. Nilsson, and M. Saers. 2007. Single malt or blended? A study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task*.

Huang, L. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL*, 586–594.

Judge, J., A. Cahill, and J. van Genabith. 2006. QuestionBank: Creating a corpus of parse-annotated questions. In *Proceedings of COLING-ACL*, 497–504.

Klein, D. and C. D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL*, 423–430.

Kübler, S., R. McDonald, and J. Nivre. 2008. *Dependency Parsing*. Morgan and Claypool.

McDonald, R. and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of EMNLP-CoNLL*, 122–131.

McDonald, R. and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, 81–88.

McDonald, R. and G. Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of IWPT*, 122–131.

McDonald, R., K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, 91–98.

McDonald, R., K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL*, 216–220.

McDonald, R.. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Miyao, Y. and J. Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of ACL*, 83–90.

Nivre, J., J. Hall, and J. Nilsson. 2006a. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, 2216–2219.

Nivre, J., J. Hall, J. Nilsson, G. Eryiğit, and S. Marinov. 2006b. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of CoNLL*, 221–225.

Nivre, J., J. Hall, J. Nilsson, A. Chanev, G. Eryiğit, S. Kübler, S. Marinov, and E. Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13:95–135.

Nivre, J. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of ACL-IJCNLP*, 351–359.

Rimell, L., S. Clark, and M. Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings EMNLP*, 813–821.

Sagae, K. and A. Lavie. 2006. Parser combination by reparsing. In *Proceedings of NAACL HLT: Short Papers*, 129–132.

Co-STAR: A Co-training Style Algorithm for Hyponymy Relation Acquisition from Structured and Unstructured Text

Jong-Hoon Oh, Ichiro Yamada, Kentaro Torisawa, and Stijn De Saeger
Language Infrastructure Group, MASTAR Project,
National Institute of Information and Communications Technology (NICT)
{rovellia,iyamada,torisawa,stijn}@nict.go.jp

Abstract

This paper proposes a co-training style algorithm called Co-STAR that acquires hyponymy relations simultaneously from structured and unstructured text. In Co-STAR, two independent processes for hyponymy relation acquisition – one handling structured text and the other handling unstructured text – collaborate by repeatedly exchanging the knowledge they acquired about hyponymy relations. Unlike conventional co-training, the two processes in Co-STAR are applied to different source texts and training data. We show the effectiveness of this algorithm through experiments on large-scale hyponymy-relation acquisition from Japanese Wikipedia and Web texts. We also show that Co-STAR is robust against noisy training data.

1 Introduction

Acquiring semantic knowledge, especially semantic relations between lexical terms, is regarded as a crucial step in developing high-level natural language applications. This paper proposes Co-STAR (a **Co**-training **ST**yle **A**lgorithm for hyponymy **R**elation acquisition from structured and unstructured text). Similar to co-training (Blum and Mitchell, 1998), two hyponymy relation extractors in Co-STAR, one for structured and the other for unstructured text, iteratively collaborate to boost each other's performance.

Many algorithms have been developed to automatically acquire semantic relations from structured and unstructured text. Because term pairs are encoded in structured and unstructured text in different styles, different kinds of evidence have been used for semantic relation acquisition:

Evidence from unstructured text: lexico-syntactic patterns and distributional similarity (Ando et al., 2004; Hearst, 1992; Pantel et al., 2009; Snow et al., 2006; De Saeger et al., 2009; Van Durme and Pasca, 2008);

Evidence from structured text: topic hierarchy, layout structure of documents, and HTML tags (Oh et al., 2009; Ravi and Pasca, 2008; Sumida and Torisawa, 2008; Shinzato and Torisawa, 2004).

Recently, researchers have used both structured and unstructured text for semantic-relation acquisition, with the aim of exploiting such different kinds of evidence at the same time. They either tried to improve semantic relation acquisition by putting the different evidence together into a single classifier (Pennacchiotti and Pantel, 2009) or to improve the coverage of semantic relations by combining and ranking the semantic relations obtained from two source texts (Talukdar et al., 2008).

In this paper we propose an algorithm called Co-STAR. The main contributions of this work can be summarized as follows.

- Co-STAR is a semi-supervised learning method composed of two parallel and iterative processes over structured and unstructured text. It was inspired by bilingual co-training, which is a framework for hyponymy relation acquisition from source texts in two languages (Oh et al., 2009). Like bilingual co-training, two processes in Co-STAR operate independently on structured text and unstructured text. These two processes are trained in a supervised manner with their initial training data and then each of them tries to enlarge the existing training data of the other by iteratively exchanging what they have learned (more precisely, by transferring reliable classification results on common instances to one another) (see Section 4 for comparison Co-STAR and bilingual co-training). Unlike the ensemble semantic framework (Pennacchiotti and Pantel, 2009), Co-STAR does not have a single "*master*" classifier or ranker to integrate the different evidence found in structured and unstructured text. We experimentally show that, at least in our setting, Co-STAR works better than a single "*master*" classifier.

- Common relation instances found in both structured and unstructured text act as a *communication channel* between the two acquisition processes. Each process in Co-STAR classifies common relation instances and then transfers its high-confidence classification results to training data of the other process (as shown in Fig. 1), in order to improve classification results of the other process. Moreover, the efficiency of this exchange can be boosted by increasing the "bandwidth" of this channel. For this purpose each separate acquisition process automatically generates a set of relation instances that are likely to be *negative*. In our experiments, we show that the above idea proved highly effective.

- Finally, the acquisition algorithm we propose is robust against noisy training data. We show this by training one classifier in Co-STAR with manually labeled data and training the other with automatically generated but noisy training data. We found that Co-STAR performs well in this setting. This issue is discussed in Section 6.

This paper is organized as follows. Sections 2 and 3 precisely describe our algorithm. Section 4 describes related work. Sections 5 and 6 describe our experiments and present their results. Conclusions are drawn in Section 7.

2 Co-STAR

Co-STAR consists of two processes that simultaneously but independently extract and classify hyponymy relation instances from structured and unstructured text. The core of Co-STAR is the collaboration between the two processes, which continually exchange and compare their acquired knowledge on hyponymy relations. This collaboration is made possible through common instances shared by both processes. These common instances are classified separately by each process, but high-confidence classification results by one process can be transferred as new training data to the other.

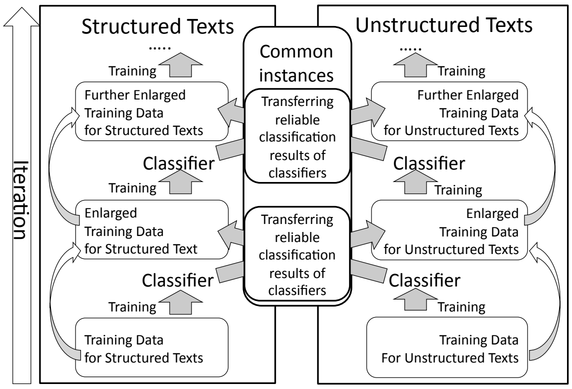


Figure 1: Concept of *Co-STAR*.

2.1 Common Instances

Let S and U represent a source (i.e. corpus) of structured and unstructured text, respectively. In this paper, we use the hierarchical layout of Wikipedia articles and the Wikipedia category system as structured text S (see Section 3.1), and a corpus of ordinary Web pages as unstructured text U. Let X_S and X_U denote a set of hyponymy relation candidates extracted from S and U, respectively. X_S is extracted from the hierarchical layout of Wikipedia articles (Oh et al., 2009) and X_U is extracted by lexico-syntactic patterns for hyponymy relations (i.e., *hyponym* such as *hyponymy*) (Ando et al., 2004) (see Section 3 for a detailed explanation)

We define two types of common instances, called "*genuine*" common instances (G) and "*virtual*" common instances (V). The set of common instances is denoted by $Y = G \cup V$. Genuine common instances are hyponymy relation candidates found in both S and U ($G = X_S \cap X_U$). On the other hand, term pairs are obtained as virtual common instances when:

- 1) they are extracted as hyponymy relation candidates in either S or U and;
- 2) they do not seem to be a hyponymy relation in the other text

The first condition corresponds to $X_S \oplus X_U$. Term pairs satisfying the second condition are defined as R_S and R_U, where $R_S \cap X_S = \phi$ and $R_U \cap X_U = \phi$.

R_S contains term pairs that are found in the Wikipedia category system but neither term appears as ancestor of the other[1]. For example, (*nutrition,protein*) and (*viruses,viral disease*), respectively, hold a category-article relation, where *nutrition* is not ancestor of *viruses* and vice versa in the Wikipedia category system. Here, term pairs, such as (*nutrition, viruses*) and (*viral disease, nutrition*), can be ones in R_S.

R_U is a set of term pairs extracted from U when:

- they are not hyponymy relation candidates in X_U and;
- they regularly co-occur in the same sentence as arguments of the same verb (e.g., A *cause* B or A *is made by* B);

As a result, term pairs in R_U are thought as holding some other semantic relations (e.g., A and B in "A *cause* B" may hold a cause/effect relation) than hyponymy relation. Finally, virtual common instances are defined as:

- $V = (X_S \oplus X_U) \cap (R_S \cup R_U)$

The virtual common instances, from the viewpoint of either S or U, are unlikely to hold a hyponymy relation even if they are extracted as hyponymy relation candidates in the other text. Thus many virtual common instances would be a negative example for hyponymy relation acquisition. On the other hand, genuine common instances (hyponymy relation candidates found in both S and U) are more likely to hold a hyponymy relation than virtual common instances.

In summary, genuine and virtual common instances can be used as different ground for collaboration as well as broader collaboration channel between the two processes than genuine common instances used alone.

2.2 Algorithm

We assume that classifier c assigns class label $cl \in \{yes, no\}$ ("yes" (hyponymy relation) or "no" (not a hyponymy relation)) to instances in $x \in X$ with confidence value $r \in \mathbb{R}^+$, a non-negative real number. We denote the classification result by classifier c as $c(x) = (x, cl, r)$. We used support vector machines (SVMs) in our experiments and the absolute value of the distance between a sample and the hyperplane determined by the SVMs as confidence value r.

```
**Input**: Common instances ($Y = G \cup V$) and the initial training data ($L_S^0$ and $L_U^0$)
**Output**: Two classifiers ($c_S^n$ and $c_U^n$)
$i = 0$
**repeat**
    $c_S^i := LEARN(L_S^i)$
    $c_U^i := LEARN(L_U^i)$
    $CR_S^i := \{c_S^i(y) | y \in Y, y \notin L_S^i \cup L_U^i\}$
    $CR_U^i := \{c_U^i(y) | y \in Y, y \notin L_S^i \cup L_U^i\}$
    **for** each $(y, cl_S, r_S) \in TopN(CR_S^i)$ and $(y, cl_U, r_U) \in CR_U^i$ **do**
        **if** ($r_S > \alpha$ and $r_U < \beta$)
        or ($r_S > \alpha$ and $cl_S = cl_U$) **then**
            $L_U^{(i+1)} := L_U^{(i+1)} \cup \{(y, cl_S)\}$
        **end if**
    **end for**
    **for** each $(y, cl_U, r_U) \in TopN(CR_U^i)$ and $(y, cl_S, r_S) \in CR_S^i$ **do**
        **if** ($r_U > \alpha$ and $r_S < \beta$)
        or ($r_U > \alpha$ and $cl_S = cl_U$) **then**
            $L_S^{(i+1)} := L_S^{(i+1)} \cup \{(y, cl_U)\}$
        **end if**
    **end for**
    $i = i + 1$
**until** stop condition is met
```

Figure 2: Co-STAR algorithm

[1] A term pair often holds a hyponymy relation if one term in the term pair is a parent of the other in the Wikipedia category system (Suchanek et al., 2007).

The Co-STAR algorithm is given in Fig. 2. The algorithm is interpreted as an iterative procedure 1) to train classifiers (c_U^i, c_S^i) with the existing training data (L_S^i and L_U^i) and 2) to select new training instances from the common instances to be added to existing training data. These are repeated until stop condition is met.

In the initial stage, two classifiers c_S^0 and c_U^0 are trained with manually prepared labeled instances (or training data) L_S^0 and L_U^0, respectively. The learning procedure is denoted by $c = LEARN(L)$ in lines 5–6, where c is a resulting classifier. Then c_S^i and c_U^i are applied to classify common instances in Y (lines 7–8). We denote CR_S^i as a set of the classification results of c_S^i for common instances, which are not included in the current training data $L_S^i \cup L_U^i$. Lines 9–13 describe a way of selecting instances in CR_S^i to be added to the existing training data in U. During the selection, c_S^i acts as a teacher and c_U^i as a student. $TopN(CR_S^i)$ is a set of $c_S^i(y) = (y, cl_S, r_S)$, whose r_S is the top-N highest in CR_S^i. (In our experiments, $N = 900$.) The teacher instructs his student the class label of y if the teacher can decide the class label of y with a certain level of confidence ($r_S > \alpha$) and the student satisfies one of the following two conditions:

- the student agrees with the teacher on class label of y ($cl_S = cl_U$) or

- the student's confidence in classifying y is low ($r_U < \beta$)

$r_U < \beta$ enables the teacher to instruct his student in spite of their disagreement over a class label. If one of the two conditions is satisfied, (y, cl_S) is added to existing labeled instances $L_U^{(i+1)}$. The roles are reversed in lines 14–18, so that c_U^i becomes the teacher and c_S^i the student.

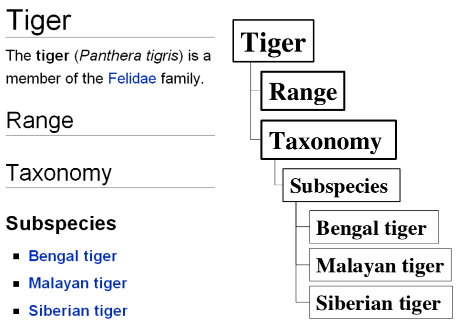
The iteration stops if the change in the difference between the two classifiers is stable enough. The stability is estimated by $d(c_S^i, c_U^i)$ in Eq. (1), where σ^i represents the change in the average difference between the confidence values of the two classifiers in classifying common instances. We terminate the iteration if $d(c_S^i, c_U^i)$ is smaller than 0.001 in three consecutive rounds (Wang and Zhou, 2007).

$$d(c_S^i, c_U^i) = |\sigma^i - \sigma^{(i-1)}|/|\sigma^{(i-1)}| \qquad (1)$$

3 Hyponymy Relation Acquisition

In this section we explain how each process extracts hyponymy relations from its respective text source either Wikipedia or Web pages. Each process extracts hyponymy relation candidates (denoted by (*hyper,hypo*) in this section). Because there are many non-hyponymy relations in these candidates[2], we classify hyponymy relation candidates into correct hyponymy relation or not. We used SVMs (Vapnik, 1995) for the classification in this paper.

3.1 Acquisition from Wikipedia



(a) Layout structure (b) Tree structure

Figure 3: Example borrowed from Oh et al. (2009): Layout and tree structures of Wikipedia article TIGER

We follow the method in Oh et al. (2009) for acquiring hyponymy relations from the Japanese Wikipedia. Every article is transformed into a tree structure as shown in Fig. 3, based on the items in its hierarchical layout including *title*, *(sub)section headings*, and *list items*. Candidate relations are extracted from this tree structure by regarding a node as a hypernym candidate and all of its subordinate nodes as potential hyponyms of the hypernym candidate (e.g., (TIGER, TAXONOMY) and (TIGER, SIBERIAN TIGER) from Fig. 3). We obtained 1.9×10^7 Japanese hyponymy relation candidates from Wikipedia.

[2] Only 25–30% of candidates was true hyponymy relation in our experiments.

	Type	Description
Feature from Wikipedia ("WikiFeature")	Lexical	Morphemes and POS of *hyper* and *hypo*; *hyper* and *hypo* themselves
	Structure	Distance between *hyper* and *hypo* in a tree structure; Lexical patterns for article or section names, where listed items often appear; Frequently used section headings in Wikipedia (e.g., "Reference"); Layout item type (e.g., section or list); Tree node type (e.g., root or leaf); Parent and children nodes of *hyper* and *hypo*
	Infobox	Attribute type and its value obtained from Wikipedia infoboxes
Feature from Web texts ("WebFeature")	Lexical	Morphemes and POS of *hyper* and *hypo*; *hyper* and *hypo* themselves
	Pattern	Lexico-syntactic patterns applied to *hyper* and *hypo*; PMI score between pattern and hyponymy relation candidate *(hyper,hypo)*
	Collocation	PMI score between *hyper* and *hypo*
	Noun Class	Noun classes relevant to *hyper* and *hypo*

Table 1: Feature sets (WikiFeature and WebFeature): *hyper* and *hypo* represent hypernym and hyponym parts of hyponymy relation candidates, respectively.

As features for classification we used lexical, structure, and infobox information from Wikipedia (WikiFeature), as shown in Table 1. Because they are the same feature sets as those used in Oh et al. (2009), here we just give a brief overview of the feature sets. Lexical features[3] are used to recognize the lexical evidence for hyponymy relations encoded in *hyper* and *hypo*. For example, the common head morpheme *tiger* in (TIGER, BENGAL TIGER) can be used as the lexical evidence. Such information is provided along with the words/morphemes and the parts of speech of *hyper* and *hypo*, which can be multi-word/morpheme nouns.

Structure features provide evidence found in layout or tree structures for hyponymy relations. For example, hyponymy relations (TIGER, BENGAL TIGER) and (TIGER,MALAYAN TIGER) can be obtained from tree structure "(root node, children nodes of *Subspecies*)" in Fig 3.

3.2 Acquisition from Web Texts

As the target for hyponymy relation acquisition from the Web, we used 5×10^7 pages from the TSUBAKI corpus (Shinzato et al., 2008), a 10^8 page Japanese Web corpus that was dependency parsed with KNP (Kurohashi-Nagao Parser) (Kurohashi and Kawahara, 2005). Hyponymy relation candidates are extracted from the corpus based on the lexico-syntactic patterns such as "*hypo* nado *hyper* (*hyper* such as *hypo*)" and "*hypo* to iu *hyper* (*hyper* called *hypo*)" (Ando et al., 2004). We extracted 6×10^6 Japanese hyponymy relation candidates from the Japanese Web texts. Features (WebFeature) used for classification are summarized in Table 1. Similar to the hyponymy relation acquisition from Wikipedia, lexical features are used to recognize the lexical evidence for hyponymy relations.

Lexico-syntactic patterns for hyponymy relation show different coverage and accuracy in hyponymy relation acquisition (Ando et al., 2004). Further if multiple lexico-syntactic patterns support acquisition of hyponymy relation candidates, these candidates are more likely to be actual hyponymy relations. The pattern feature of hyponymy relation candidates is used for these evidence.

We use PMI (point-wise mutual information) of hyponymy relation candidate (*hyper*, *hypo*) as a collocation feature (Pantel and Ravichandran, 2004), where we assume that *hyper* and *hypo* in candidates would frequently co-occur in the same sentence if they hold a hyponymy relation.

Semantic noun classes have been regarded as useful information in semantic relation acquisition (De Saeger et al., 2009). EM-based clustering (Kazama and Torisawa, 2008) is used for obtaining 500 semantic noun classes[4] from 5×10^5 nouns (including single-word and multi-word ones) and their 4×10^8 dependency relations with 5×10^5 verbs and other nouns in our target Web

[3] MeCab (`http://mecab.sourceforge.net/`) was used to provide the lexical features.

[4] Because EM clustering provides a probability distribution over noun class nc, we obtain discrete classes of each noun n with a probability threshold $p(nc|n) \geq 0.2$ (De Saeger et al., 2009).

	Co-training (Blum and Mitchell, 1998)	Bilingual co-training (Oh et al., 2009)	Co-STAR (Proposed method)
Instance space	Same	Different	Almost different
Feature space	Split by human decision	Split by languages	Split by source texts
Common instances	Genuine-common (or All unlabeled) instances	Genuine-common instances (Translatable)	Genuine-common and virtual-common instances

Table 2: Differences among co-training, bilingual co-training, and Co-STAR

corpus. For example, noun class C_{311} includes biological or chemical substances such as *tatou* (*polysaccharide*) and *yuukikagoubutsu* (*organic compounds*). Noun classes (i.e., C_{311}) relevant to *hyper* and *hypo*, respectively, are used as a noun class feature.

4 Related Work

There are two frameworks, which are most relevant to our work – bilingual co-training and ensemble semantics.

The main difference between bilingual co-training and Co-STAR lies in an instance space. In bilingual co-training, instances are in different spaces divided by languages while, in Co-STAR, many instances are in different spaces divided by their source texts. Table 2 shows differences between co-training, bilingual co-training and Co-STAR.

Ensemble semantics is a relation acquisition framework, where semantic relation candidates are extracted from multiple sources and a single ranker ranks or classifies the candidates in the final step (Pennacchiotti and Pantel, 2009). In ensemble semantics, one ranker is in charge of ranking all candidates extracted from multiple sources; while one classifier classifies candidates extracted from one source in Co-STAR.

5 Experiments

We used the July version of Japanese Wikipedia (jawiki-20090701) as structured text. We randomly selected 24,000 hyponymy relation candidates from those identified in Wikipedia and manually checked them. 20,000 of these samples were used as training data for our initial classifier, the rest was equally divided into development and test data for Wikipedia. They are called "WikiSet." As unstructured text, we used 5×10^7 Japanese Web pages in the TSUBAKI corpus (Shinzato et al., 2008). Here, we manually checked 9,500 hyponymy relation candidates selected randomly from Web texts. 7,500 of these were used as training data. The rest was split into development and test data. We named this data "WebSet".

In both classifiers, the development data was used to select the optimal parameters, and the test data was used to evaluate our system. We used TinySVM (TinySVM, 2002) with a polynomial kernel of degree 2 as a classifier. α (the threshold value indicating high confidence), β (the threshold value indicating low confidence), and $TopN$ (the maximum number of training instances to be added to the existing training data in each iteration) were selected through experiments on the development set. The combination of $\alpha = 1$, $\beta = 0.3$, and $TopN$=900 showed the best performance and was used in the following experiments. Evaluation was done by precision (P), recall (R), and F-measure (F).

5.1 Results

We compare six systems. Three of these, B1–B3, show the effect of different feature sets ("WikiFeature" and "WebFeature" in Table 1) and different training data. We trained two separate classifiers in B1 and B2, while we integrated feature sets and training data for training a single classifier in B3. The classifiers in these three systems are trained with manually prepared training data ("WikiSet" and "WebSet"). For the purpose of our experiment, we consider **B3** as the closest possible approximation of the ensemble semantics framework (Pennacchiotti and Pantel, 2009).

- **B1** consists of two completely independent classifiers. Both S and U classifiers are trained and tested on their own feature and data sets (respectively "WikiSet + WikiFeature" and "WebSet + WebFeature").

- **B2** is the same as B1, except that both classifiers are trained with all available training data — WikiSet and WebSet are combined (27,500 training instances in total). However, each classifier only uses its own feature set (WikiFeature or WebFeature)[5].

- **B3** adds a *master* classifier to **B1**. This third classifier is trained on the complete 27,500 training instances (same as **B2**) using all available features from Table 1, including each instance's SVM scores obtained from the two **B1** classifiers[6]. The verdict of the master classifier is considered to be the final classification result.

The other three systems, BICO, Co-B, and Co-STAR (our proposed method), are for comparison between bilingual co-training (Oh et al., 2009) (BICO) and variants of Co-STAR (Co-B and Co-STAR). Especially, we prepared Co-B and Co-STAR to show the effect of different configurations of common instances on the Co-STAR algorithm. We use both B1 and B2 as the initial classifiers of Co-B and Co-STAR. We notate Co-B and Co-STAR without '∗' when B1 is used as their initial classifier and those with '∗' when B2 is used.

- **BICO** implements the bilingual co-training algorithm of (Oh et al., 2009), in which two processes collaboratively acquire hyponymy relations in two *different languages*. For BICO, we prepared 20,000 English and 20,000 Japanese training samples (Japanese ones are the same as training data in the WikiSet) by hand.

- **Co-B** is a variant of Co-STAR that uses only the genuine-common instances as common instances (67,000 instances)[7], to demonstrate the effectiveness of the virtual common instances.

- **Co-STAR** is our proposed method, which uses both genuine-common and virtual-common instances (643,000 instances in total).

	WebSet			WikiSet		
	P	R	F	P	R	F
B1	84.3	65.2	73.5	87.8	74.7	80.7
B2	83.4	69.6	75.9	87.4	79.5	83.2
B3	82.2	72.0	76.8	86.1	77.7	81.7
BICO	N/A	N/A	N/A	84.5	**81.8**	83.1
Co-B	**86.2**	63.5	73.2	**89.7**	74.1	81.2
Co-B∗	85.5	69.9	77.0	89.6	76.5	82.5
Co-STAR	85.9	76.0	80.6	88.0	**81.8**	**84.8**
Co-STAR∗	83.3	**80.7**	**82.0**	87.6	**81.8**	84.6

Table 3: Comparison of different systems

Table 3 summarizes the result. Features for common instances in Co-B and Co-STAR are prepared in the same way as training data in B2, so that both classifiers can classify the common instances with their trained feature sets.

Comparison between B1–B3 shows that B2 and B3 outperform B1 in F-measure. More training data used in B2–B3 (27,500 instances for both WebSet and WikiSet) results in higher performance than that of B1 (7,500 and 20,000 instances used separately). We think that the lexical features, assigned regardless of source text to instances in B2–B3, are mainly responsible for the performance gain over B1, as they are the least domain-dependent type of features. B2–B3 are composed of different number of classifiers, each of which is trained with different feature sets and training instances. Despite this difference, B2 and B3 showed similar performance in F-measure.

Co-STAR outperformed the algorithm similar to the ensemble semantics framework (B3), although we admit that a more extensive comparison is desirable. Further Co-STAR outperformed BICO. While the manual cost for building the initial training data used in Co-STAR and BICO is hard to quantify, Co-STAR achieves better performance with fewer training data in total (27,500 instances) than BICO (40,000 instances). The difference in performance between Co-B and Co-STAR shows the effectiveness of

[5] Note that training instances from WebSet (or WikiSet) can have WikiFeature (or WebFeature) if they also appear in Wikipedia (or Web corpus). But they can always have lexical feature, the common feature set between WikiFeature and WebFeature.

[6] SVM scores are assigned to the instances in training data in a 10-fold cross validation manner.

[7] Co-B can be considered as conventional co-training (Blum and Mitchell, 1998) in the sense that two classifiers collaborate through actual common instances.

the automatically generated virtual-common instances. From these comparison, we can see that virtual-common instances coupled with genuine-common instances can be leveraged to enable more effective collaboration between the two classifiers in Co-STAR.

As a result, our proposed method outperforms the others in F-measure by 1.4–8.5%. We obtained 4.3×10^5 hyponymy relations from Web texts and 4.6×10^6 ones from Wikipedia[8].

6 Co-STAR with Automatically Generated Training Data

For Co-STAR, we need two sets of manually prepared training data, one for structured text and the other for unstructured text. As in any other supervised system, the cost of preparing the training data is an important issue. We therefore investigated whether Co-STAR can be trained for a lower cost by generating more of its training data automatically.

We automatically built training data for Web texts by using definition sentences[9] and category names in the Wikipedia articles, while we stuck to manually prepared training data for Wikipedia. To obtain hypernyms from Wikipedia article names, we used definition-specific lexico-syntactic patterns such as "*hyponym* is *hypernym*" and "*hyponym* is a type of *hypernym*" (Kazama and Torisawa, 2007; Sumida and Torisawa, 2008). Then, we extracted hyponymy relations consisting of pairs of Wikipedia category names and their member articles when the Wikipedia category name and the hypernym obtained from the definition of the Wikipedia article shared the same head word. Next, we selected a subset of the extracted hyponymy relations that are also hyponymy relation candidates in Web texts, as positive instances for hyponymy relation acquisition from Web text. We obtained around 15,000 positive instances in this way. Negative instances were chosen from virtual-common instances, which also originated from the Wikipedia category system and hyponymy relation candidates in Web texts (around 293,000 instances).

The automatically built training data was noisy and its size was much bigger than manually prepared training data in WebSet. Thus 7,500 instances as training data (the same number of manually built training data in WebSet) were randomly chosen from the positive and negative instances with a positive:negative ratio of 1:4[10].

	WebSet			WikiSet		
	P	R	F	P	R	F
B1	81.0	47.6	60.0	**87.8**	74.7	80.7
B2	80.0	55.4	65.5	87.1	79.5	83.1
B3	82.0	33.7	47.8	87.1	75.6	81.0
Co-STAR	**82.2**	60.8	69.9	87.3	80.7	83.8
Co-STAR$*$	79.2	**69.6**	**74.1**	87.0	**81.8**	**84.4**

Table 4: Results with automatically generated training data

With the automatically built training data for Web texts and manually prepared training data for Wikipedia, we evaluated B1–B3 and Co-STAR, which are the same systems in Table 3. The results in Table 4 are encouraging. Co-STAR was robust even when faced with noisy training data. Further Co-STAR showed better performance than B1–B3, although its performance in Table 4 dropped a bit compared to Table 3. This result shows that we can reduce the cost of manually preparing training data for Co-STAR with only small loss of the performance.

7 Conclusion

This paper proposed Co-STAR, an algorithm for hyponymy relation acquisition from structured and unstructured text. In Co-STAR, two independent processes of hyponymy relation acquisition from structured texts and unstructured texts, collaborate in an iterative manner through common instances. To improve this collaboration, we introduced virtual-common instances.

Through a series of experiments, we showed that Co-STAR outperforms baseline systems and virtual-common instances can be leveraged to achieve better performance. We also showed that Co-STAR is robust against noisy training data, which requires less human effort to prepare it.

[8] We obtained them with 90% precision by setting the SVM score threshold to 0.23 for Web texts and 0.1 for Wikipedia.

[9] The first sentences of Wikipedia articles.

[10] We select the ratio by testing different ratio from 1:2 to 1:5 with our development data in WebSet and B1.

References

Ando, Maya, Satoshi Sekine, and Shun Ishiza. 2004. Automatic extraction of hyponyms from Japanese newspaper using lexico-syntactic patterns. In *Proc. of LREC '04*.

Blum, Avrim and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

De Saeger, Stijn, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, and Masaki Murata. 2009. Large scale relation acquisition using class dependent patterns. In *Proc. of ICDM 2009*, pages 764–769.

Hearst, Marti A. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics*, pages 539–545.

Kazama, Jun'ichi and Kentaro Torisawa. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *Proc. of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 698–707.

Kazama, Jun'ichi and Kentaro Torisawa. 2008. Inducing gazetteers for named entity recognition by large-scale clustering of dependency relations. In *Proceedings of ACL-08: HLT*, pages 407–415.

Kurohashi, Sadao and Daisuke Kawahara. 2005. KNP (Kurohashi-Nagao Parser) 2.0 users manual.

Oh, Jong-Hoon, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Bilingual co-training for monolingual hyponymy-relation acquisition. In *Proc. of ACL-09: IJCNLP*, pages 432–440.

Pantel, Patrick and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *Proc. of HLT-NAACL '04*, pages 321–328.

Pantel, Patrick, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of EMNLP '09*, pages 938–947.

Pennacchiotti, Marco and Patrick Pantel. 2009. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 238–247.

Ravi, Sujith and Marius Pasca. 2008. Using structured text for large-scale attribute extraction. In *CIKM-08*, pages 1183–1192.

Shinzato, Keiji and Kentaro Torisawa. 2004. Extracting hyponyms of prespecified hypernyms from itemizations and headings in web documents. In *Proceedings of COLING '04*, pages 938–944.

Shinzato, Keiji, Tomohide Shibata, Daisuke Kawahara, Chikara Hashimoto, and Sadao Kurohashi. 2008. Tsubaki: An open search engine infrastructure for developing new information access. In *Proceedings of IJCNLP '08*, pages 189–196.

Snow, Rion, Daniel Jurafsky, and Andrew Y. Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 801–808.

Suchanek, Fabian M., Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A Core of Semantic Knowledge. In *Proc. of WWW '07*, pages 697–706.

Sumida, Asuka and Kentaro Torisawa. 2008. Hacking Wikipedia for hyponymy relation acquisition. In *Proc. of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, pages 883–888, January.

Talukdar, Partha Pratim, Joseph Reisinger, Marius Pasca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proc. of EMNLP08*, pages 582–590.

TinySVM. 2002. `http://chasen.org/~taku/software/TinySVM`.

Van Durme, Benjamin and Marius Pasca. 2008. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *Proc. of AAAI08*, pages 1243–1248.

Vapnik, Vladimir N. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Wang, Wei and Zhi-Hua Zhou. 2007. Analyzing co-training style algorithms. In *ECML '07: Proceedings of the 18th European conference on Machine Learning*, pages 454–465.

Simple and Efficient Algorithm for Approximate Dictionary Matching

Naoaki Okazaki
University of Tokyo
okazaki@is.s.u-tokyo.ac.jp

Jun'ichi Tsujii
University of Tokyo
University of Manchester
National Centre for Text Mining
tsujii@is.s.u-tokyo.ac.jp

Abstract

This paper presents a simple and efficient algorithm for approximate dictionary matching designed for similarity measures such as cosine, Dice, Jaccard, and overlap coefficients. We propose this algorithm, called *CPMerge*, for the τ-overlap join of inverted lists. First we show that this task is solvable exactly by a τ-overlap join. Given inverted lists retrieved for a query, the algorithm collects fewer candidate strings and prunes unlikely candidates to efficiently find strings that satisfy the constraint of the τ-overlap join. We conducted experiments of approximate dictionary matching on three large-scale datasets that include person names, biomedical names, and general English words. The algorithm exhibited scalable performance on the datasets. For example, it retrieved strings in 1.1 ms from the string collection of Google Web1T unigrams (with cosine similarity and threshold 0.7).

1 Introduction

Languages are sufficiently flexible to be able to express the same meaning through different diction. At the same time, inconsistency of surface expressions has persisted as a serious problem in natural language processing. For example, in the biomedical domain, *cardiovascular disorder* can be described using various expressions: *cardiovascular diseases*, *cardiovascular system disorder*, and *disorder of the cardiovascular system*. It is a nontrivial task to find the entry from these surface expressions appearing in text.

This paper addresses *approximate dictionary matching*, which consists of finding all strings in a string collection V such that they have similarity that is no smaller than a threshold α with a query string x. This task has a broad range of applications, including spelling correction, flexible dictionary look-up, record linkage, and duplicate detection (Henzinger, 2006; Manku et al., 2007).

Formally, the task obtains a subset $\mathcal{Y}_{x,\alpha} \subseteq V$,

$$\mathcal{Y}_{x,\alpha} = \{y \in V \mid \mathrm{sim}(x, y) \geq \alpha\}, \qquad (1)$$

where $\mathrm{sim}(x, y)$ presents the similarity between x and y. A naïve solution to this task is to compute similarity values $|V|$ times, i.e., between x and every string $y \in V$. However, this solution is impractical when the number of strings $|V|$ is huge (e.g., more than one million).

In this paper, we present a simple and efficient algorithm for approximate dictionary matching designed for similarity measures such as cosine, Dice, Jaccard, and overlap coefficients. Our main contributions are twofold.

1. We show that the problem of approximate dictionary matching is solved exactly by a *τ-overlap join* (Sarawagi and Kirpal, 2004) of inverted lists. Then we present *CPMerge*, which is a simple and efficient algorithm for the τ-overlap join. In addition, the algorithm is easily implemented.

2. We demonstrate the efficiency of the algorithm on three large-scale datasets with person names, biomedical concept names,

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 851–859,
Beijing, August 2010

and general English words. We compare the algorithm with state-of-the-art algorithms, including Locality Sensitive Hashing (Ravichandran et al., 2005; Andoni and Indyk, 2008) and DivideSkip (Li et al., 2008). The proposed algorithm retrieves strings the most rapidly, e.g., in 1.1 ms from Google Web1T unigrams (with cosine similarity and threshold 0.7).

2 Proposed Method

2.1 Necessary and sufficient conditions

In this paper, we assume that the *features* of a string are represented arbitrarily by a set. Although it is important to design a string representation for an accurate similarity measure, we do not address this problem: our emphasis is *not on designing a better representation for string similarity* but *on establishing an efficient algorithm*.

The most popular representation is given by *n-grams*: all substrings of size n in a string. We use trigrams throughout this paper as an example of string representation. For example, the string “methyl sulphone” is expressed by 17 elements of letter trigrams[1], {'$$m', '$me', 'met', 'eth', 'thy', 'hyl', 'yl_', 'l_s', '_su', 'sul', 'ulp', 'lph', 'pho', 'hon', 'one', 'ne$', 'e$$'}. We insert two $s before and after the string to denote the start or end of the string. In general, a string x consisting of $|X|$ letters yields $(|x| + n - 1)$ elements of n-grams. We call $|x|$ and $|X|$ the *length* and *size*, respectively, of the string x.

Let X and Y denote the feature sets of the strings x and y, respectively. The cosine similarity between the two strings x and y is,

$$\text{cosine}(X, Y) = \frac{|X \cap Y|}{\sqrt{|X||Y|}}. \tag{2}$$

Table 1: Conditions for each similarity measure

Measure	$\min |Y|$	$\max |Y|$	$\tau (= \min |X \cap Y|)$
Dice	$\frac{\alpha}{2-\alpha}|X|$	$\frac{2-\alpha}{\alpha}|X|$	$\frac{1}{2}\alpha(|X| + |Y|)$
Jaccard	$\alpha|X|$	$|X|/\alpha$	$\frac{\alpha(|X|+|Y|)}{1+\alpha}$
Cosine	$\alpha^2|X|$	$|X|/\alpha^2$	$\alpha\sqrt{|X||Y|}$
Overlap	—	—	$\alpha \min\{|X|, |Y|\}$

By integrating this definition with Equation 1, we obtain the necessary and sufficient condition for approximate dictionary matching,

$$\left\lceil \alpha\sqrt{|X||Y|} \right\rceil \leq |X \cap Y| \leq \min\{|X|, |Y|\}. \tag{3}$$

This inequality states that two strings x and y must have at least $\tau = \left\lceil \alpha\sqrt{|X||Y|} \right\rceil$ features in common. When ignoring $|X \cap Y|$ in the inequality, we have an inequality about $|X|$ and $|Y|$,

$$\left\lceil \alpha^2|X| \right\rceil \leq |Y| \leq \left\lfloor \frac{|X|}{\alpha^2} \right\rfloor \tag{4}$$

This inequality presents the search range for retrieving similar strings; that is, we can ignore strings whose feature size is out of this range. Other derivations are also applicable to similarity measures, including Dice, Jaccard, and overlap coefficients. Table 1 summarizes the conditions for these similarity measures.

We explain one usage of these conditions. Let query string x = “methyl sulphone” and threshold for approximate dictionary matching $\alpha = 0.7$ with cosine similarity. Representing the strings with letter trigrams, we have the size of x, $|X| = 17$. The inequality 4 gives the search range of $|Y|$ of the retrieved strings, $9 \leq |Y| \leq 34$. Presuming that we are searching for strings of $|Y| = 16$, we obtain the necessary and sufficient condition for the approximate dictionary matching from the inequality 3, $\tau = 12 \leq |X \cap Y|$. Thus, we need to search for strings that have at least 12 letter trigrams that overlap with X. When considering a string y = “methyl sulfone”, which is a spelling variant of y (*ph* → *f*), we confirm that the string is a solution for approximate dictionary matching because $|X \cap Y| = 13$ $(\geq \tau)$. Here, the actual similarity is $\text{cosine}(X, Y) = 13/\sqrt{17 \times 16} = 0.788$ $(\geq \alpha)$.

2.2 Data structure and algorithm

Algorithm 1 presents the pseudocode of the approximate dictionary matching based on Table 1.

[1] In practice, we attach ordinal numbers to n-grams to represent multiple occurrences of n-grams in a string (Chaudhuri et al., 2006). For example, the string “prepress”, which contains two occurrences of the trigram 'pre', yields the set {'$$p'#1, '$pr'#1, 'pre'#1, 'rep'#1, 'epr'#1, 'pre'#2, 'res'#1, 'ess'#1, 'ss$'#1, 's$$'#1}.

```
Input: $V$: collection of strings
Input: $x$: query string
Input: $\alpha$: threshold for the similarity
Output: $\mathcal{Y}$: list of strings similar to the query
1 $X \leftarrow$ string_to_feature($x$);
2 $\mathcal{Y} \leftarrow$[];
3 for $l \leftarrow$ min_y($|X|$, $\alpha$) to max_y($|X|$, $\alpha$) do
4     $\tau \leftarrow$ min_overlap($|X|$, $l$, $\alpha$);
5     $R \leftarrow$ overlapjoin($X$, $\tau$, $V$, $l$);
6     foreach $r \in R$ do append $r$ to $\mathcal{Y}$;
7 end
8 return $\mathcal{Y}$;
```

Algorithm 1: Approximate dictionary matching.

Given a query string x, a collection of strings V, and a similarity threshold α, the algorithm computes the size range (line 3) given by Table 1. For each size l in the range, the algorithm computes the minimum number of overlaps τ (line 4). The function `overlapjoin` (line 5) finds similar strings by solving the following problem (τ-*overlap join*): given a list of features of the query string X and the minimum number of overlaps τ, enumerate strings of size l in the collection V such that they have at least τ feature overlaps with X.

To solve this problem efficiently, we build an inverted index that stores a mapping from the features to their originating strings. Then, we can perform the τ-overlap join by finding strings that appear at least τ times in the inverted lists retrieved for the query features X.

Algorithm 2 portrays a naïve solution for the τ-overlap join (AllScan algorithm). In this algorithm, function `get`(V, l, q) returns the inverted list of strings (of size l) for the feature q. In short, this algorithm scans strings in the inverted lists retrieved for the query features X, counts the frequency of occurrences of every string in the inverted lists, and returns the strings whose frequency of occurrences is no smaller than τ.

```
Input: $X$: array of features of the query string
Input: $\tau$: minimum number of overlaps
Input: $V$: collection of strings
Input: $l$: size of target strings
Output: $R$: list of strings similar to the query
$M \leftarrow \{\}$;
$R \leftarrow []$;
foreach $q \in X$ do
    foreach $i \in$ get($V$, $l$, $q$) do
        $M[i] \leftarrow M[i] + 1$;
        if $\tau \leq M[i]$ then
            append $i$ to $R$;
        end
    end
end
return $R$;
```

Algorithm 2: AllScan algorithm.

This algorithm is inefficient in that it scans all strings in the inverted lists. The number of scanned strings is large, especially when some query features appear frequently in the strings, e.g., `'s$$'` (words ending with 's') and `'pre'` (words with substring 'pre'). To make matters worse, such features are too common for characterizing string similarity. The AllScan algorithm is able to maintain numerous candidate strings in M, but most candidates are not likely to qualified because they have few overlaps with X.

To reduce the number of the candidate strings, we refer to *signature-based algorithms* (Arasu et al., 2006; Chaudhuri et al., 2006):

Property 1 *Let there be a set (of size h) X and a set (of any size) Y. Consider any subset $Z \subseteq X$ of size $(h - \tau + 1)$. If $|X \cap Y| \geq \tau$, then $Z \cap Y \neq \phi$.*

We explain one usage of this property. Let query string x = "methyl sulphone" and its trigram set X be features (therefore, $|X| = h = 17$). Presuming that we seek strings whose trigrams are size 16 and have 12 overlaps with X, then string y must have at least one overlap with any subset of size 6 (= 17 − 12 + 1) of X. We call the subset *signatures*. The property leads to an algorithmic design by which we obtain a small set of candidate strings from the inverted lists for signatures, $(|X| - \tau + 1)$ features in X, and verify whether each candidate string satisfies the τ overlap with the remaining $(\tau - 1)$ n-grams.

Algorithm 3 presents the pseudocode employing this idea. In line 1, we arrange the features in X in ascending order of the number of strings in their inverted lists. We denote the k-th element in the ordered features as X_k ($k \in \{0, ..., |X| - 1\}$), where the index number begins with 0. Based on this notation, X_0 and $X_{|X|-1}$ are the most uncommon and the most common features in X, respectively.

In lines 2–7, we use $(|X| - \tau + 1)$ features

```
    Input: $X$: array of features of the query string
    Input: $\tau$: minimum number of overlaps
    Input: $V$: collection of strings
    Input: $l$: size of target strings
    Output: $R$: list of strings similar to the query
1   sort elements in $X$ by order of $|$get $(V, l, X_k)|$;
2   $M \leftarrow \{\}$;
3   for $k \leftarrow 0$ to $(|X| - \tau)$ do
4       foreach $s \in$ get $(V, l, X_k)$ do
5           $M[s] \leftarrow M[s] + 1$;
6       end
7   end
8   $R \leftarrow []$;
9   for $k \leftarrow (|X| - \tau + 1)$ to $(|X| - 1)$ do
10      foreach $s \in M$ do
11          if bsearch(get $(V, l, X_k)$, $s$) then
12              $M[s] \leftarrow M[s] + 1$;
13          end
14          if $\tau \leq M[s]$ then
15              append $s$ to $R$;
16              remove $s$ from $M$;
17          else if $M[s] + (|X| - k - 1) < \tau$ then
18              remove $s$ from $M$;
19          end
20      end
21  end
22  return $R$;
```

Algorithm 3: CPMerge algorithm.

$X_0, ..., X_{|X|-\tau}$ to generate a compact set of candidate strings. The algorithm stores the occurrence count of each string s in $M[s]$. In lines 9–21, we increment the occurrence counts if each of $X_{|X|-\tau+1}, ..., X_{|X|-1}$ inverted lists contain the candidate strings. For each string s in the candidates (line 10), we perform a binary search on the inverted list (line 11), and increment the overlap count if the string s exists (line 12). If the overlap counter of the string reaches τ (line 14), then we append the string s to the result list R and remove s from the candidate list (lines 15–16). We prune a candidate string (lines 17–18) if the candidate is found to be unreachable for τ overlaps even if it appears in all of the unexamined inverted lists.

3 Experiments

We report the experimental results of approximate dictionary matching on large-scale datasets with person names, biomedical names, and general English words. We implemented various systems of approximate dictionary matching.

- **Proposed**: CPMerge algorithm.
- **Naive**: Naïve algorithm that computes the cosine similarity $|V|$ times for every query.
- **AllScan**: AllScan algorithm.
- **Signature**: CPMerge algorithm without pruning; this is equivalent to Algorithm 3 without lines 17–18.
- **DivideSkip**: our implementation of the algorithm (Li et al., 2008)[2].
- **Locality Sensitive Hashing (LSH)** (Andoni and Indyk, 2008): This baseline system follows the design of previous work (Ravichandran et al., 2005). This system *approximately* solves Equation 1 by finding dictionary entries whose LSH values are within the (bit-wise) hamming distance of θ from the LSH value of a query string. To adapt the method to approximate dictionary matching, we used a 64-bit LSH function computed with letter trigrams. By design, this method does not find an exact solution to Equation 1; in other words, the method can miss dictionary entries that are actually similar to the query strings. This system has three parameters, θ, q (number of bit permutations), and B (search width), to control the tradeoff between retrieval speed and recall[3]. Generally speaking, increasing these parameters improves the recall, but slows down the speed. We determined $\theta = 24$ and $q = 24$ experimentally[4], and measured the performance when $B \in \{16, 32, 64\}$.

The systems, excluding LSH, share the same implementation of Algorithm 1 so that we can specifically examine the differences of the algorithms for τ-overlap join. The C++ source code of the system used for this experiment is available[5]. We ran all experiments on an application server running Debian GNU/Linux 4.0 with Intel Xeon 5140 CPU (2.33 GHz) and 8 GB main memory.

[2] We tuned parameter values $\mu \in \{0.01, 0.02, 0.04, 0.1, 0.2, 0.4, 1, 2, 4, 10, 20, 40, 100\}$ for each dataset. We selected the parameter with the fastest response.

[3] We followed the notation of the original paper (Ravichandran et al., 2005) here. Refer to the original paper for definitions of the parameters θ, q, and B.

[4] q was set to 24 so that the arrays of shuffled hash values are stored in memory. We chose $\theta = 24$ from $\{8, 16, 24\}$ because it showed a good balance between accuracy and speed.

[5] http://www.chokkan.org/software/simstring/

3.1 Datasets

We used three large datasets with person names (IMDB actors), general English words (Google Web1T), and biomedical names (UMLS).

- **IMDB actors**: This dataset comprises actor names extracted from the IMDB database[6]. We used all actor names (1,098,022 strings; 18 MB) from the file `actors.list.gz`. The average number of letter trigrams in the strings is 17.2. The total number of trigrams is 42,180. The system generated index files of 83 MB in 56.6 s.
- **Google Web1T unigrams**: This dataset consists of English word unigrams included in the Google Web1T corpus (LDC2006T13). We used all word unigrams (13,588,391 strings; 121 MB) in the corpus after removing the frequency information. The average number of letter trigrams in the strings is 10.3. The total number of trigrams is 301,459. The system generated index files of 601 MB in 551.7 s.
- **UMLS**: This dataset consists of English names and descriptions of biomedical concepts included in the Unified Medical Language System (UMLS). We extracted all English concept names (5,216,323 strings; 212 MB) from `MRCONSO.RRF.aa.gz` and `MRCONSO.RRF.ab.gz` in UMLS Release 2009AA. The average number of letter trigrams in the strings is 43.6. The total number of trigrams is 171,596. The system generated index files of 1.1 GB in 1216.8 s.

For each dataset, we prepared 1,000 query strings by sampling strings randomly from the dataset. To simulate the situation where query strings are not only identical but also similar to dictionary entries, we introduced random noise to the strings. In this experiment, one-third of the query strings are unchanged from the original (sampled) strings, one-third of the query strings have one letter changed, and one-third of the query strings have two letters changed. When changing a letter, we randomly chose a letter position from a uniform distribution, and replaced the letter at the position with an ASCII letter randomly chosen from a uniform distribution.

3.2 Results

To examine the scalability of each system, we controlled the number of strings to be indexed from 10%–100%, and issued 1,000 queries. Figure 1 portrays the average response time for retrieving strings whose cosine similarity values are no smaller than 0.7. Although LSH (B=16) seems to be the fastest in the graph, this system missed many true positives[7]; the recall scores of approximate dictionary matching were 15.4% (IMDB), 13.7% (Web1T), and 1.5% (UMLS). Increasing the parameter B improves the recall at the expense of the response time. LSH (B=64)[8]. It not only ran slower than the proposed method, but also suffered from low recall scores, 25.8% (IMDB), 18.7% (Web1T), and 7.1% (UMLS). LSH was useful only when we required a quick response much more than recall.

The other systems were guaranteed to find the exact solution (100% recall). The proposed algorithm was the fastest of all exact systems on all datasets: the response times per query (100% index size) were 1.07 ms (IMDB), 1.10 ms (Web1T), and 20.37 ms (UMLS). The response times of the Naïve algorithm were too slow, 32.8 s (IMDB), 236.5 s (Web1T), and 416.3 s (UMLS).

The proposed algorithm achieved substantial improvements over the AllScan algorithm: the proposed method was 65.3 times (IMDB), 227.5 times (Web1T), and 13.7 times (UMLS) faster than the Naïve algorithm. We observed that the Signature algorithm, which is Algorithm 3 without lines 17–18, did not perform well: The Signature algorithm was 1.8 times slower (IMDB), 2.1 times faster (Web1T), and 135.0 times slower (UMLS) than the AllScan algorithm. These results indicate that it is imperative to minimize the number of candidates to reduce the number of binary-search operations. The proposed algorithm was 11.1–13.4 times faster than DivideSkip.

Figure 2 presents the average response time

[6] `ftp://ftp.fu-berlin.de/misc/movies/database/`

[7] Solving Equation 1, all systems are expected to retrieve the exact set of strings retrieved by the Naïve algorithm.

[8] The response time of LSH (B=64) on the IMDB dataset was 29.72 ms (100% index size).

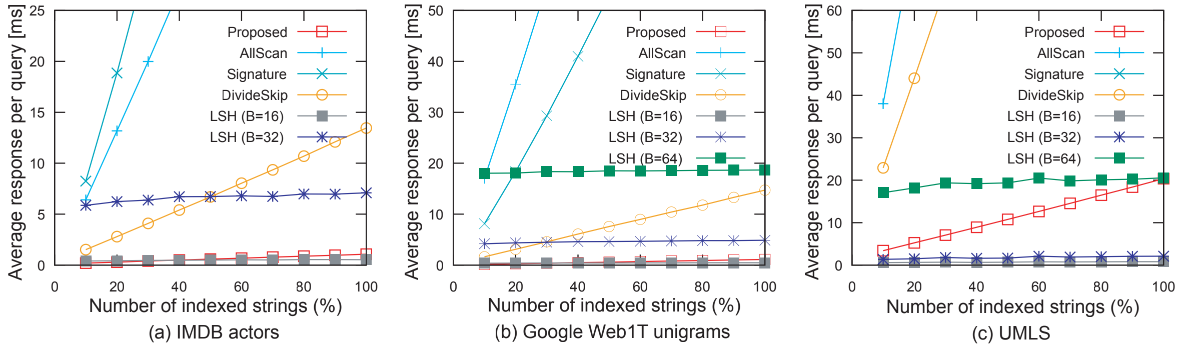


Figure 1: Average response time for processing a query (cosine similarity; $\alpha = 0.7$).

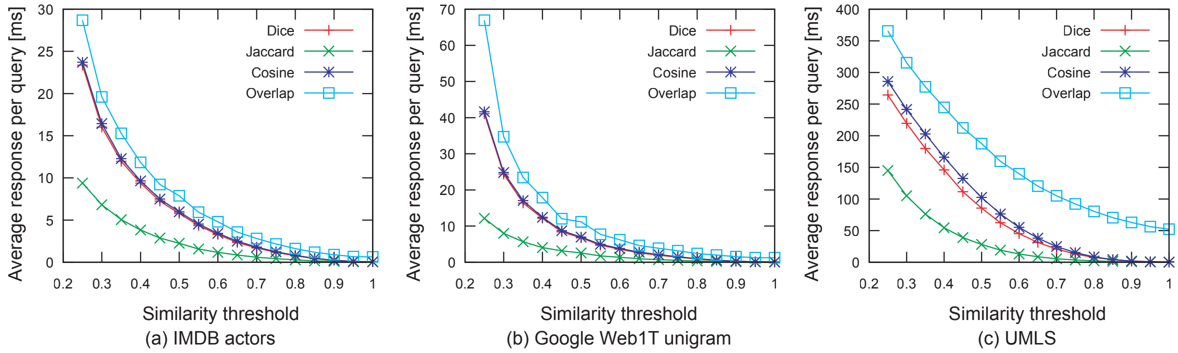


Figure 2: Average response time for processing a query.

of the proposed algorithm for different similarity measures and threshold values. When the similarity threshold is lowered, the algorithm runs slower because the number of retrieved strings $|\mathcal{Y}|$ increases exponentially. The Dice coefficient and cosine similarity produced similar curves.

Table 2 summarizes the run-time statistics of the proposed method for each dataset (with cosine similarity and threshold 0.7). Using the IMDB dataset, the proposed method searched for strings whose size was between 8.74 and 34.06; it retrieved 4.63 strings per query string. The proposed algorithm scanned 279.7 strings in 4.6 inverted lists to obtain 232.5 candidate strings. The algorithm performed a binary search on 4.3 inverted lists containing 7,561.8 strings in all. In contrast, the AllScan algorithm had to scan 16,155.1 strings in 17.7 inverted lists and considered 9,788.7 candidate strings, and found only 4.63 similar strings.

This table clearly demonstrates three key contributions of the proposed algorithm for efficient approximate dictionary matching. First, the proposed algorithm scanned far fewer strings than did the AllScan algorithm. For example, to obtain candidate strings in the IMDB dataset, the proposed algorithm scanned 279.7 strings, whereas the AllScan algorithm scanned 16,155.1 strings. Therefore, the algorithm examined only 1.1%–3.5% of the strings in the entire inverted lists in the three datasets. Second, the proposed algorithm considered far fewer candidates than did the AllScan algorithm: the number of candidate strings considered by the algorithm was 1.2%–6.6% of those considered by the AllScan algorithm. Finally, the proposed algorithm read fewer inverted lists than did the AllScan algorithm. The proposed algorithm actually read 8.9 (IMDB), 6.0 (Web1T), and 31.7 (UMLS) inverted lists during the experiments[9]. These values indicate that the proposed algorithm can solve τ-overlap join problems by checking only 50.3% (IMDB), 53.6% (Web1T), and 51.9% of the total inverted lists re-

[9] These values are 4.6 + 4.3, 3.1 + 2.9, and 14.3 + 17.4.

Table 2: Run-time statistics of the proposed algorithm for each dataset

Averaged item	IMDB	Web1T	UMLS	Description
$\min \lvert y \rvert$	8.74	5.35	21.87	minimum size of trigrams of target strings
$\max \lvert y \rvert$	34.06	20.46	88.48	maximum size of trigrams of target strings
τ	14.13	9.09	47.77	minimum number of overlaps required/sufficient per query
$\lvert \mathcal{Y} \rvert$	4.63	3.22	111.79	number of retrieved strings per query
Total				— averaged for each query and target size:
# inverted lists	17.7	11.2	61.1	number of inverted lists retrieved for a query
# strings	16 155.1	52 557.6	49 561.4	number of strings in the inverted list
# unique strings	9 788.7	44 834.6	17 457.5	number of unique strings in the inverted list
Candidate stage				— averaged for each query and target size:
# inverted lists	4.6	3.1	14.3	number of inverted lists scanned for generating candidates
# strings	279.7	552.7	1 756.3	number of strings scanned for generating candidates
# candidates	232.5	523.7	1 149.7	number of candidates generated for a query
Validation stage				— averaged for each query and target size:
# inverted lists	4.3	2.9	17.4	number of inverted lists examined by binary search for a query
# strings	7 561.8	19 843.6	20 443.7	number of strings targeted by binary search

trieved for queries.

4 Related Work

Numerous studies have addressed approximate dictionary matching. The most popular configuration uses n-grams as a string representation and the edit distance as a similarity measure. Gravano et al. (1998; 2001) presented various filtering strategies, e.g., count filtering, position filtering, and length filtering, to reduce the number of candidates. Kim et al. (2005) proposed two-level n-gram inverted indices (n-Gram/2L) to eliminate the redundancy of position information in n-gram indices. Li et al. (2007) explored the use of variable-length grams (VGRAMs) for improving the query performance. Lee et al. (2007) extended n-grams to include wild cards and developed algorithms based on a replacement semi-lattice. Xiao et al. (2008) proposed the Ed-Join algorithm, which utilizes mismatching n-grams.

Several studies addressed different paradigms for approximate dictionary matching. Bocek et al. (2007) presented the Fast Similarity Search (FastSS), an enhancement of the neighborhood generation algorithms, in which multiple variants of each string record are stored in a database. Wang et al. (2009) further improved the technique of neighborhood generation by introducing partitioning and prefix pruning. Huynh et al. (2006) developed a solution to the k-mismatch problem in compressed suffix arrays. Liu et al. (2008) stored string records in a trie, and proposed a framework called TITAN. These studies are specialized for the edit distance measure.

A few studies addressed approximate dictionary matching for similarity measures such as cosine and Jaccard similarities. Chaudhuri et al. (2006) proposed the SSJoin operator for similarity joins with several measures including the edit distance and Jaccard similarity. This algorithm first generates *signatures* for strings, finds all pairs of strings whose signatures overlap, and finally outputs the subset of these candidate pairs that satisfy the similarity predicate. Arasu et al. (2006) addressed *signature schemes*, i.e., methodologies for obtaining signatures from strings. They also presented an implementation of the SSJoin operator in SQL. Although we did not implement this algorithm in SQL, it is equivalent to the Signature algorithm in Section 3.

Sarawagi and Kirpal (2004) proposed the MergeOpt algorithm for the τ-overlap join to approximate string matching with overlap, Jaccard, and cosine measures. This algorithm splits inverted lists for a given query A into two groups, S and L, maintains a heap to collect candidate strings on S, and performs a binary search on L to verify the condition of the τ-overlap join for each candidate string. Their subsequent work includes an efficient algorithm for the top-k search of the overlap join (Chandel et al., 2006).

Li et al. (2008) extended this algorithm to the SkipMerge and DivideSkip algorithms. The SkipMerge algorithm uses a heap to compute the τ-overlap join on entire inverted lists A, but has an additional mechanism to increment the frontier pointers of inverted lists efficiently based on the strings popped most recently from the heap. Consequently, SkipMerge can reduce the number of strings that are pushed to the heap. Similarly to the MergeOpt algorithm, DivideSkip splits inverted lists A into two groups S and L, but it applies SkipMerge to S. In Section 3, we reported the performance of DivideSkip.

Charikar (2002) presented the Locality Sensitive Hash (LSH) function (Andoni and Indyk, 2008), which preserves the property of cosine similarity. The essence of this function is to map strings into N-bit hash values where the bitwise hamming distance between the hash values of two strings approximately corresponds to the angle of the two strings. Ravichandran et al. (2005) applied LSH to the task of noun clustering. Adapting this algorithm to approximate dictionary matching, we discussed its performance in Section 3.

Several researchers have presented refined similarity measures for strings (Winkler, 1999; Cohen et al., 2003; Bergsma and Kondrak, 2007; Davis et al., 2007). Although these studies are sometimes regarded as a research topic of approximate dictionary matching, they assume that two strings for the target of similarity computation are given; in other words, it is out of their scope to find strings in a large collection that are similar to a given string. Thus, it is a reasonable approach for an approximate dictionary matching to quickly collect candidate strings with a loose similarity threshold, and for a refined similarity measure to scrutinize each candidate string for the target application.

5 Conclusions

We present a simple and efficient algorithm for approximate dictionary matching with the cosine, Dice, Jaccard, and overlap measures. We conducted experiments of approximate dictionary matching on large-scale datasets with person names, biomedical names, and general English words. Even though the algorithm is very simple, our experimental results showed that the proposed algorithm executed very quickly. We also confirmed that the proposed method drastically reduced the number of candidate strings considered during approximate dictionary matching. We believe that this study will advance practical NLP applications for which the execution time of approximate dictionary matching is critical.

An advantage of the proposed algorithm over existing algorithms (e.g., MergeSkip) is that it does not need to read all the inverted lists retrieved by query n-grams. We observed that the proposed algorithm solved τ-overlap joins by checking approximately half of the inverted lists (with cosine similarity and threshold $\alpha = 0.7$). This characteristic is well suited to processing compressed inverted lists because the algorithm needs to decompress only half of the inverted lists. It is natural to extend this study to compressing and decompressing inverted lists for reducing disk space and for improving query performance (Behm et al., 2009).

Acknowledgments

This work was partially supported by Grants-in-Aid for Scientific Research on Priority Areas (MEXT, Japan) and for Solution-Oriented Research for Science and Technology (JST, Japan).

References

Andoni, Alexandr and Piotr Indyk. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122.

Arasu, Arvind, Venkatesh Ganti, and Raghav Kaushik. 2006. Efficient exact set-similarity joins. In *VLDB '06: Proceedings of the 32nd International Conference on Very Large Data Bases*, pages 918–929.

Behm, Alexander, Shengyue Ji, Chen Li, and Jiaheng Lu. 2009. Space-constrained gram-based indexing for efficient approximate string search. In *ICDE '09: Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 604–615.

Bergsma, Shane and Grzegorz Kondrak. 2007. Alignment-based discriminative string similarity. In *ACL '07: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 656–663.

Bocek, Thomas, Ela Hunt, and Burkhard Stiller. 2007. Fast similarity search in large dictionaries. Technical Report ifi-2007.02, Department of Informatics (IFI), University of Zurich.

Chandel, Amit, P. C. Nagesh, and Sunita Sarawagi. 2006. Efficient batch top-k search for dictionary-based entity recognition. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*.

Charikar, Moses S. 2002. Similarity estimation techniques from rounding algorithms. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388.

Chaudhuri, Surajit, Venkatesh Ganti, and Raghav Kaushik. 2006. A primitive operator for similarity joins in data cleaning. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*.

Cohen, William W., Pradeep Ravikumar, and Stephen E. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, pages 73–78.

Davis, Jason V., Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. 2007. Information-theoretic metric learning. In *ICML '07: Proceedings of the 24th International Conference on Machine Learning*, pages 209–216.

Gravano, Luis, Panagiotis G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan, and Divesh Srivastava. 2001. Approximate string joins in a database (almost) for free. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 491–500.

Henzinger, Monika. 2006. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR '06: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 284–291.

Huynh, Trinh N. D., Wing-Kai Hon, Tak-Wah Lam, and Wing-Kin Sung. 2006. Approximate string matching using compressed suffix arrays. *Theoretical Computer Science*, 352(1-3):240–249.

Kim, Min-Soo, Kyu-Young Whang, Jae-Gil Lee, and Min-Jae Lee. 2005. n-Gram/2L: a space and time efficient two-level n-gram inverted index structure. In *VLDB '05: Proceedings of the 31st International Conference on Very Large Data Bases*, pages 325–336.

Lee, Hongrae, Raymond T. Ng, and Kyuseok Shim. 2007. Extending q-grams to estimate selectivity of string matching with low edit distance. In *VLDB '07: Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 195–206.

Li, Chen, Bin Wang, and Xiaochun Yang. 2007. Vgram: improving performance of approximate queries on string collections using variable-length grams. In *VLDB '07: Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 303–314.

Li, Chen, Jiaheng Lu, and Yiming Lu. 2008. Efficient merging and filtering algorithms for approximate string searches. In *ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, pages 257–266.

Liu, Xuhui, Guoliang Li, Jianhua Feng, and Lizhu Zhou. 2008. Effective indices for efficient approximate string search and similarity join. In *WAIM '08: Proceedings of the 2008 The Ninth International Conference on Web-Age Information Management*, pages 127–134.

Manku, Gurmeet Singh, Arvind Jain, and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 141–150.

Navarro, Gonzalo and Ricardo Baeza-Yates. 1998. A practical q-gram index for text retrieval allowing errors. *CLEI Electronic Journal*, 1(2).

Ravichandran, Deepak, Patrick Pantel, and Eduard Hovy. 2005. Randomized algorithms and nlp: using locality sensitive hash function for high speed noun clustering. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 622–629.

Sarawagi, Sunita and Alok Kirpal. 2004. Efficient set joins on similarity predicates. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 743–754.

Wang, Wei, Chuan Xiao, Xuemin Lin, and Chengqi Zhang. 2009. Efficient approximate entity extraction with edit distance constraints. In *SIGMOD '09: Proceedings of the 35th SIGMOD International Conference on Management of Data*, pages 759–770.

Winkler, William E. 1999. The state of record linkage and current research problems. Technical Report R99/04, Statistics of Income Division, Internal Revenue Service Publication.

Xiao, Chuan, Wei Wang, and Xuemin Lin. 2008. Ed-Join: an efficient algorithm for similarity joins with edit distance constraints. In *VLDB '08: Proceedings of the 34th International Conference on Very Large Data Bases*, pages 933–944.

Latent Mixture of Discriminative Experts for Multimodal Prediction Modeling

Derya Ozkan, Kenji Sagae and Louis-Philippe Morency
USC Institute for Creative Technologies
{ozkan,sagae,morency}@ict.usc.edu

Abstract

During face-to-face conversation, people naturally integrate speech, gestures and higher level language interpretations to predict the right time to start talking or to give backchannel feedback. In this paper we introduce a new model called Latent Mixture of Discriminative Experts which addresses some of the key issues with multimodal language processing: (1) temporal synchrony/asynchrony between modalities, (2) micro dynamics and (3) integration of different levels of interpretation. We present an empirical evaluation on listener nonverbal feedback prediction (e.g., head nod), based on observable behaviors of the speaker. We confirm the importance of combining four types of multimodal features: lexical, syntactic structure, eye gaze, and prosody. We show that our Latent Mixture of Discriminative Experts model outperforms previous approaches based on Conditional Random Fields (CRFs) and Latent-Dynamic CRFs.

1 Introduction

Face-to-face communication is highly interactive. Even when only one person speaks at a time, other participants exchange information continuously amongst themselves and with the speaker through gestures, gaze and prosody. These different channels contain complementary information essential to interpretation and understanding of human behaviors (Oviatt, 1999). Psycholinguistic studies also suggest that gesture and speech come from a single underlying mental process, and they are related both temporally and semantically (McNeill, 1992; Cassell and Stone, 1999; Kendon, 2004).

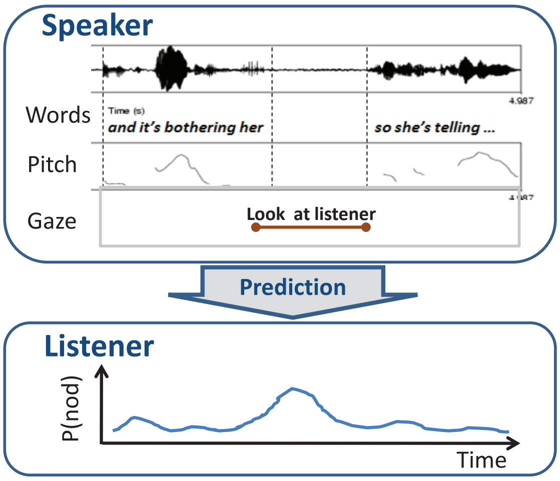


Figure 1: Example of multimodal prediction model: listener nonverbal backchannel prediction based on speaker's speech and eye gaze. As the speaker says the word *her*, which is the end of the clause (*her* is also the object of the verb *bothering*), and lowers the pitch while looking back at the listener and eventually pausing, the listener is then very likely to head nod (i.e., nonverbal backchannel).

A good example of such complementarity is how people naturally integrate speech, gestures and higher level language to predict when to give backchannel feedback. Building computational models of such a predictive process is challenging since it involves micro dynamics and temporal relationship between cues from different modalities (Quek, 2003). Figure 1 shows an example of backchannel prediction where a listener head nod is more likely. For example, a temporal sequence from the speaker where he/she reaches the end of segment (syntactic feature) with a low pitch and looks at the listener before pausing is a good opportunity for the listener to give nonverbal feedback (e.g., head nod). These prediction models have broad applicability, including the improvement of nonverbal behavior recognition, the synthesis of natural animations for robots and virtual humans, the training of cultural-specific nonverbal behaviors, and the diagnoses of social disorders (e.g., autism spectrum disorder).

In this paper we introduce a new model called Latent Mixture of Discriminative Experts (LMDE) which addresses some of the key issues with multimodal language processing: (1) temporal synchrony/asynchrony between modalities, (2) micro dynamics and (3) integration of different levels of interpretation. We present an empirical evaluation on nonverbal feedback prediction (e.g., head nod) confirming the importance of combining different types of multimodal features. We show that our LMDE model outperforms previous approaches based Conditional Random Fields (CRFs) and Latent-Dynamic CRFs.

2 Related Work

Earlier work in multimodal language processing focused on multimodal dialogue systems where the gestures and speech may be constrained (Johnston, 1998; Jurafsky et al., 1998). Most of the research in multimodal language processing over the past decade fits within two main trends that have emerged: (1) recognition of individual multimodal actions such as speech and gestures (e.g, (Eisenstein et al., 2008; Frampton et al., 2009; Gravano et al., 2007)), and (2) recognition/summarization of the social interaction between more than one participants (e.g., meeting analysis (Heylen and op den Akker, 2007; Moore, 2007; Murray and Carenini, 2009; Jovanovic et al., 2006)).

The work described in this paper can be seen from a third intermediate category where multimodal cues from one person is used to predict the social behavior of another participant. This type of predictive models has been mostly studied in the context of embodied conversational agents (Nakano et al., 2003; Nakano et al., 2007). In particular, backchannel feedback (the nods and paraverbals such as “uh-hu” and “mm-hmm” that listeners produce as someone is speaking) has received considerable interest due to its pervasiveness across languages and conversational contexts and this paper addresses the problem of how to predict and generate this important class of dyadic nonverbal behavior.

Several researchers have developed models to predict when backchannel should happen. In general, these results are difficult to compare as they utilize different corpora and present varying evaluation metrics. Ward and Tsukahara (2000) propose a unimodal approach where backchannels are associated with a region of low pitch lasting 110ms during speech. Models were produced manually through an analysis of English and Japanese conversational data. Nishimura et al. (2007) present a unimodal decision-tree approach for producing backchannels based on prosodic features. Cathcart et al. (2003) propose a unimodal model based on pause duration and trigram part-of-speech frequency. The model was constructed by identifying, from the HCRC Map Task Corpus (Anderson et al., 1991), trigrams ending with a backchannel. Fujie et al. (2004) used Hidden Markov Models to perform head nod recognition. In their paper, they combined head gesture detection with prosodic low-level features from the same person to determine strongly positive, weak positive and negative responses to yes/no type utterances.

In recent years, great research has shown the strength of latent variable models for natural language processing (Blunsom et al., 2008). One of the most relevant works is that of Eisenstein and Davis (2007), which presents a latent conditional model for fusion of multiple modalities (speech and gestures). One of the key difference of our work is that we are explicitly modeling the micro dynamics and temporal relationship between modalities.

3 Multimodal Prediction Models

Human face-to-face communication is a little like a dance, in that participants continuously adjust their behaviors based on verbal and nonverbal displays and signals. A topic of central interest in modeling such behaviors is the patterning of interlocutor actions and interactions, moment-by-moment, and one of the key challenges is identifying the patterns that best predict specific actions. Thus we are interested in developing predictive models of communication dynamics that integrate previous and current actions from all interlocutors to anticipate the most likely next actions of one or all interlocutors. Humans are good at this: they have an amazing ability to predict, at a micro-level, the actions of an interlocutor (Bavelas et al., 2000); and we know that better predictions can correlate with more empathy and better outcomes (Goldberg, 2005; Fuchs, 1987).

With turn-taking being perhaps the best-known example, we now know a fair amount about some aspects of communication dynamics, but much less about others. However, recent advances in machine learning and experimental methods, and recent findings from a variety of perspectives, including conversation analysis, social signal processing, adaptation, corpus analysis and modeling, perceptual experiments, and dialog systems-building and experimentation, mean that the time is ripe to start working towards more comprehensive predictive models.

The study of multimodal prediction models bring a new series of research challenges:

Multimodal asynchrony While speech and gestures seem to come from a single underlying mental process (McNeill, 1992), they not always happen at the same time, making it hard for earlier multimodal fusion approaches based on synchrony. A multimodal prediction model needs to be able to learn automatically the temporal relationship (and relative importance) between modalities.

Micro dynamics The dynamic between multimodal signals should be taken at a micro level since many of the interactions between speech and gesture happen at the sub-gesture level or sub-word level (Quek, 2003). Typical word-based sampling may not be sufficient and instead a higher sampling rate should be used.

Limited annotated data Given the time requirement to correctly annotate multimodal data, most multimodal datasets contain only a limited number of labeled examples. Since many machine learning algorithms rely on a large training corpus, effective training of a predictive model on multimodal datasets is challenging.

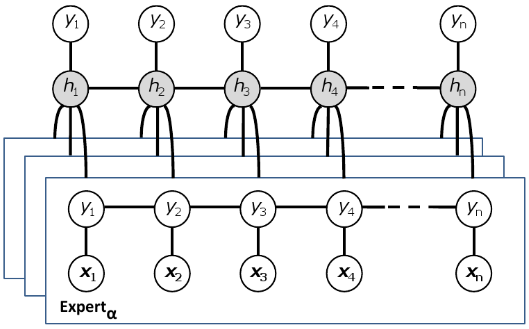


Figure 2: Latent Mixture of Discriminative Experts: a new dynamic model for multimodal fusion. In this graphical model, x_j represents the j^{th} multimodal observation, h_j is a hidden state assigned to x_j , and y_j the class label of x_j. Gray circles are latent variables. The micro dynamics and multimodal temporal relationships are automatically learned by the hidden states h_j during the learning phase.

4 Latent Mixture of Discriminative Experts

In this paper we present a multimodal fusion algorithm, called Latent Mixture of Discriminative Experts (shown in Figure 2), that addresses the three challenges discussed in the previous section. The hidden states of LMDE automatically learn the temporal asynchrony between modalities. By using a constant sample rate of 30Hz in our experiments, we can model the micro dynamics of speech and prosody (e.g., change of intonation in the middle of a word). And finally, by training separate experts for each modalities, we improve the prediction performance even with limited datasets.

The task of our LMDE model is to learn a mapping between a sequence of multimodal observations $\mathbf{x} = \{x_1, x_2, ..., x_m\}$ and a sequence of labels $\mathbf{y} = \{y_1, y_2, ..., y_m\}$. Each y_j is a class label for the j^{th} frame of a video sequence and is a member of a set $\mathcal{Y}$ of possible class labels, for example, $\mathcal{Y} = \{\texttt{head-nod}, \texttt{other-gesture}\}$.

Each frame observation x_j is represented by a feature vector $\phi(x_j) \in \mathbf{R}^d$, for example, the prosodic features at each sample. For each sequence, we also assume a vector of "sub-structure" variables $\mathbf{h} = \{h_1, h_2, ..., h_m\}$. These variables are not observed in the training examples and will therefore form a set of hidden variables in the model.

Following Morency et al. (2007), we define our LMDE model as follows:

$$P(\mathbf{y} \mid \mathbf{x}, \theta) = \sum_{\mathbf{h}} P(\mathbf{y} \mid \mathbf{h}, \mathbf{x}, \theta) P(\mathbf{h} \mid \mathbf{x}, \theta) \quad (1)$$

where θ is the model parameters that is to be estimated from training data.

To keep training and inference tractable, Morency et al. (2007) restrict the model to have disjoint sets of hidden states associated with each class label. Each h_j is a member of a set $\mathcal{H}_{y_j}$ of possible hidden states for the class label y_j. $\mathcal{H}$, the set of all possible hidden states, is defined to be the union of all $\mathcal{H}_y$ sets. Since sequences which have any $h_j \notin \mathcal{H}_{y_j}$ will by definition have $P(\mathbf{y} \mid \mathbf{h}, \mathbf{x}, \theta) = 0$, latent conditional model becomes:

$$P(\mathbf{y} \mid \mathbf{x}, \theta) = \sum_{\mathbf{h}:\forall h_j \in \mathcal{H}_{y_j}} P(\mathbf{h} \mid \mathbf{x}, \theta). \quad (2)$$

What differentiates our LMDE model from the original work of Morency et al. is the definition of $P(\mathbf{h}|\mathbf{x}, \theta)$:

$$P(\mathbf{h}| \mathbf{x}, \theta) = \frac{\exp\begin{pmatrix} \sum_l \theta_l \cdot \mathbf{T}_l(\mathbf{h}, \mathbf{x}) + \\ \sum_\alpha \theta_\alpha \cdot P_\alpha(\mathbf{y}|\mathbf{x}, \lambda_\alpha) \end{pmatrix}}{\mathcal{Z}(\mathbf{x}, \theta)}, \quad (3)$$

where $\mathcal{Z}$ is the partition function and $P_\alpha(\mathbf{y}|\mathbf{x})$ is the conditional distribution of the expert indexed by α. The expert conditional distributions are defined $P_\alpha(\mathbf{y}|\mathbf{x}, \lambda_\alpha)$ using the usual conditional random field formulation:

$$P_\alpha(\mathbf{y}| \mathbf{x}, \lambda_\alpha) = \frac{\exp\left(\sum_k \lambda_{\alpha,k} \cdot \mathbf{F}_{\alpha,k}(\mathbf{y}, \mathbf{x})\right)}{\mathcal{Z}_\alpha(\mathbf{x}, \lambda_\alpha)}, \quad (4)$$

$F_{\alpha,k}$ is defined as

$$\mathbf{F}_{\alpha,k}(\mathbf{y}, \mathbf{x}) = \sum_{j=1}^{m} f_{\alpha,k}(y_{j-1}, y_j, \mathbf{x}, j),$$

and each feature function $f_{\alpha,k}(y_{j-1}, y_j, \mathbf{x}, j)$ is either a state function $s_k(y_j, \mathbf{x}, j)$ or a transition function $t_k(y_{j-1}, y_j, \mathbf{x}, j)$. State functions s_k depend on a single hidden variable in the model while transition functions t_k can depend on pairs of hidden variables. $\mathbf{T}_l(\mathbf{h}, \mathbf{x})$, defined in Equation 3, is a special case, summing only over the transition feature functions $t_l(h_{l-1}, h_l, \mathbf{x}, l)$. Each expert α contains a different subset of $f_{\alpha,k}(y_{j-1}, y_j, \mathbf{x}, j)$. These feature functions are defined in Section 5.2.

4.1 Learning Model Parameters

Given a training set consisting of n labeled sequences $(\mathbf{x_i}, \mathbf{y_i})$ for $i = 1...n$, training is done in a two step process. First each expert α is trained following (Kumar and Herbert., 2003; Lafferty et al., 2001) objective function to learn the parameter λ^*_α:

$$L(\lambda_\alpha) = \sum_{i=1}^{n} \log P_\alpha(\mathbf{y}_i \mid \mathbf{x}_i, \lambda_\alpha) - \frac{1}{2\sigma^2}||\lambda_\alpha||^2 \quad (5)$$

The first term in Eq. 5 is the conditional log-likelihood of the training data. The second term is the log of a Gaussian prior with variance σ^2, i.e., $P(\lambda_\alpha) \sim \exp\left(\frac{1}{2\sigma^2}||\lambda_\alpha||^2\right)$.

Then the marginal probabilities $P_\alpha(y_j = a \mid \mathbf{y}, \mathbf{x}, \lambda^*_\alpha)$, are computed using belief propagation and used as input for Equation 3. The optimal parameter θ^* was learned using the log-likelyhood of the conditional probability defined in Equation 2 (i.e., no regularization).

4.2 Inference

For testing, given a new test sequence $\mathbf{x}$, we want to estimate the most probable sequence of labels $\mathbf{y}^*$ that maximizes our LMDE model:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \sum_{\mathbf{h}:\forall h_i \in \mathcal{H}_{y_i}} P(\mathbf{h} \mid \mathbf{x}, \theta^*) \quad (6)$$

5 Experimental Setup

We evaluate our Latent Mixture of Discriminative Experts on the multimodal task of predicting listener nonverbal backchannel (i.e., head nods). Backchannel feedback (the nods and paraverbals such as "uh-hu" and "mm-hmm" that listeners produce as some is speaking) has received considerable interest due to its pervasiveness across languages and conversational contexts.

5.1 Dataset

We are using the RAPPORT dataset from (Maatman et al., 2005), which contains 47 dyadic interactions between a speaker and a listener. Data is drawn from a study of face-to-face narrative discourse ("quasi-monologic" storytelling). In this dataset, participants in groups of two were told they were participating in a study to evaluate a communicative technology. Subjects were randomly assigned the role of speaker and listener. The speaker viewed a short segment of a video clip taken from the Edge Training Systems, Inc. Sexual Harassment Awareness video. After the speaker finished viewing the video, the listener was led back into the computer room, where the speaker was instructed to retell the stories portrayed in the clips to the listener. The listener was asked to not talk during the story retelling. Elicited stories were approximately two minutes in length on average. Participants sat approximately 8 feet apart. Video sequences were manually annotated to determine the ground truth head nod labels. A total of 587 head nods occured over all video sequences.

5.2 Multimodal Features

This section describes the different multimodal features used to create our five experts.

PROSODY Prosody refers to the rhythm, pitch and intonation of speech. Several studies have demonstrated that listener feedback is correlated with a speaker's prosody (Nishimura et al., 2007; Ward and Tsukahara, 2000; Cathcart et al., 2003). For example, Ward and Tsukahara (2000) show that short listener backchannels (listener utterances like "ok" or "uh-huh" given during a speaker's utterance) are associated with a lowering of pitch over some interval. Listener feedback often follows speaker pauses or filled pauses such as "um" (see (Cathcart et al., 2003)). Using openSMILE (Eyben et al., 2009) toolbox, we extract the following prosodic features, including standard linguistic annotations and the prosodic features suggested by Ward and Tsukhara: downslopes in pitch continuing for at least 40ms, regions of pitch lower than the 26th percentile continuing for at least 110ms (i.e., lowness), drop or rise in energy of speech (i.e., energy edge), Fast drop or rise in energy of speech (i.e., energy fast edge), vowel volume (i.e., vowels are usually spoken softer) and Pause in speech (i.e., no speech).

VISUAL GESTURES Gestures performed by the speaker are often correlated with listener feedback (Burgoon et al., 1995). Eye gaze, in particular, has often been implicated as eliciting listener feedback. Thus, we manually annotate the following contextual feature: speaker looking at the listener.

LEXICAL Some studies have suggested an association between lexical features and listener feedback (Cathcart et al., 2003). Using the transcriptions, we included all individual words (i.e., unigrams) spoken by the speaker during the interactions.

SYNTACTIC STRUCTURE Finally, we attempt to capture syntactic information that may provide relevant cues by extracting four types of features from a syntactic dependency structure corresponding to the utterance. The syntactic structure is produced automatically using a CRF part-of-speech (POS) tagger and a data-driven left-to-right shift-reduce dependency parser (Sagae and Tsujii, 2007), both trained on POS tags and dependency trees extracted from the Switchboard section of the Penn Treebank (Marcus et al., 1994), converted to dependency trees using the Penn2Malt tool[1]. The four syntactic features are:

- Part-of-speech tags for each word (e.g. noun, verb, etc.), taken from the output of the POS tagger
- Grammatical function for each word (e.g. subject, object, etc.), taken directly from the dependency labels produced by the parser
- Part-of-speech of the syntactic head of each word, taken from the dependency links produced by the parser
- Distance and direction from each word to its syntactic head, computed from the dependency links produced by the parser

[1] http://w3.msi.vxu.se/ nivre/research/Penn2Malt.html

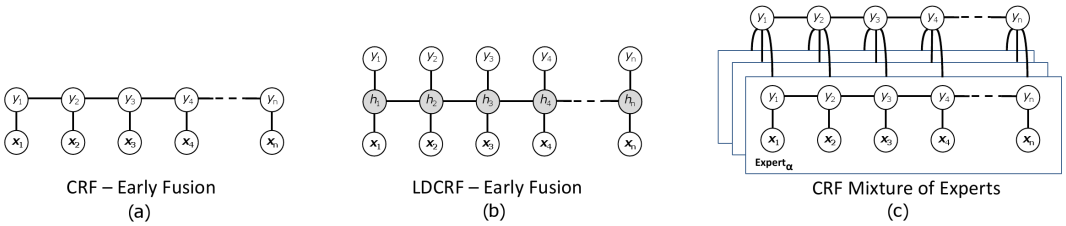

y_1 y_2 y_3 y_4 y_n
h_1 h_2 h_3 h_4 h_n
x_1 x_2 x_3 x_4 x_n
Expert$_\alpha$
CRF – Early Fusion
(a)
LDCRF – Early Fusion
(b)
CRF Mixture of Experts
(c)

Figure 3: Baseline Models: **a)** Conditional Random Fields (CRF), **b)** Latent Dynamic Conditional Random Fields(LDCRF), **c)** CRF Mixture of Experts (no latent variable)

Although our current method for extracting these features requires that the entire utterance be available for processing, this provides us with a first step towards integrating information about syntactic structure in multimodal prediction models. Many of these features could in principle be computed incrementally with only a slight degradation in accuracy, with the exception of features that require dependency links where a word's syntactic head is to the right of the word itself. We leave an investigation that examines only syntactic features that can be produced incrementally in real time as future work.

5.3 Baseline Models

INDIVIDUAL EXPERTS Our first baseline model consists of a set of CRF chain models, each trained with different set of multimodel features (as described in the previous section). In other words, only visual, prosodic, lexical or syntactic features are used to train a single CRF expert. In one CRF chain model, each gesture class corresponds to a state label. (See Figure 3a).

MULTIMODAL CLASSIFIERS (EARLY FUSION) Our second baseline consists of two models: CRF and LDCRF (Morency et al., 2007). To train these models, we concatenate all multimodal features (lexical, syntactic, prosodic and visual) in one input vector. Graphical representation of these baseline models are given in Figure 3.

CRF MIXTURE OF EXPERTS To show the importance of latent variable in our LMDE model, we trained a CRF-based mixture of discriminative experts. This model is similar to the Logarithmic Opinion Pool (LOP) CRF suggested by Smith et al. (2005). The training is performed in two steps. A graphical representation of a CRF Mixture of experts is given in the last graph of Figure 3.

5.4 Methodology

We performed held-out testing by randomly selecting a subset of 11 interactions (out of 47) for the test set. The training set contains the remaining 36 dyadic interactions. All models in this paper were evaluated with the same training and test sets. Validation of all model parameters (regularization term and number of hidden states) was performed using a 3-fold cross-validation strategy on the training set. The regularization term was validated with values $10^k, k = -1..3$. Three different number of hidden states were tested for the LMDE models: 2, 3 and 4.

The performance is measured by using the F-measure. This is the weighted harmonic mean of precision and recall. Precision is the probability that predicted backchannels correspond to actual listener behavior. Recall is the probability that a backchannel produced by a listener in our test set was predicted by the model. We use the same weight for both precision and recall, so-called F_1. During validation we find all the peaks (i.e., local maxima) from the marginal probabilities. These backchannel hypotheses are filtered using the optimal threshold from the validation set. A backchannel (i.e., head nod) is predicted correctly if a peak happens during an actual listener backchannel with high enough probability. The same evaluation measurement is applied to all models.

The training of all CRFs and LDCRFs were done using the hCRF library[2]. The LMDE model was implemented in Matlab[3] based on the hCRF

[2] http://sourceforge.net/projects/hrcf/

[3] The source code is available at: http://projects.ict.usc.edu/multicomp/.

Table 1: Comparison of individual experts with our Latent Mixture of Discriminative Experts (LMDE).

Expert	Precision	Recall	f1
Lexical	0.1647	0.3305	0.2198
Prosody	0.1396	0.9112	0.2421
Syntactic	0.1833	0.4663	0.2632
POS	0.1935	0.4514	0.2709
Eye Gaze	0.1573	0.1741	0.1653
LMDE	0.2295	0.5677	**0.3268**

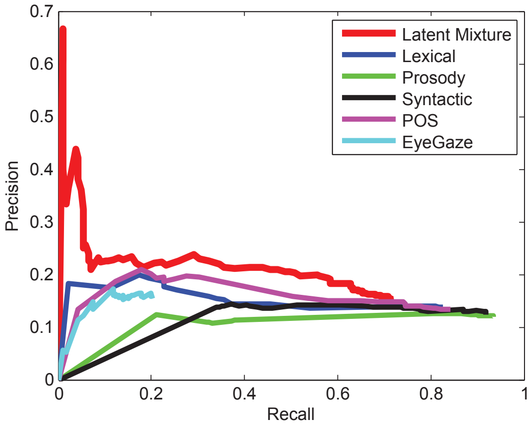


Figure 4: Comparison of individual experts with our LMDE model.

library.

6 Results and Discussion

In this section we present the results of our empirical evaluation designed to test the three main characteristics of the LMDE model: (1) integration of multiple sources of information, (2) late fusion approach and (3) latent variable which models the hidden dynamic between experts. We also present an analysis of the output probabilities from the LMDE model and individual experts.

Individual Experts We trained one individual expert for each feature types: visual, prosodic, lexical and syntactic features (both part-of speech and syntactic structure). Precision, recall and F_1 values for each individual expert and our LMDE model are shown in Table 1 and Figure 4.

Pairwise two-tailed t-test comparison between our LMDE model and individual experts shows a significant difference for Lexical, Prosody, Syntactic and Eye gaze, with respective p-values of 0.0037, 0.0379, 0.0400 and 0.0233. Even though some experts may not perform well individually (e.g., eye gaze), they can bring important information once merged with others. Table 1 shows that our LMDE model was able to take advantage of the complementary information from each expert.

Table 2: Comparison of our Latent Mixture of Discriminative Experts (LMDE) with two early fusion technique (CRF vs LDCRF) and the CRF Mixture of Experts (Smith et al., 2005).

model	Precision	Recall	f1
LMDE	0.2295	0.5677	**0.3268**
Early CRF	0.13958	0.9245	0.2425
Early LDCRF	0.1826	0.2484	0.2105
Mixture CRF	0.1502	0.2712	0.1934

Late Fusion We compare our approach with two early fusion models: CRF and Latent-dynamic CRF (see Figure 3). Table 2 summarizes the results. The CRF model learns direct weights between input features and the gesture labels. The LDCRF is able to model more complex dynamics between input features with the latent variable. We can see that our LMDE model outperforms both early fusion approaches because of its late fusion approach. Pairwise two-tailed t-test analysis gives p-values of 0.0481 and 0.0748, for CRF and LDCRF respectively.

Latent Variable The CRF Mixture of Experts (2005) directly merges the expert outputs while our model uses a latent variable to model the hidden dynamic between experts (see Figure 3). Table 2 summarizes the results. Pairwise two-tailed t-test comparison between these two models shows a significant difference with a p-value of 0.0062. This result is important since it shows that our LMDE model does learn the hidden interaction between experts.

Model analysis To understand the multimodal integration which happens at the latent variable level in our LMDE model, Figure 5 shows the output probabilities for all five individual experts as well as our model. The strength of the latent variable is to enable different weigting

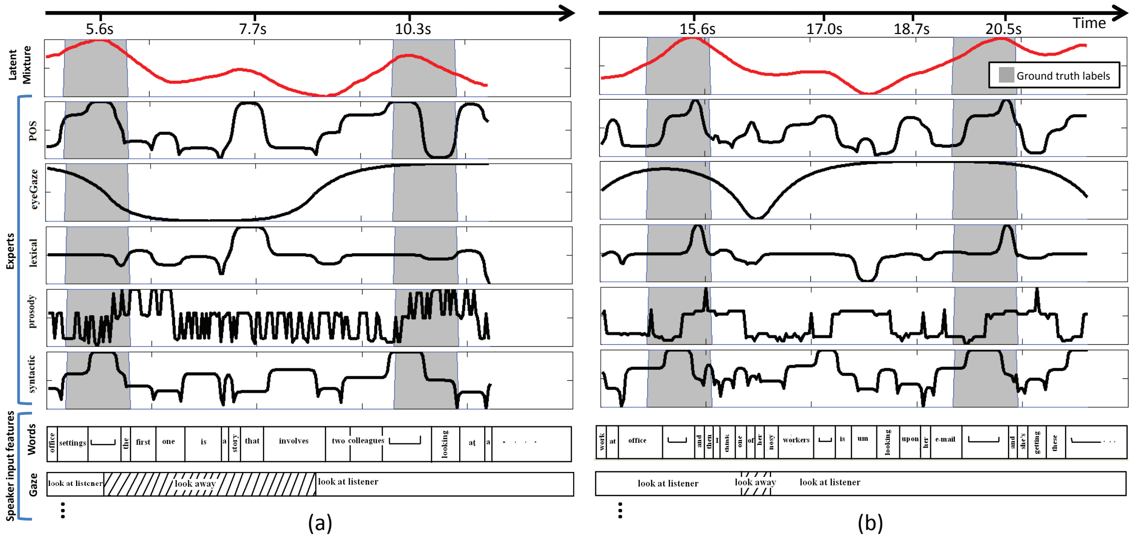


Figure 5: Output probabilities from LMDE and individual experts for two different sub-sequences. The gray areas in the graph corresponds to ground truth backchannel feedbacks of the listener.

of the experts at different point in time.

By analyzing the sequence (a), we observe that both the POS and Syntactic experts learned that when no words are present (i.e., pause) there is a high likelihood of backchennel feedback from the listener (shown at 5.6s and 10.3s). These two experts are highly weighted (by one of the hidden state) during this part of the sequence. Also, both the Lexical and POS experts learned that the word "'that"' (and its part-of-speech) are important but since the speaker is not looking at the listener when saying it, the output from LMDE model is low (see Figure 5, Sequence (a), 7.7s).

By analyzing sequence (b), we see that the Lexical and POS experts learned the importance of the "'and"' at 15.6s and 20.5s. More importantly, we can see at 17.0s and 18.7s that the influence of the POS and Syntactic experts have been reduced in the LMDE output probability. This difference of weighting shows that a different hidden state is active during Sequence (b).

7 Conclusion

In this paper we introduced a new model called Latent Mixture of Discriminative Experts (LMDE) for learning predictive models of human communication behaviors. Many of the interactions between speech and gesture happen at the sub-gesture or sub-word level. LMDE learns automatically the temporal relationship between different modalities. Since, we train separate experts for each modality, LMDE is capable of improving the prediction performance even with limited datasets.

We evaluated our model on the task of nonverbal feedback prediction (e.g., head nod). Our experiments confirm the importance of combining the four types of multimodal features: lexical, syntactic structure, eye gaze, and prosody. LMDE is a generic model that can be applied to a wide range of problems. As future work, we are planning to test our model on dialog act classification and multimodal behavior recognition tasks.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 0917321 and the U.S. Army Research, Development, and Engineering Command (RDECOM). The content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

Anderson, H., M. Bader, E.G. Bard, G. Doherty, S. Garrod, S. Isard, J. Kowtko, J. McAllister, J. Miller, C. Sotillo,

H. Thompson, and R. Weinert. 1991. The mcrc map task corpus. *Language and Speech*, 34(4):351–366.

Bavelas, J.B., L. Coates, and T. Johnson. 2000. Listeners as co-narrators. *JPSP*, 79(6):941–952.

Blunsom, P., T. Cohn, and M. Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *ACL: HLT*, pages 200–208.

Burgoon, Judee K., Lesa A. Stern, and Leesa Dillman. 1995. *Interpersonal adaptation: Dyadic interaction patterns*. Cambridge University Press, Cambridge.

Cassell, J. and M. Stone. 1999. Living hand to mouth: Psychological theories about speech and gesture in interactive dialogue systems. In *AAAI*.

Cathcart, N., Jean Carletta, and Ewan Klein. 2003. A shallow model of backchannel continuers in spoken dialogue. In *EACL*, pages 51–58.

Eisenstein, J., R. Barzilay, and R. Davis. 2008. Gestural cohesion for topic segmentation. In *ACL: HLT*, pages 852–860.

Eisentein, J. and R. Davis. 2007. Conditional modality fusion for coreference. In *ACL*, pages 352–359.

Eyben, Florian, Martin Wöllmer, and Björn Schuller. 2009. openEAR - Introducing the Munich Open-Source Emotion and Affect Recognition Toolkit. In *ACII*, pages 576–581.

Frampton, M., J. Huang, T. Bui, and S. Peters. 2009. Real-time decision detection in multi-party dialogue. In *EMNLP*, pages 1133–1141.

Fuchs, D. 1987. Examiner familiarity effects on test performance: implications for training and practice. *Topics in Early Childhood Special Education*, 7:90–104.

Fujie, Shinya, Yasuhi Ejiri, Kei Nakajima, Yosuke Matsusaka, and Tetsunori Kobayashi. 2004. A conversation robot using head gesture recognition as para-linguistic information. In *RO-MAN*, pages 159–164.

Goldberg, S.B. 2005. The secrets of successful mediators. *Negotiation Journal*, 21(3):365–376.

Gravano, A., S. Benus, H. Chavez, J. Hirschberg, and L. Wilcox. 2007. On the role of context and prosody in the interpretation and 'okay'. In *ACL*, pages 800–807.

Heylen, D. and R. op den Akker. 2007. Computing backchannel distributions in multi-party conversations. In *ACL:EmbodiedNLP*, pages 17–24.

Johnston, M. 1998. Multimodal language processing. In *ICSLP*.

Jovanovic, N., R. op den Akker, and A. Nijholt. 2006. Adressee identification in face-to-face meetings. In *EACL*.

Jurafsky, D., E. Shriberg, B. Fox, and T. Curl. 1998. Lexical, prosodic and syntactic cures for dialog acts. In *Workshop on Discourse Relations*, pages 114–120.

Kendon, A. 2004. *Gesture: Visible Action as Utterance*. Cambridge University Press.

Kumar, S. and M. Herbert. 2003. Discriminative random fields: A framework for contextual interaction in classification. In *ICCV*.

Lafferty, J., A. McCallum, and F. Pereira. 2001. Conditional random fields: probabilistic models for segmenting and labelling sequence data. In *ICML*.

Maatman, M., J. Gratch, and S. Marsella. 2005. Natural behavior of a listening agent. In *IVA*.

Marcus, Mitchell, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *ACL:HLT*, pages 114–119.

McNeill, D. 1992. *Hand and Mind: What Gestures Reveal about Thought*. Univ. Chicago Press.

Moore, P.-Y. Hsueh J. 2007. What decisions have you made: Automatic decision detection in conversational speech. In *NAACL-HLT*, pages 25–32.

Morency, Louis-Philippe, Ariadna Quattoni, and Trevor Darrell. 2007. Latent-dynamic discriminative models for continuous gesture recognition. In *CVPR*.

Murray, G. and G. Carenini. 2009. Predicting subjectivity in multimodal conversations. In *EMNLP*, pages 1348–1357.

Nakano, Reinstein, Stocky, and Justine Cassell. 2003. Towards a model of face-to-face grounding. In *ACL*.

Nakano, Y., K. Murata, M. Enomoto, Y. Arimoto, Y. Asa, and H. Sagawa. 2007. Predicting evidence of understanding by monitoring user's task manipulation in multimodal conversations. In *ACL*, pages 121–124.

Nishimura, Ryota, Norihide Kitaoka, and Seiichi Nakagawa. 2007. A spoken dialog system for chat-like conversations considering response timing. *LNCS*, 4629:599–606.

Oviatt, S. 1999. Ten myths of multimodal interaction. *Communications of the ACM*.

Quek, F. 2003. The catchment feature model for multimodal language analysis. In *ICCV*.

Sagae, Kenji and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *ACL*, pages 1044–1050.

Smith, A., T. Cohn, and M. Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *ACL*, pages 18–25.

Ward, N. and W. Tsukahara. 2000. Prosodic features which cue back-channel responses in english and japanese. *Journal of Pragmatics*, 23:1177–1207.

Text Summarization of Turkish Texts using Latent Semantic Analysis

Makbule Gulcin Ozsoy
Dept. of Computer Eng.
Middle East Tech. Univ.
e1395383@ceng.metu.edu.tr

Ilyas Cicekli
Dept. of Computer Eng.
Bilkent University
ilyas@cs.bilkent.edu.tr

Ferda Nur Alpaslan
Dept. of Computer Eng.
Middle East Tech. Univ.
alpaslan@ceng.metu.edu.tr

Abstract

Text summarization solves the problem of extracting important information from huge amount of text data. There are various methods in the literature that aim to find out well-formed summaries. One of the most commonly used methods is the Latent Semantic Analysis (LSA). In this paper, different LSA based summarization algorithms are explained and two new LSA based summarization algorithms are proposed. The algorithms are evaluated on Turkish documents, and their performances are compared using their ROUGE-L scores. One of our algorithms produces the best scores.

1 Introduction

The exponential growth in text documents brings the problem of finding out whether a text document meets the needs of a user or not. In order to solve this problem, text summarization systems which extract brief information from a given text are created. By just looking at the summary of a document, a user can decide whether the document is of interest to him/her without looking at the whole document.

The aim of a text summarization system is to generate a summary for a given document such that the generated summary contains all necessary information in the text, and it does not include redundant information. Summaries can have different forms (Hahn and Mani, 2000). Extractive summarization systems collect important sentences from the input text in order to generate summaries. Abstractive summarization systems do not collect sentences from the input text, but they try to capture the main concepts in the text, and generate new sentences to represent these main concepts. Abstractive summarization approach is similar to the way that human summarizers follow. Since creating abstractive summaries is a more complex task, most of automatic text summarization systems are extractive summarization systems.

Summarization methods can be categorized according to what they generate and how they generate it (Hovy and Lin, 1999). A summary can be extracted from a single document or from multiple documents. If a summary is generated from a single document, it is known as single-document summarization. On the other hand, if a single summary is generated from multiple documents on the same subject, this is known as multi-document summarization. Summaries are also categorized as generic summaries and query-based summaries. Generic summarization systems generate summaries containing main topics of documents. In query-based summarization, the generated summaries contain the sentences that are related to the given queries.

Extractive summarization systems determine the important sentences of the text in order to put them into the summary. The important sentences of the text are the sentences that represent the main topics of the text. Summarization systems use different approaches to determine the important sentences (Hahn and Mani, 2000; Hovy and Lin, 1999). Some of them look surface clues such as the position of the sentence and the words that are contained in the sentence. Some summarization systems use more semantic oriented analysis such as lexical chains in order to determine the important sentences. Lately, an algebraic method known as Latent Semantic Analysis (LSA) is used in the determination of

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 869–876,
Beijing, August 2010

the important sentences, and successful results are obtained (Gong and Liu, 2001).

In this paper, we present a generic extractive Turkish text summarization system based on LSA. We applied the known text summarization approaches based on LSA in order to extract the summaries of Turkish texts. One of the main contributions of this paper is the introduction of two new summarization methods based on LSA. One of our methods produced much better results than the results of the other known methods.

The rest of the paper is organized as follows. Section 2 presents the related work in summarization. Section 3 explains the LSA approach in detail. Then, the existing algorithms that use different LSA approaches are presented (Gong and Liu, 2001; Steinberger and Jezek 2004; Murray et al., 2005), and two new algorithms are proposed in Section 4. Section 5 presents the evaluation results of these algorithms, and Section 6 presents the concluding remarks.

2 Related Work

Text summarization is an active research area of natural language processing. Its aim is to extract short representative information from input documents. Since the 1950s, various methods are proposed and evaluated. The first studies conducted on text summaries use simple features like terms from keywords/key phrases, terms from user queries, frequency of words, and position of words/sentences (Luhn, 1958).

The use of statistical methods is another approach used for summary extraction. The most well known project that uses statistical approach is the SUMMARIST (Hovy and Lin, 1999). In this project, natural language processing methods are used together with the concept relevance information. The concept relevance information is extracted from dictionaries and WordNet.

Text connectivity is another approach used for summarization. The most well-known algorithm that uses text connectivity is the lexical chains method (Barzilay and Elhadad, 1997; Ercan and Cicekli, 2008). In lexical chains method, WordNet and dictionaries are used to determine semantic relations between words where semantically related words construct lexical chains. Lexical chains are used in the determination of the important sentences of the text.

TextRank (Mihalcea and Tarau, 2004) is a summarization algorithm which is based on graphs, where nodes are sentences and edges represent similarity between sentences. The similarity value is decided by using the overlapping terms. Cluster Lexrank (Qazvinian and Radev, 2008) is another graph-based summarization algorithm, and it tries to find important sentences in a graph in which nodes are sentences and edges are similarities.

In recent years, algebraic methods are used for text summarization. Most well-known algebraic algorithm is Latent Semantic Analysis (LSA) (Landauer et al., 1998). This algorithm finds similarity of sentences and similarity of words using an algebraic method, namely Singular Value Decomposition (SVD). Besides text summarization, the LSA algorithm is also used for document clustering and information filtering.

3 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is an algebraic-statistical method that extracts meaning of words and similarity of sentences using the information about the usage of the words in the context. It keeps information about which words are used in a sentence, while preserving information of common words among sentences. The more common words between sentences mean that those sentences are more semantically related.

LSA method can represent the meaning of words and the meaning of sentences simultaneously. It averages the meaning of words that a sentence contains to find out the meaning of that sentence. It represents the meaning of words by averaging the meaning of sentences that contain this word.

LSA method uses Singular Value Decomposition (SVD) for finding out semantically similar words and sentences. SVD is a method that models relationships among words and sentences. It has the capability of noise reduction, which leads to an improvement in accuracy.

LSA has three main limitations. The first limitation is that it uses only the information in the input text, and it does not use the information of world knowledge. The second limitation is that it does not use the information of word order, syntactic relations, or morphologies. Such information is used for finding out the meaning of words and texts. The third limitation is that the performance of the algorithm decreases with large and inhomogeneous data. The decrease in performance is observed since SVD which is a very complex algorithm is used for finding out the similarities.

All summarization methods based on LSA use three main steps. These steps are as follows:

1. *Input Matrix Creation:* A matrix which represents the input text is created. The columns of the matrix represent the sentences of the input text and the rows represent the words. The cells are filled out to represent the importance of words in sentences using different approaches, whose details are described in the rest of this section. The created matrix is sparse.
2. *Singular Value Decomposition (SVD):* Singular value decomposition is a mathematical method which models the relationships among terms and sentences. It decomposes the input matrix into three other matrices as follows:

   $$A = U \sum V^T$$

   where A is the input matrix with dimensions *m x n*, U is an *m x n* matrix which represents the description of the original rows of the input matrix as a vector of extracted concepts, $\sum$ is an *n x n* diagonal matrix containing scaling values sorted in descending order, and V is an *m x n* matrix which represents the description of the original columns of input matrix as a vector of the extracted concepts.
3. *Sentence Selection*: Different algorithms are proposed to select sentences from the input text for summarization using the results of SVD. The details of these algorithms are described in Section 4.

The creation of the input matrix is important for summarization, since it affects the resulting matrices of SVD. There are some ways to reduce the row size of the input matrix, such as eliminating words seen in stop words list, or using the root words only. There are also different approaches to fill out the input matrix cell values, and each of them affects the performance of the summarization system differently. These approaches are as follows:

1. *Number of Occurrence:* The cell is filled with the frequency of the word in the sentence.
2. *Binary Representation of Number of Occurrence:* If the word is seen in the sentence, the cell is filled with 1; otherwise it is filled with 0.
3. *TF-IDF (Term Frequency–Inverse Document Frequency):* The cell is filled with TF-IDF value of the word. This method evaluates the importance of words in a sentence. The importance of a word is high if it is frequent in the sentence, but less frequent in the document. TF-IDF is equal to TF*IDF, and TF and IDF are computed as follows:

   $$tf\,(i,j) = n(i,j) \;/\; \sum_k n(k,j)$$

   where $n(i,j)$ is the number of occurrences of the considered word *i* in sentence *j*, and $\sum_k n(k,j)$ is the sum of number of occurrences of all words in sentence *j*.

   $$idf\,(i) = log(\,|D| \,/\, d_i)$$

   where $|D|$ is the total number of sentences in the input text, and d_i is the number of sentences where the word *i* appears
4. *Log Entropy*: The cell is filled with log-entropy value of the word, and it is computed as follows.

   $$sum = \sum_j p(i,j)\ log_2(p(i,j))$$
   $$global(i) = 1 + (sum \,/\, log_2(n))$$
   $$local(i,j)= log_2(1 + f(i,j))$$
   $$log\text{-}entropy = global*local$$

   where $p(i,j)$ is the probability of word *i* that is appeared in sentence *j*, $f(i,j)$ is the number of times word *i* appeared in sentence *j*, and *n* is the number of sentences in the document.
5. *Root Type:* If the root type of the word is noun, the related cell is filled with the frequency of the word in the sentence; otherwise the cell is filled with 0.
6. *Modified TF-IDF:* First the matrix is filled with TF-IDF values. Then, the average TF-IDF values in each row are calculated. If the value in the cell is less than or equal to the average value, the cell value is set to 0. This is our new approach which is proposed to eliminate the noise from the input matrix.

4 Text Summarization

The algorithms in the literature that use LSA for text summarization perform the first two steps of LSA algorithm in the same way. They differ in the way they fill out the input matrix cells.

4.1 Sentence Selection Algorithms in Literature

4.1.1. Gong & Liu (Gong and Liu, 2001)

After performing the first two steps of the LSA algorithm, Gong & Liu summarization algorithm uses V^T matrix for sentence selection. The columns of V^T matrix represent the sentences of the input matrix and the rows of it represent the concepts that are obtained from SVD method. The most important concept in the text is placed in the first row, and the row order indicates the importance of concepts. Cells of this matrix give information about how much the sentence is related to the given concept. A higher cell value means the sentence is more related to the concept.

In Gong & Liu summarization algorithm, the first concept is chosen, and then the sentence most related to this concept is chosen as a part of the resulting summary. Then the second concept is chosen, and the same step is executed. This repetition of choosing a concept and the sentence most related to that concept is continued until a predefined number of sentences are extracted as a part of the summary. In Figure 1, an example V^T matrix is given. First, the concept *con0* is chosen, and then the sentence *sent1* is chosen, since it has the highest cell value in that row.

There are some disadvantages of this algorithm, which are defined by Steinberger and Jezek (2004). First, the reduced dimension size has to be the same as the summary length. This approach may lead to the extraction of sentences from less significant concepts. Second, there exist some sentences that are related to the chosen concept somehow, but do not have the highest cell value in the row of that concept. These kinds of sentences cannot be included in the resulting summary by this algorithm. Third, all chosen concepts are thought to be in the same importance level, but some of those concepts may not be so important in the input text.

	sent0	sent1	sent2	sent3	sent4
con0	0,557	0,691	0,241	0,110	0,432
con1	0,345	0,674	0,742	0,212	0,567
con2	0,732	0,232	0,435	0,157	0,246
con3	0,628	0,836	0,783	0,265	0,343

Figure 1. Gong & Liu approach: From each row of V^T matrix which represents a concept, the sentence with the highest score is selected. This is repeated until a predefined number of sentences are collected.

4.1.2. Steinberger & Jezek (Steinberger and Jezek 2004)

As in the Gong & Liu summarization algorithm, the first two steps of LSA algorithm are executed before selecting sentences to be a part of the resulting summary. For sentence selection, both V and Σ matrixes are used.

The sentence selection step of this algorithm starts with the calculation of the length of each sentence vector which is represented by a row in V matrix. In order to find the length of a sentence vector, only concepts whose indexes are less than or equal to the number of dimension in the new space is used. The dimension of a new space is given as a parameter to the algorithm. The concepts which are highly related to the text are given more importance by using the values in Σ matrix as a multiplication parameter. If the dimension of the new space is *n*, the length of the sentence *i* is calculated as follows:

$$length_i = \sqrt{\sum_{j=1}^{n} V_{ij} * \Sigma_{jj}}$$

After the calculation of sentence lengths, the longest sentences are chosen as a part of the resulting summary. In Figure 2, an example V matrix is given, and the dimension of the new space is assumed to be 3. The lengths of the sentences are calculated using the first three concepts. Since the sentence *sent2* has the highest length, it is extracted first as a part of the summary.

The aim of this algorithm is to get rid of the disadvantages of Gong & Liu summarization algorithm, by choosing sentences which are related to all important concepts and at the same time choosing more than one sentence from an important topic.

	con0	con1	con2	con3	length
sent0	0,846	0,334	0,231	0,210	0,432
sent1	0,455	0,235	0,432	0,342	0,543
sent2	0,562	0,632	0,735	0,857	0,723
sent3	0,378	0,186	0,248	0,545	0,235

Figure 2. Steinberger & Jezek approach: For each row of V matrix, the lengths of sentences using n concepts are calculated. The value n is given as an input parameter. ∑ matrix values are also used as importance parameters in the length calculations.

	sent0	sent1	sent2	sent3	sent4
con0	0,557	0,691	0,241	0,110	0,432
con1	0,345	0,674	0,742	0,212	0,567
con2	0,732	0,232	0,435	0,157	0,246
con3	0,628	0,836	0,783	0,265	0,343

Figure 3. Murray & Renals & Carletta approach: From each row of V^T matrix, concepts, one or more sentences with the higher scores are selected. The number of sentences to be selected is decided by using ∑ matrix.

4.1.3. Murray & Renals & Carletta (Murray et al., 2005)

The first two steps of the LSA algorithm are executed, as in the previous algorithms before the construction of the summary. V^T and ∑ matrices are used for sentence selection.

In this approach, one or more sentences are collected from the topmost concepts in V^T matrix. The number of sentences to be selected depends on the values in the ∑ matrix. The number of sentences to be collected for each topic is determined by getting the percentage of the related singular value over the sum of all singular values, which are represented in the ∑ matrix. In Figure 3, an example V^T matrix is given. Let's choose two sentences from *con0,* and one sentence from *con1*. Thus, the sentences *sent1* and *sent0* are selected from *con0,* and *sent2* is selected from *con1* as a part of the summary.

This approach tries to solve the problems of Gong & Liu's approach. The reduced dimension has not to be same as the number of sentences in the resulting summary. Also, more than one sentence can be chosen even they do not have the highest cell value in the row of the related concept.

4.2 Proposed Sentence Selection Algorithms

The analysis of input documents indicates that some sentences, especially the ones in the introduction and conclusion parts of the documents, belong to more than one main topic. In order to observe whether these sentences are important or they cause noise in matrices of LSA, we propose a new method, named as *Cross*.

Another concern about matrices in LSA is that the concepts that are found after the SVD step may represent main topics or subtopics. So, it is important to determine whether the found concepts are main topics or subtopics. This causes the ambiguity that whether these concepts are subtopics of another main topic, or all the concepts are main topics of the input document. We propose another new method, named as *Topic*, in order to distinguish main topics from subtopics and make sentence selections from main topics.

4.2.1. Cross Method

In this approach, the first two steps of LSA are executed in the same way as the other approaches. As in the Steinberger and Jezek approach, the V^T matrix is used for sentence selection. The proposed approach, however, preprocesses the V^T matrix before selecting the sentences. First, an average sentence score is calculated for each concept which is represented by a row of V^T matrix. If the value of a cell in that row is less than the calculated average score of that row, the score in the cell is set to zero. The main idea is that there can be sentences such that they are not the core sentences representing the topic, but they are related to the topic in some way. The preprocessing step removes the overall effect of such sentences.

After preprocessing, the steps of Steinberger and Jezek approach are followed with a modification. In our Cross approach, first the cell values are multiplied with the values in the ∑ matrix, and the total lengths of sentence vectors, which are represented by the columns of the V^T matrix, are calculated. Then, the longest sentence vectors are collected as a part of the resulting summary.

In Figure 4, an example V^T matrix is given. First, the average scores of all concepts are calculated, and the cells whose values are less than the average value of their row are set to zero.

The boldface numbers are below row averages in Figure 4, and they are set to zero before the calculation of the length scores of sentences. Then, the length score of each sentence is calculated by adding up the concept scores of sentences in the updated matrix. In the end, the sentence *sent1* is chosen for the summary as the first sentence, since it has the highest length score.

	sent0	sent1	sent2	sent3	average
con0	0,557	0,691	**0,241**	**0,110**	0,399
con1	**0,345**	0,674	0,742	**0,212**	0,493
con2	0,732	**0,232**	0,435	**0,157**	0,389
con3	**0,628**	**0,436**	0,783	0,865	0,678
con4	0,557	0,691	**0,241**	0,710	0,549
length	1,846	2,056	1,960	1,575	

Figure 4. Cross approach: For each row of V^T matrix, the cell values are set to zero if they are less than the row average. Then, the cell values are multiplied with the values in the $\sum$ matrix, and the lengths of sentence vectors are found, by summing up all concept values in columns of V^T matrix, which represent the sentences.

4.2.2. Topic Method

The first two steps of LSA algorithm are executed as in the other approaches. For sentence selection, the V^T matrix is used. In the proposed approach, the main idea is to decide whether the concepts that are extracted from the matrix V^T are really main topics of the input text, or they are subtopics. After deciding the main topics which may be a group of subtopics, the sentences are collected as a part of the summary from the main topics.

In the proposed algorithm, a preprocessing step is executed, as in the Cross approach. First, for each concept which is represented by a row of V^T matrix, the average sentence score is calculated and the values less than this score are set to zero. So, a sentence that is not highly related to a concept is removed from the concept in the V^T matrix. Then, the main topics are found. In order to find out the main topics, a *concept x concept* matrix is created by summing up the cell values that are common between the concepts. After this step, the strength values of the concepts are calculated. For this calculation, each concept is thought as a node, and the similarity values in *concept x concept* matrix are considered as edge scores. The strength value of each concept is calculated by summing up the values in each row in *concept x concept* matrix. The topics with the highest strength values are chosen as the main topic of the input text.

	sent0	sent1	sent2	sent3	average
con0	0,557	0,691	**0,241**	**0,110**	0,399
con1	**0,345**	0,674	0,742	**0,212**	0,493
con2	0,732	**0,232**	0,435	**0,157**	0,389
con3	**0,628**	**0,436**	0,783	0,865	0,678
con4	0,557	0,691	**0,241**	0,710	0,549

⇩

	con0	con1	con2	con3	con4	strength
con0	1,248	1,365	1,289	0	2,496	6,398
con1	1,365	1,416	1,177	1,525	1,365	6,848
con2	1,289	1,177	0,732	1,218	1,289	5,705
con3	0	1,525	1,218	1,648	1,575	5,966
con4	2,496	1,365	1,289	1,575	1,958	8,683

⇩

	sent0	sent1	sent2	sent3
con0	0,557	0.691	0	0
con1	0	0,674	0,742	0
con2	0,732	0	0,435	0
con3	0	0	0,783	0,865
con4	0,557	0,691	0	0,710

Figure 5. Topic approach: From each row of V^T matrix, concepts, the values are set to zero if they are less than the row average. Then *concept x concept* similarity matrix is created, and the strength values of concepts are calculated, which show how strong the concepts are related to the other concepts. Then the concept whose strength value is highest is chosen, and the sentence with the highest score from that concept is collected. The sentence selection s repeated until a predefined number of sentences is collected.

After the above steps, sentence selection is performed in a similar manner to Gong and Liu approach. For each main topic selected, the sentence with the highest score is chosen. This selection is done until predefined numbers of sentences are collected.

In Figure 5, an example V^T matrix is given. First, the average scores of each concept is calculated and shown in the last column of the matrix. The cell values that are less than the row average value (boldface numbers in Figure 5) are set to zero. Then, a *concept x concept* matrix is created by filling a cell with the summation of the cell values that are common between those two concepts. The strength values of the concepts are calculated by summing up the concept values, and the strength values are shown in the last column of the related matrix. A higher strength value indicates that the concept is much more related to the other concepts, and it is one of the main topics of the input text. After finding out the main topic which is the concept *con4* in this example, the sentence with the highest cell value which is sentence *sent3* is chosen as a part of the summary.

5 Evaluation

Two different sets of scientific articles in Turkish are used for the evaluation our summarization approach. The articles are chosen from different areas, such as medicine, sociology, psychology, having fifty articles in each set. The second data set has longer articles than the first data set. The abstracts of these articles, which are human-generated summaries, are used for comparison. The sentences in the abstracts may not match with the sentences in the input text. The statistics about these data sets are given in Table 1.

	DS1	DS2
Number of documents	50	50
Sentences per document	89,7	147,3
Words per document	2302,2	3435
Words per sentence	25,6	23,3

Table 1. Statistics of datasets

Evaluation of summaries is an active research area. Judgment of human evaluators is a common approach for the evaluation, but it is very time consuming and may not be objective. Another approach that is used for summarization evaluation is to use the ROUGE evaluation approach (Lin and Hovy, 2003), which is based on n-gram co-occurrence, longest common subsequence and weighted longest common subsequence between the ideal summary and the extracted summary. Although we obtained all ROUGE results (ROUGE-1, ROUGE-2, ROUGE-3, ROUGE-W and ROUGE-L) in our evaluations, we only report ROUGE-L results in this paper. The discussions that are made depending on our ROUGE-L results are also applicable to other ROUGE results. Different LSA approaches are executed using different matrix creation methods.

	G&L	S&J	MRC	Cross	Topic
frequency	0,236	0,250	0,244	0,302	0,244
binary	0,272	0,275	0,274	0,313	0,274
tf-idf	0,200	0,218	0,213	0,304	0,213
logentropy	0,230	0,250	0,235	0,302	0,235
root type	0,283	0,282	0,289	0,320	0,289
mod. tf-idf	0,195	0,221	0,223	0,290	0,223

Table 2. ROUGE-L scores for the data set DS1

In Table 2, it can be observed that the Cross method has the highest ROUGE scores for all matrix creation techniques. The Topic method has the same results with Murray & Renals & Carletta approach, and it is better than the Gong & Liu approach.

Table 2 indicates that all algorithms give their best results when the input matrix is created using the root type of words. Binary and log-entropy approaches also produced good results. Modified tf-idf approach, which is proposed in this paper, did not work well for this data set. The modified tf-idf approach lacks performance because it removes some of the sentences/words from the input matrix, assuming that they cause noise. The documents in the data set DS1 are shorter documents, and most of words/sentences in shorter documents are important and should be kept.

Table 3 indicates that the best F-score is achieved for all when the log-entropy method is used for matrix creation. Modified tf-idf approach is in the third rank for all algorithms. We can also observe that, creating matrix according to the root types of words did not work well for this data set.

Given the evaluation results it can be said that *Cross* method, which is proposed in this paper, is a promising approach. Also *Cross* approach is not affected from the method of matrix creation. It produces good results when it is compared against an abstractive summary which is created by a human summarizer.

	G&L	S&J	MRC	Cross	Topic
frequency	0,256	0,251	0,259	0,264	0,259
binary	0,191	0,220	0,189	0,274	0,189
tf-idf	0,230	0,235	0,227	0,266	0,227
logentropy	0,267	0,245	0,268	0,267	0,268
root type	0,194	0,222	0,197	0,263	0,197
mod. tf-idf	0,234	0,239	0,232	0,268	0,232

Table 3. ROUGE-L scores for the data set DS2

6 Conclusion

The growth of text based resources brings the problem of getting the information matching needs of user. In order to solve this problem, text summarization methods are proposed and evaluated. The research on summarization started with the extraction of simple features and improved to use different methods, such as lexical chains, statistical approaches, graph based approaches, and algebraic solutions. One of the algebraic-statistical approaches is Latent Semantic Analysis method.

In this study, text summarization methods which use Latent Semantic Analysis are explained. Besides well-known Latent Semantic Analysis approaches of Gong & Liu, Steinberger & Jezek and Murray & Renals & Carletta, two new approaches, namely Cross and Topic, are proposed.

Two approaches explained in this paper are evaluated using two different datasets that are in Turkish. The comparison of these approaches is done using the ROUGE-L F-measure score. The results show that the Cross method is better than all other approaches. Another important result of this approach is that it is not affected by different input matrix creation methods.

In future work, the proposed approaches will be improved and evaluated in English texts as well. Also, ideas that are used in other methods, such as graph based approaches, will be used together with the proposed approaches to improve the performance of summarization.

Acknowledgments

This work is partially supported by The Scientific and Technical Council of Turkey Grant “TUBITAK EEEAG-107E151”.

References

Barzilay, R. and Elhadad, M. 1997. Using Lexical Chains for Text Summarization. *Proceedings of the ACL/EACL'97 Workshop on Intelligent Scalable Text Summarization*, pages 10-17.

Ercan G. and Cicekli, I. 2008. Lexical Cohesion based Topic Modeling for Summarization. *Proceedings of 9th Int. Conf. Intelligent Text Processing and Computational Linguistics (CICLing-2008),* pages 582-592.

Gong, Y. and Liu, X. 2001. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. *Proceedings of SIGIR'01*.

Hahn, U. and Mani, I. 2000. The challenges of automatic summarization. *Computer,* **33**, 29–36.

Hovy, E. and Lin, C-Y. 1999. Automated Text Summarization in SUMMARIST. I. Mani and M.T. Maybury (eds.), *Advances in Automatic Text Summarization*, The MIT Press, pages 81-94.

Landauer, T.K., Foltz, P.W. and Laham, D. 1998. An Introduction to Latent Semantic Analysis. *Discourse Processes*, 25, 259-284.

Lin, C.Y. and Hovy, E.. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. *Proceedings of 2003 Conf. North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL-2003),* pages 71-78.

Luhn, H.P. 1958. The Automatic Creation of Literature Abstracts. *IBM Journal of Research Development* 2(2), 159-165.

Mihalcea, R. and Tarau, P. 2004. Text-rank - bringing order into texts. *Proceeding of the Conference on Empirical Methods in Natural Language Processing*.

Murray, G., Renals, S. and Carletta, J. 2005. Extractive summarization of meeting recordings. *Proceedings of the 9th European Conference on Speech Communication and Technology*.

Qazvinian, V. and Radev, D.R. 2008. Scientific paper summarization using citation summary networks. *Proceedings of COLING2008*, Manchester, UK, pages 689-696.

Steinberger, J. and Jezek, K. 2004. Using Latent Semantic Analysis in Text Summarization and Summary Evaluation. *Proceedings of ISIM '04*, pages 93-100.

Knowing What to Believe
(when you already know something)

Jeff Pasternack **Dan Roth**
University of Illinois, Urbana-Champaign
{jpaster2, danr}@uiuc.edu

Abstract

Although much work in NLP has focused on simply determining what a document *means*, we also must know whether or not to *believe* it. Fact-finding algorithms attempt to identify the "truth" among competing claims in a corpus, but fail to take advantage of the user's prior knowledge and presume that truth itself is universal and objective rather than subjective. We introduce a framework for incorporating prior knowledge into *any* fact-finding algorithm, expressing both general "common-sense" reasoning and specific facts already known to the user as first-order logic and translating this into a tractable linear program. As our results show, this approach scales well to even large problems, both reducing error and allowing the system to determine truth respective to the user rather than the majority. Additionally, we introduce three new fact-finding algorithms capable of outperforming existing fact-finders in many of our experiments.

1 Introduction

Although establishing the trustworthiness of the information presented to us has always been a challenge, the advent of the Information Age and the Internet has made it more critical. Blogs, wikis, message boards and other collaborative media have eliminated the high entry barrier–and, with it, the enforced journalistic standards–of older, established media such as newspapers and television, and even these sometimes loosen their fact-checking in the face of increased competitive pressure. Consequently, we find that corpora derived from these sources now offer far more numerous views of far more questionable veracity. If one author claims Mumbai is the largest city in the world, and another claims it is Seoul, who do we believe? One or both authors could be intentionally lying, honestly mistaken or, alternatively, of different viewpoints of what constitutes a "city" (the city proper? The metropolitan area?) Truth is not objective: there may be many valid definitions of "city", but we should believe the claim that accords with our *user's* viewpoint. Note that the user may be another computational system rather than a human (e.g. building a knowledge base of city sizes for question answering), and often neither the user's nor the information source's perspective will be explicit (e.g. an author will not fully elaborate "the largest city by metropolitan area bounded by...") but will instead be implied (e.g. a user's statement that "I already know the population of city A is X, city B is Y..." implies that his definition of a city accords with these figures).

The most basic approach is to take a vote: if multiple claims are mutually exclusive of each other, select the one asserted by the most sources. In our experiments, sources will be the authors of the document containing the claim, but other sources could be publishers/websites (when no authorship is given), an algorithm that outputs claims, etc. Although sometimes competitive, we found voting to be generally lackluster. A class of algorithms called fact-finders are often a dramatic improvement, but are incapable of taking advantage of the user's prior knowledge. Our framework translates prior knowledge (expressed as first-order logic) into a linear program that constrains the claim beliefs produced by a fact-finder, ensuring that our belief state is consistent with both common sense ("cities usually grow") and known facts ("Los Angeles is more populous than Wichita"). While in the past first-order logic has been translated to NP-hard integer linear programs, we use polynomial-time-solvable linear programs, al-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 877–885,
Beijing, August 2010

lowing us to readily scale to large problems with extensive prior knowledge, as demonstrated by our experiments.

We next discuss related work, followed by a more in-depth description of the fact-finding algorithms used in our experiments, including three novel, high-performing algorithms: Average·Log, Investment, and PooledInvestment. We then present the framework's mechanics and the translation of first-order logic into a linear program. Finally, we present our experimental setup and results over three domains chosen to illustrate different aspects of the framework, demonstrating that both our new fact-finders and our framework offer performance improvements over the current state of the art.

2 Related Work

The broader field of trust can be split into three areas of interest[1]: theoretical, reputation-based, and information-based.

2.1 Theoretical

Marsh (1994) observes that trust can be global (e.g. eBay's feedback scores), personal (each person has their own trust values), or situational (personal and specific to a context). Fact-finding algorithms are based on global trust, while our framework establishes personal trust by exploiting the user's individual prior knowledge.

Probabilistic logics have been explored as an alternate method of reasoning about trust. Manchala (1998) utilizes fuzzy logic (Novak et al., 1999), an extension of propositional logic permitting [0,1] belief over propositions. Yu and Singh (2003) employs Dempster-Shafer theory (Shafer, 1976), with belief triples (mass, belief, and plausibility) over *sets* of possibilities to permit the modeling of ignorance, while Josang et al. (2006) uses the related subjective logic (Josang, 1997). While our belief in a claim is decidedly Bayesian (the probability that the claim is true), "unknowns" (discussed later) allow us to reason about ignorance as subjective logic and Dempster-Shafer do, but with less complexity.

2.2 Reputation-based

Reputation-based systems determine an entity's trust or standing among peers via transitive recommendations, as PageRank (Brin and Page, 1998) does among web pages, Advogato (Levien, 2008) does among people, and Eigentrust (Kamvar et al., 2003) does among peers in a network. Some, such as Hubs and Authorities (Kleinberg, 1999), are readily adapted to fact-finding, as demonstrated later.

2.3 Information-Based

Information-based approaches utilize content (rather than peer recommendations) to compute trust, and are often specialized for a particular domain. For example, (Zeng et al., 2006) and Wikitrust (Adler and de Alfaro, 2007) determine trust in a wiki's text passages from sequences of revisions but lack the claim-level granularity and general applicability of fact-finders.

Given a large set of sources making conflicting claims, fact-finders determine "the truth" by iteratively updating their parameters, calculating belief in facts based on the trust in their sources, and the trust in sources based on belief in their facts. TruthFinder (Yin et al., 2008) is a straightforward implementation of this idea. AccuVote (Dong et al., 2009a; Dong et al., 2009b) improves on this by using calculated source dependence (where one source derives its information from another) to give higher credibility to independent sources. (Galland et al., 2010)'s 3-Estimates algorithm incorporates the estimated "hardness" of a fact, such that knowing the answer to an easy question earns less trust than to a hard one. Except for AccuVote (whose model of repeated source-to-source copying is inapplicable to our experimental domains) we experimented over all of these algorithms.

3 Fact-Finding

We have a set of sources S each asserting a set of claims C_s, with $C = \bigcup_{s \in S} C_s$. Each claim $c \in C$ belongs to a *mutual exclusion set* $M_c \subseteq C$, a set of claims (including c) that are mutually exclusive with one another; for example, "John was born in 1960" and "John was born in 1965" are mutually exclusive because a person cannot be born in more than one year. If c is not mutually exclusive

[1] Following the division proposed by Artz and Gil (2007); see also (Sabater and Sierra, 2005) for a survey from a different perspective.

to any other claims, then $M_c = \{c\}$. Assuming there exists exactly one true claim $\overline{c}$ in each mutual exclusion set M, our goal is to predict $\overline{c}$ for each M, with accuracy measured by the number of successful predictions divided by the number of mutual exclusion sets, ignoring trivially correct claims that are the sole members of their mutual exclusion set. To this end, fact-finding algorithms iterate to find the trustworthiness of each source $T^i(s)$ at iteration i in terms of the belief in its claims in the previous iteration $B^{i-1}(C_s)$, and belief in each claim $B^i(c)$ in terms of $T^i(S_c)$, where $S_c = \{s : s \in S, c \in C_s\}$ is the set of all sources asserting c. Note that "trustworthiness" and "belief" as used within a fact-finding algorithm typically do not have meaningful semantics (i.e. they are not $[0, 1]$ Bayesian probabilities). Iteration continues until convergence or some predefined stop criteria.

3.1 Priors

Except for 3-Estimates (where the priors are dictated by the algorithm itself), every fact-finder requires priors for $B^0(C)$. For each fact-finder we chose from $B^0_{voted}(c) = |S_c|/\sum_{d\in M_c}|S_d|$, $B^0_{uniform}(c) = 1/|M_c|$, and $B^0_{fixed}(c) = 0.5$.

3.2 Algorithms

3.2.1 Sums (Hubs and Authorities)

Hubs and Authorities (Kleinberg, 1999) gives each page a hub score and an authority score, where its hub score is the sum of the authority of linked pages and its authority is the sum of the hub scores of pages linking to it. This is adapted to fact-finding by viewing sources as hubs (with 0 authority) and claims as authorities (with 0 hub score):

$$T^i(s) = \sum_{c\in C_s} B^{i-1}(c) \quad B^i(c) = \sum_{s\in S_c} T^i(s)$$

We normalize to prevent $T^i(s)$ and $B^i(c)$ from growing unbounded (dividing by $\max_s T^i(s)$ and $\max_c B^i(c)$, respectively), a technique also used with the Investment and Average·Log algorithms (discussed next); this avoids numerical overflow. B^0_{fixed} priors are used.

3.2.2 Average·Log

Computing $T(s)$ as an average of belief in its claims overestimates the trustworthiness of a source with relatively few claims; certainly a source with 90% accuracy over a hundred examples is more trustworthy than a source with 90% accuracy over ten. However, summing the belief in claims allows a source with 10% accuracy to obtain a high trustworthiness score by simply making many claims. Average·Log attempts a compromise, while still using Sums' B^i update rule and B^0_{fixed} priors.

$$T^i(s) = \log|C_s| \cdot \frac{\sum_{c\in C_s} B^{i-1}(c)}{|C_s|}$$

3.2.3 Investment

In the Investment algorithm, sources "invest" their trustworthiness uniformly among their claims. The belief in each claim then grows according to a non-linear function $\mathcal{G}$, and a source's trustworthiness is calculated as the sum of the beliefs in their claims, weighted by the proportion of trust previously contributed to each (relative to the other investors). Since claims with higher-trust sources get higher belief, these claims become relatively more believed and their sources become more trusted. We used $\mathcal{G}(x) = x^g$ with $g = 1.2$ in our experiments, together with B^0_{voted} priors.

$$T^i(s) = \sum_{c\in C_s} B^{i-1}(c) \cdot \frac{T^{i-1}(s)}{|C_s| \cdot \sum_{r\in S_c} \frac{T^{i-1}(r)}{|C_r|}}$$

$$B^i(c) = \mathcal{G}\left(\sum_{s\in S_c} \frac{T^i(s)}{|C_s|}\right)$$

3.2.4 PooledInvestment

Like Investment, sources uniformly invest their trustworthiness in claims and obtain corresponding returns, so $T^i(s)$ remains the same, but now after the belief in the claims of mutual exclusion set M have grown according to $\mathcal{G}$, they are linearly scaled such that the total belief of the claims in M remains the same as it was before applying $\mathcal{G}(x) = x^g$, with $g = 1.4$ and $B^0_{uniform}$ priors used in our experiments. Given $H^i(c) = \sum_{s\in S_c} \frac{T^i(s)}{|C_s|}$, we have:

$$B^i(c) = H^i(c) \cdot \frac{\mathcal{G}(H^i(c))}{\sum_{d\in M_c} \mathcal{G}(H^i(d))}$$

3.3 TruthFinder

TruthFinder (Yin et al., 2008) is pseudoprobabilistic: the basic version of the algorithm below calculates the “probability” of a claim by assuming that each source’s trustworthiness is the probability of it being correct and then averages claim beliefs to obtain trustworthiness scores. We also used the “full”, more complex TruthFinder, omitted here for brevity. $B^0_{uniform}$ priors are used for both.

$$T^i(s) = \frac{\sum_{c \in C_s} B^{i-1}(c)}{|C_s|}$$
$$B^i(c) = 1 - \prod_{s \in S_c} \left(1 - T^i(s)\right)$$

3.3.1 3-Estimates

3-Estimates (Galland et al., 2010), also omitted for brevity, differs from the other fact-finders by adding a third set of parameters to capture the “difficulty” of a claim, such that correctly asserting a difficult claim confers more trustworthiness than asserting an easy one; knowing the exact population of a city is harder than knowing the population of Mars (presumably 0) and we should not trust a source merely because they provide what is already common knowledge.

4 The Framework

To apply prior knowledge to a fact-finding algorithm, we translate the user’s prior knowledge into a linear program. We then iterate the following until convergence or other stopping criteria:

1. Compute $T^i(s)$ for all $s \in S$
2. Compute $B^i(c)$ for all $c \in C$
3. “Correct” beliefs $B^i(C)$ with the LP

4.1 Propositional Linear Programming

To translate prior knowledge into a linear program, we first propositionalize our first-order formulae into propositional logic (Russell and Norvig, 2003). For example, assume we know that Tom is older than John and a person has exactly one age ($\exists_{x,y} Age(Tom, x) \wedge Age(John, y) \wedge x > y) \wedge (\forall_{x,y,z} Age(x, y) \wedge y \neq z \Rightarrow \neg Age(x, z)$), and our system is considering the following claims: $Age(Tom, 30)$, $Age(Tom, 40)$, $Age(John, 25)$, $Age(John, 35)$. Our propositional clauses (after removing redundancies) are then $Age(Tom, 30) \Rightarrow Age(John, 25) \wedge (Age(Tom, 30) \oplus Age(Tom, 40)) \wedge (Age(John, 25) \oplus Age(John, 35))$.

Each claim c will be represented by a proposition, and ultimately a $[0, 1]$ variable in the linear program corresponding, informally, to $P(c)$.[2] Propositionalized constraints have previously been used with *integer* linear programming (ILP) using binary $\{0, 1\}$ values corresponding to $\{false, true\}$, to find an (exact) consistent truth assignment minimizing some cost and solve a global inference problem, e.g. (Roth and Yih, 2004; Roth and Yih, 2007). However, propositional linear programming has two significant advantages:

1. ILP is “winner take all”, shifting all belief to one claim in each mutual exclusion set (even when other claims are nearly as plausible) and finding the single most believable consistent *binary assignment*; we instead wish to find a *distribution* of belief over the claims that is consistent with our prior knowledge and as close as possible to the distribution produced by the fact-finder.
2. Linear programs can be solved in polynomial time (e.g. by interior point methods (Karmarkar, 1984)), but ILP is NP-hard.

To create our constraints, we first convert our propositional formula into conjunctive normal form. Then, for each disjunctive clause consisting of a set P of positive literals (claims) and a set N of negations of literals, we add the constraint $\sum_{c \in P} c_v + \sum_{c \in N} (1 - c_v) \geq 1$, where c_v denotes the $[0, 1]$ variable corresponding to each c. The left-hand side is the union bound of at least one of the claims being true (or false, in the case of negated literals); if this bound is at least 1, the constraint is satisfied. This optimism can dilute the strength of our constraints by ignoring potential dependence among claims: $x \Rightarrow y$, $x \vee y$ implies y is true, but since we demand only $y_v \geq x_v$ and $x_v + y_v \geq 1$ we accept any $y_v \geq 0.5$ where

[2] This is a slight mischaracterization, since our linear constraints only *approximate* intersections and unions of events (where each event is “claim c is true”), and we will be satisfying them subject to a linear cost function.

$y_v \geq x_v \geq 1 - y_v$. However, when the claims are mutually exclusive, the union bound is exact; a common constraint is of the form $q \Rightarrow r^1 \vee r^2 \vee \ldots$, where the r literals are mutually exclusive, which translates exactly to $r_v^1 + r_v^2 + \ldots \geq q_v$. Finally, observe that mutual exclusion amongst n claims c^1, c^2, …, c^n can be compactly written as $c_v^1 + c_v^2 + \ldots + c_v^n = 1$.

4.2 The Cost Function

Having seen how first-order logic can be converted to linear constraints, we now consider the cost function, a distance between the new distribution of belief satisfying our constraints and the original distribution produced by the fact-finder.

First we determine the number of "votes" received by each claim c, computed as $\omega_c = \omega(B(c))$, which should scale linearly with the certainty of the fact-finder's belief in c. Recall that the semantics of the belief score are particular to the fact-finder, so different fact-finders require different vote functions. TruthFinder has pseudo-probabilistic [0,1] beliefs, so we use $\omega_{inv}(x) = \min((1-x)^{\text{-1}}, m_{inv})$ with $m_{inv} = 10^{10}$ limiting the maximum number of votes possible; we assume $1/0 = \infty$. ω_{inv} intuitively scales with "error": a belief of 0.99 receives ten times the votes of 0.9 and has a tenth the error (0.01 vs. 0.1). For the remainder of the fact-finders whose beliefs are already "linear", we use the identity function $\omega_{idn}(x) = x$.

The most obvious choice for the cost function might be to minimize "frustrated votes": $\sum_{c \in C} \omega_c (1 - c_v)$. Unfortunately, this results in the linear solver generally assigning 1 to the variable in each mutual exclusion set with the most votes and 0 to all others (except when constraints prevent this), shifting all belief to the highest-vote claim and yielding poor performance. Instead, we wish to satisfy the constraints while keeping each c_v close to ω_c / ω_{M_c}, where $\omega_{M_c} = \sum_{d \in M_c} \omega_d$, and so shift belief among claims as little as possible. We use a weighted Manhattan distance called **VoteDistance**, where the cost for increasing the belief in a claim is proportional to the number of votes against it, and the cost for decreasing belief is proportional to the number of votes for it:

$$\sum_{c \in C} \max \left(\begin{matrix} (\omega_{M_c} - \omega_c) \cdot (c_v - \omega_c / \omega_{M_c}), \\ \omega_c \cdot (\omega_c / \omega_{M_c} - c_v) \end{matrix} \right)$$

Thus, the belief distribution found by our LP will be the one that satisfies the constraints while simultaneously minimizing the number of votes frustrated by the change from the original distribution. Note that for any linear expressions e and f we can implement $\max(e, f)$ in the objective function by replacing it with a new $[-\infty, \infty]$ helper variable x and adding the linear constraints $x \geq e$ and $x \geq f$.

4.3 From Values to Votes to Belief

Solving the LP gives us $[0, 1]$ values for each variable c_v, but we need to calculate an updated belief $B(c)$. We propose two methods for this:

Vote Conservation: $B(c) = \omega^{-1}(c_v \cdot \omega_{M_c})$
Vote Loss: $B(c) = \omega^{-1}(\min(\omega_c,\ c_v \cdot \omega_{M_c}))$

ω^{-1} is an inverse of the vote function: $\omega_{idn}^{-1}(x) = x$ and $\omega_{inv}^{-1}(x) = 1 - (1 + y)^{-1}$. Vote Conservation reallocates votes such that the total number of votes in each mutual exclusion set, ω_M, remains the same after the redistribution. However, if the constraints force c to lose votes, should we believe the other claims in M_c more? Under Vote Loss, a claim can *only* lose votes, ensuring that if other claims in M_c become less believable, c does not itself become more believable relative to claims in other mutual exclusion sets. We found Vote Loss just slightly better on average and used it for all reported results.

4.4 "Unknown" Augmentation

Augmenting our data with "Unknown" claims ensures that every LP is feasible and can be used to model our ignorance given a lack of sufficient information or conflicting constraints. An Unknown claim U_M is added to every mutual exclusion set M (but invisible to the fact-finder) and represents our belief that *none* of the claims in M are sufficiently supported. Now we can write the mutual exclusion constraint for M as $U_M + \sum_{c \in M} c_v = 1$. When propositionalizing FOL, if a disjunctive clause contains a non-negated literal for a claim c, then we add $\vee U_{M_c}$ to the clause.

For example, $Age(John, 35) \Rightarrow Age(Tom, 40)$ becomes $Age(John, 35) \Rightarrow Age(Tom, 40) \vee Age(Tom, Unknown)$. The only exception is when the clause contains claims from only one mutual exclusion set (e.g. "I know Sam is 50 or 60"), and so the LP can only be infeasible if the user directly asserts a contradiction (e.g. "Sam is 50 *and* Sam is 60"). The Unknown itself has a fixed number of votes that cannot be lost; this effectively "smooths" our belief in the claims and imposes a floor for believability. If $Age(Kim, 30)$ has 5 votes, $Age(Kim, 35)$ has 3 votes, and $Age(Kim, Unknown)$ is fixed at 6 votes, we hold that Kim's age is unknown due to lack of evidence. The number of votes that should be given to each Unknown for this purpose depends, of course, on the particular fact-finder and ω function used; in our experiments, we are not concerned with establishing ignorance and thus assign 0 votes.

5 Experiments

Experiments were conducted over three domains (city population, basic biographies, and American vs. British spelling) with four datasets, all using the VoteDistance cost function and Vote Loss vote redistribution. We fixed the number of iterations of the framework (calculating $T^i(S)$, $B^i(S)$ and then solving the LP) at 20, which was found sufficient for all fact-finders. To evaluate accuracy, after the final iteration we look at each mutual exclusion set M and predict the highest-belief claim $c \in M$ (or, if u_M had the highest belief, the second-highest claim), breaking ties randomly, and check that it is the true claim t_M. We omit any M that does not contain a true claim (all known claims are false) and any M that is trivially correct (containing only one claim other than u_M). All results are shown in Table 1. **Vote** is the baseline, choosing either the claim occurring in the most Wikipedia revisions (in the Pop dataset) or claimed by the most sources (for all other datasets). **Sum** is Sums (Hubs and Authorities), **3Est** is 3-Estimates, **TF$^{\text{s}}$** is simplified TruthFinder, **TF$^{\text{c}}$** is "full" TruthFinder, **A·L** is Average·Log, **Inv**$^{1.2}$ is Investment with $g = 1.2$, and **Pool**$^{1.4}$ is PooledInvestment with $g = 1.4$.

5.1 IBT vs. L+I

We can enforce our prior knowledge against the beliefs produced by the fact-finder in each iteration, or we can apply these constraints just once, after running the fact-finder for 20 iterations without interference. By analogy to (Punyakanok et al., 2005), we refer to these approaches as inference based training (IBT) and learning + inference (L+I), respectively. Our results show that while L+I does better when prior knowledge is not entirely correct (e.g. "Growth" in the city population domain), generally performance is comparable when the effect of the constraints is mild, but IBT can outperform when prior knowledge is vital (as in the spelling domain) by allowing the fact-finder to learn from the provided corrections.

5.2 Wikipedia Infoboxes

To focus on the performance of the framework, we (like previous fact-finding work) naively assume that our data are accurately extracted, but we also require large corpora. Wikipedia Infoboxes (Wu and Weld, 2007) are a semi-structured source covering many domains with readily available authorship, and we produced our city population and basic biographic datasets from the most recent full-history dump of the English Wikipedia (taken January 2008). However, attribution is difficult: if an author edits the page but not the claim within the infobox, is the author implicitly agreeing with (and asserting) the claim? The best performance was achieved by being strict for City Population data, counting only the direct editing of a claim, and lax for Biography data, counting any edit. We hypothesize this is because editors may lack specific knowledge about a city's population (and thus fail to correct an erroneous value) but incorrect birth or death dates are more noticeable.

5.3 Results

5.3.1 City Population

We collected infoboxes for settlements (Geobox, Infobox Settlement, Infobox City, etc.) to obtain 44,761 populations claims qualified by year (e.g. $pop(Denver, 598707, 2008)$), with 4,107 authors total. We took as our "truth" U.S. census data, which gave us 308 non-trivial true facts to test against. Our "common sense" knowledge is that population grows

Table 1: Experimental Results ($\emptyset$ indicates no prior knowledge; all values are percent accuracy) Some results are omitted here (see text). A·L, Inv$^{1.2}$, Pool$^{1.4}$ are our novel algorithms

Dataset	Prior Knowledge	Vote	Sum	3Est	TF^s	TF^c	A·L	Inv$^{1.2}$	Pool$^{1.4}$
Pop	$\emptyset$	81.49	81.82	81.49	82.79	84.42	80.84	**87.99**	80.19
Pop	Growth$_{IBT}$	82.79	79.87	77.92	82.79	**86.36**	80.52	85.39	79.87
Pop	Growth$_{L+I}$	82.79	79.55	77.92	83.44	85.39	80.52	**89.29**	80.84
Pop	Larger$^{2500}_{IBT}$	85.39	85.06	80.52	86.04	87.34	84.74	**89.29**	84.09
Pop	Larger$^{2500}_{L+I}$	85.39	85.06	80.52	86.69	86.69	84.42	**89.94**	84.09
SynPop	$\emptyset$	73.45	87.76	84.87	56.12	87.07	**90.23**	89.41	90.00
SynPop	Pop±8%$_{IBT}$	88.31	95.46	92.16	**96.42**	95.46	96.15	95.46	**96.42**
SynPop	Pop±8%$_{L+I}$	88.31	94.77	92.43	82.39	95.32	95.59	**96.29**	96.01
Bio	$\emptyset$	89.80	89.53	89.80	73.04	**90.09**	89.24	88.34	90.01
Bio	CS$_{IBT}$	89.20	89.61	89.20	72.44	89.91	89.35	88.60	**90.20**
Bio	CS$_{L+I}$	89.20	89.61	89.20	57.10	90.09	89.35	88.49	**90.24**
Bio	CS+Decades$_{IBT}$	90.58	90.88	90.58	80.30	91.25	90.91	90.02	**91.32**
Bio	CS+Decades$_{L+I}$	90.58	90.91	90.58	69.27	90.95	90.91	90.09	**91.17**
Spell	$\emptyset$	13.54	9.37	11.96	**41.93**	7.93	10.23	9.36	9.65
Spell	Words$^{100}_{IBT}$	13.69	9.02	12.72	**44.28**	8.05	9.98	11.11	8.86
Spell	Words$^{100}_{L+I}$	13.69	8.86	12.08	**46.54**	8.05	9.98	9.34	7.89
Spell	CS+Words$^{100}_{IBT}$	35.10	31.88	35.10	56.52	29.79	32.85	73.59	**80.68**
Spell	CS+Words$^{100}_{L+I}$	35.10	31.72	34.62	**55.39**	22.06	32.21	30.92	29.95

over time ("Growth" in table 1); therefore, $\forall_{v,w,x,y,z} pop(v,w,y) \wedge pop(v,x,z) \wedge y < z \Rightarrow x > w$. Of course, this often does not hold true: cities can shrink, but performance was nevertheless superior to no prior knowledge whatsoever. The L+I approach does appreciably better because it avoids forcing these sometimes-incorrect constraints onto the claim beliefs while the fact-finder iterates (which would propagate the resulting mistakes), instead applying them only at the end where they can correct more errors than they create. The sparsity of the data plays a role–only a fraction of cities have population claims for multiple years, and those that do are typically larger cities where the correct claim is asserted by an overwhelming majority, greatly limiting the potential benefit of our Growth constraints. We also considered prior knowledge of the relative sizes of some cities, randomly selecting 2500 pairs of them (a, b), where a was more populous than b in year t, asserting $\forall_{x,y} pop(a,x,t) \wedge pop(b,y,t) \Rightarrow x > y$. This "Larger" prior knowledge proved more effective than our oft-mistaken Growth constraint, with modest improvement to the highest-performing Investment fact-finder, and Investment$_{L+I}$ reaches **90.91%** with 10,000 such pairs.

5.3.2 Synthetic City Population

What if attribution were certain and the data more dense? To this end we created a synthetic dataset. We chose 100 random (real) cities and created 100 authors whose individual accuracy a was drawn uniformly from $[0,1]$. Between 1 and 10 claims (also determined uniformly) were made about each city in each year from 2000 to 2008 by randomly-selected authors. For each city with true population p and year, four incorrect claims were created with populations selected uniformly from $[0.5p, 1.5p]$, each author claiming p with probability a and otherwise asserting one of the four incorrect claims. Our common-sense knowledge was that population did not change by more than 8% per year (also tried on the Wikipedia dataset but with virtually no effect). Like "Growth", "Pop±8%" does not always hold, but a change of more than 8% is much rarer than a shrinking city. These constraints greatly improved results, although we note this would diminish if inaccurate claims had less variance around the true population.

5.3.3 Basic Biographies

We scanned infoboxes to find 129,847 claimed birth dates, 34,201 death dates, 10,418 parent-child pairs, and 9,792 spouses. To get "true" birth and death dates, we extracted data from several online repositories (after satisfying ourselves that they were independent and not derived from Wikipedia!), eliminating any date these sources disagreed upon, and ultimately obtained a total of 2,685 dates to test against. Our common sense ("CS") knowledge was: nobody dies before they are born, people are infertile before the age of 7, nobody lives past 125, all spouses have overlapping lifetimes, no child is born more than a year after a parent's (father's) death, nobody has more than two parents, and nobody is born or dies after 2008 (the "present day", the year of the Wikipedia dump). Applying this knowledge roughly halved convergence times, but had little effect on the results due to data sparsity similar to that seen in the population data–while we know many birthdays and some death dates, relatively few biographies had parent-child and spouse claims. To this we also added knowledge of the decade (but not the exact date) in which 15,145 people were born ("CS+Decades"). Although common sense alone does not notably improve results, it does very well in conjunction with specific knowledge.

5.3.4 American vs. British Spelling

Prior knowledge allows us to find a truth that conforms with the user's viewpoint, even if that viewpoint differs from the norm. After obtaining a list of words with spellings that differed between American and British English (e.g. "color" vs. "colour"), we examined the British National Corpus as well as Washington Post and Reuters news articles, taking the source's (the article author's) use of a disputed word as a claim that his spelling was correct. Our goal was to find the "true" British spellings that conformed to a British viewpoint, but American spellings predominate by far. Consequently, without prior knowledge the fact-finders do very poorly against our test set of 694 British words, predicting American spelling instead in accordance with the great majority of authors (note that accuracy from an American perspective is 1−"British" accuracy). Next we assumed that the user already knew the correct spelling of 100 random words (removing these from the test set, of course), but with little effect. Finally, we added our common sense ("CS") knowledge: if a spelling a is correct and of length ≥ 4, then if a is a substring of b, $a \Leftrightarrow b$ (e.g. colour $\Leftrightarrow$ colourful). Furthermore, while we do not know a priori whether a spelling is American or British, we do know if e and f are different spellings of the same word, and, if two such spellings have a chain of implication between them, we can break all links in this chain (while some American spellings will still be linked to British spellings, this removes most such errors). Interestingly, common sense alone actually *hurts* results (e.g. PooledInvestment (IBT) gets 6.2%), as it essentially makes the fact-finders more adept at finding the predominant American spellings! However, when some correct spellings are known, results improve greatly and demonstrate IBT's ability to spread strong prior knowledge, easily surpassing L+I. Results improve further with more known spellings (PooledInvestment gets **84.86%** with CS+Words$^{200}_{IBT}$).

6 Conclusion

We have introduced a new framework for incorporating prior knowledge into a fact-finding system, along with several new high-performing fact-finding algorithms (Investment, PooledInvestment, and Average·Log). While the benefits of prior knowledge were most dramatic in the Spelling domain, we saw gains from both "common sense" and specific knowledge in all experiments–even the difficult Biography domain saw faster convergence with common sense alone and notably higher results when specific knowledge was added. We find that while prior knowledge is helpful in reducing error, when the user's viewpoint disagrees with the norm it becomes absolutely essential and, formulated as a linear program, it need not be the computational burden that might otherwise be expected.

Acknowledgements

This research was partly sponsored by the Army Research Laboratory (ARL) (accomplished under Cooperative Agreement Number W911NF-09-2-0053). Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the ARL.

References

Adler, B T and L de Alfaro. 2007. A content-driven reputation system for the Wikipedia. *WWW '07*, 7:261–270.

Artz, D and Y Gil. 2007. A survey of trust in computer science and the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58–71, June.

Brin, S and L Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117.

Dong, X, L Berti-equille, and D Srivastava. 2009a. Integrating conflicting data: the role of source dependence. *Technical report, AT&T Labs-Research, Florham Park, NJ*.

Dong, X.L., L. Berti-Equille, and Divesh Srivastava. 2009b. Truth discovery and copying detection in a dynamic world. *VLDB*, 2(1):562–573.

Galland, Alban, Serge Abiteboul, A. Marian, and Pierre Senellart. 2010. Corroborating information from disagreeing views. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 131–140. ACM.

Josang, A., S. Marsh, and S. Pope. 2006. Exploring different types of trust propagation. *Lecture Notes in Computer Science*, 3986:179.

Josang, A. 1997. Artificial reasoning with subjective logic. *2nd Australian Workshop on Commonsense Reasoning*.

Kamvar, S, M Schlosser, and H Garcia-molina. 2003. The Eigentrust algorithm for reputation management in P2P networks. *WWW '03*.

Karmarkar, N. 1984. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395.

Kleinberg, J M. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632.

Levien, R. 2008. Attack-resistant trust metrics. *Computing with Social Trust*, pages 121–132.

Manchala, D.W. 1998. Trust metrics, models and protocols for electronic commerce transactions. *Proceedings. 18th International Conference on Distributed Computing Systems (Cat. No.98CB36183)*, pages 312–321.

Marsh, S. 1994. Formalising Trust as a Computational Concept. *PhD thesis, University of Stirling*.

Novak, V, I Perfilieva, and J Mockof. 1999. *Mathematical principles of fuzzy logic*. Kluwer Academic Publishers.

Punyakanok, V., D. Roth, W. Yih, and D. Zimak. 2005. Learning and inference over constrained output. In *International Joint Conference on Artificial Intelligence*, volume 19.

Roth, Dan and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8.

Roth, D and W Yih. 2007. Global Inference for Entity and Relation Identification via a Linear Programming Formulation. In Getoor, Lise and Ben Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press.

Russell, Stuart and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition.

Sabater, Jordi and Carles Sierra. 2005. Review on Computational Trust and Reputation Models. *Artificial Intelligence Review*, 24(1):33–60, September.

Shafer, G. 1976. *A mathematical theory of evidence*. Princeton University Press Princeton, NJ.

Wu, Fei and Daniel S. Weld. 2007. Autonomously semantifying wikipedia. *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management - CIKM '07*, page 41.

Yin, Xiaoxin, Philip S. Yu, and Jiawei Han. 2008. Truth Discovery with Multiple Conflicting Information Providers on the Web. *IEEE Transactions on Knowledge and Data Engineering*, 20(6):796–808.

Yu, Bin and Munindar P. Singh. 2003. Detecting deception in reputation management. *Proceedings of the second international joint conference on Autonomous agents and multiagent systems - AAMAS '03*, page 73.

Zeng, H, M Alhossaini, L Ding, R Fikes, and D L McGuinness. 2006. Computing trust from revision history. *Intl. Conf. on Privacy, Security and Trust*.

Using Web-scale N-grams to Improve Base NP Parsing Performance

Emily Pitler
Computer and Information Science
University of Pennsylvania
epitler@seas.upenn.edu

Shane Bergsma
Department of Computing Science
University of Alberta
sbergsma@ualberta.ca

Dekang Lin
Google, Inc.
lindek@google.com

Kenneth Church
Human Language Technology Center of Excellence
Johns Hopkins University
kenneth.church@jhu.edu

Abstract

We use web-scale N-grams in a base NP parser that correctly analyzes 95.4% of the base NPs in natural text. Web-scale data improves performance. That is, there is no data like more data. Performance scales log-linearly with the number of parameters in the model (the number of unique N-grams). The web-scale N-grams are particularly helpful in harder cases, such as NPs that contain conjunctions.

1 Introduction

Noun phrases (NPs) provide an index to the world's information. About 70% of web queries are NPs (Barr et al., 2008). A robust NP parser could help search engines improve retrieval performance on multi-word NP queries (Zhai, 1997). For example, by knowing the correct parse of "washed (baby carrots)," a search engine could ensure that returned pages (and advertisements) concern clean carrots rather than clean babies. NP structure is also helpful for query expansion and substitution (Jones et al., 2006).

This paper is concerned with base NP parsing. We are given a base NP string as input, and the task is to produce a parse tree as output. Base NPs are NPs that do not contain embedded noun phrases. These are sometimes called NP chunks, or core/non-recursive NPs (Church, 1988; Ramshaw and Marcus, 1995). Correctly parsing (or, equivalently, bracketing) base NPs is challenging because the same part-of-speech (POS) sequence can be parsed differently depending on the specific words involved. For example, "*retired (science teacher)*" and "*(social science) teacher*" have different structures even though they have identical POS sequences.

Lexical statistics are therefore needed in order to parse the above examples, and they must be computed over a lot of text to avoid sparsity. All of our lexical statistics are derived from a new and improved web-scale N-gram corpus (Lin et al., 2010), which we call Google V2.

Despite the importance of base NPs, most sentence parsers do not parse base NPs, since the main training corpus for parsers, the Penn Treebank (PTB) (Marcus et al., 1994), leaves a flat structure for base NPs. Recent annotations by Vadas and Curran (2007a) added NP structure to the PTB. We use these annotations (described in Section 3) for our experiments.

NP parsers usually focus on bracketing three-word noun compounds. Parsing three-word noun compounds is a fairly artificial task; we show that sequences of three nouns make up less than 1% of the three-word-or-longer base NPs in natural text. As the NP length increases, the number of possible binary trees (parses) increases with the Catalan numbers (Church and Patil, 1982). NPs of length three have just two possible parses (chance is 50%), while NPs of length six already have forty-two possible parses (chance is 2%). Long NPs therefore provide much more opportunity to improve performance over the baseline. In Table 1 (Section 7), we show the distribution of base NP length in the PTB. While most NPs are of length three, NP length has a long tail.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 886–894,
Beijing, August 2010

The three-word noun compound assumption also restricts research to the case in which all words are nouns, while base NPs also contain determiners, possessives, adjectives, and conjunctions. Conjunctions and their scopes are particularly challenging. For example, in the NP, "*French television and movie producers*," a parser should conjoin "(*television*) *and* (*movie*)," as opposed to "(*French television*) *and* (*movie*)," "(*French television*) *and* (*movie producers*)" or "(*television*) *and* (*movie producers*)."

To resolve these issues, we train a classifier which uses contextual information from the entire NP and lexical statistics derived from the web-scale N-gram corpus to predict if a given span is a constituent. Our parser then uses this classifier to produce a score for every possible NP-internal bracketing and creates a chart of bracketing scores. This chart can be used as features in a full sentence parser or parsed directly with a chart parser. Our parses are highly accurate, creating a strong new standard for this task.

Finally, we present experiments that investigate the effects of N-gram frequency cutoffs and various sources of N-gram data. We show an interesting relationship between accuracy and the number of unique N-gram types in the data.

2 Related Work

2.1 Three-Word Noun Compounds

The most commonly used data for NP parsing is from Lauer (1995), who extracted 244 three-word noun compounds from the Grolier encyclopedia. When there are only three words, this task reduces to a binary decision:

- Left Branching: * [*retired science*] *teacher*
- Right Branching: *retired* [*science teacher*]

In Lauer (1995)'s set of noun compounds, two-thirds are left branching.

The main approach to these three-word noun compounds has been to compute association statistics between pairs of words and then choose the bracketing that corresponds to the more highly associated pair. The two main models are the *adjacency model* (Marcus, 1980; Liberman and Sproat, 1992; Pustejovsky et al., 1993; Resnik, 1993) and the *dependency model* (Lauer, 1995). Under the adjacency model, the bracketing decision is made by comparing the associations between words one and two versus words two and three (i.e. comparing *retired science* versus *science teacher*). In contrast, the dependency model compares the associations between one and two versus one and three (*retired science* versus *retired teacher*). Lauer (1995) compares the two models and finds the dependency model to be more accurate.

Nakov and Hearst (2005) compute the association scores using frequencies, conditional probabilities, χ^2, and mutual information, for both pairs of words and for linguistically-motivated paraphrases. Lapata and Keller (2005) found that using web-scale data for associations is better than using the (smaller) 100M-word British National Corpus.

2.2 Longer NPs

Focusing on only the three word case misses a large opportunity for base NP parsing. NPs longer than three words commonly occur, making up 29% of our test set. In addition, a chance baseline does exponentially worse as the length of the NP increases. These longer NPs are therefore a major opportunity to improve overall base NP parsing.

Since in the general case, NP parsing can no longer be thought of as a single binary classification problem, different strategies are required.

Barker (1998) reduces the task of parsing longer NPs to making sequential three-word decisions, moving a sliding window along the NP. The window is first moved from right-to-left, inserting right bracketings, and then again from left-to-right, finalizing left bracketings. While Barker (1998) assumes that these three-word decisions can be made in isolation, this is not always valid.[1] Vadas and Curran (2007b) employ Barker's algorithm, but use a supervised classifier to make the sequential bracketing decisions. Because these approaches rely on a sequence of binary decisions,

[1] E.g., although the right-most three words are identical in 1) "*soap opera stars and television producers*," and 2) "*movie and television producers*," the initial right-bracketing decision for "*and television producers*" should be different in each.

early mistakes can cascade and lead to a chain of incorrect bracketings.

Our approach differs from previous work in NP parsing; rather than greedily inserting brackets as in Barker's algorithm, we use dynamic programming to find the global maximum-scoring parse. In addition, unlike previous approaches that have used local features to make local decisions, we use the full NP to score each potential bracketing.

A related line of research aims to *segment* longer phrases that are queried on Internet search engines (Bergsma and Wang, 2007; Guo et al., 2008; Tan and Peng, 2008). Bergsma and Wang (2007) focus on NP queries of length four or greater. They use supervised learning to make segmentation decisions, with features derived from the noun compound bracketing literature. Evaluating the benefits of *parsing* NP queries, rather than simply segmenting them, is a natural application of our system.

3 Annotated Data

Our training and testing data are derived from recent annotations by Vadas and Curran (2007a). The original PTB left a flat structure for base noun phrases. For example, "*retired science teacher,*" would be represented as:

(NP (JJ retired) (NN science) (NN teacher))

Vadas and Curran (2007a) annotated NP-internal structure by adding annotations whenever there is a left-bracketing. If no annotations were added, right-branching is assumed. The inter-annotator agreement for exactly matching the brackets on an NP was 98.5%.

This data provides a valuable new resource for parsing research, but little work has so far made use of it. Vadas and Curran (2007b) perform some preliminary experiments on NP bracketing, but use gold standard part-of-speech and named-entity annotations as features in their classifier. Our work establishes a strong and realistic standard on this data; our results will serve as a basis for further research on this topic.

4 Unlabeled N-gram Data

All of our N-gram features described in Section 6.1 rely on probabilities derived from unlabeled data. To use the largest amount of data possible, we exploit web-scale N-gram corpora. N-gram counts are an efficient way to compress large amounts of data (such as all the text on the web) into a manageable size. An N-gram corpus records how often each unique sequence of words occurs. Co-occurrence probabilities can be calculated directly from the N-gram counts. To keep the size manageable, N-grams that occur with a frequency below a particular threshold can be filtered.

The corpus we use is **Google V2** (Lin et al., 2010): a new N-gram corpus with N-grams of length 1-5 that we created from the same 1 trillion word snapshot of the web as Google N-grams Version 1 (Brants and Franz, 2006), but with several enhancements. Duplicate sentences are removed, as well as "sentences" which are probably noise (indicated by having a large proportion of non-alphanumeric characters, being very long, or being very short). Removing duplicate sentences is especially important because automatically-generated websites, boilerplate text, and legal disclaimers skew the source web data, with sentences that may have only been authored once occurring millions of times. We use the suffix array tools described in Lin et al. (2010) to quickly extract N-gram counts.

5 Base NP Parsing Approach

Our goal is to take a base NP string as input and produce a parse tree as output. In practice, it would be most useful if the NP parse could be integrated into a sentence parser. Previous NP parsers are difficult to apply in practice.[2] Work in prepositional phrase attachment that assumes gold-standard knowledge of the competing attachment sites has been criticized as unrealistic (Atterer and Schütze, 2007).

Our system can easily be integrated into full parsers. Its input can be identified quickly and reliably and its output is compatible with downstream parsers.

[2] For example, Vadas and Curran (2007b) report results on NP parsing, but these results include NPs containing prepositional or adverbial phrases (confirmed by personal communication). Practical application of their system would therefore require resolving prepositional phrase attachment as a preprocessing step.

Our parser's input is base NPs, which can be identified with very high accuracy. Kudo and Matsumoto (2001) report 95.8% NP chunking accuracy on PTB data.

Once provided with an NP, our system uses a supervised classifier to predict the probability of a particular contiguous subsequence (span) of the NP being a constituent, given the entire NP as context. This probability can be inserted into the chart that a standard chart parser would use.

For example, the base NP "*French television and movie producers*" would be decomposed into nine different classification problems, scoring the following potential bracketings:

(French television) and movie producers
French (television and) movie producers
(French television and) movie producers ...
French television and (movie producers)

In Section 6, we detail the set of statistical and structural features used by the classifier.

The output of our classifier can be easily used as a feature in a full-sentence structured prediction parser, as in Taskar et al. (2004). Alternatively, our work could be integrated into a full-sentence parser by using our feature representations directly in a discriminative CFG parser (Finkel et al., 2008), or in a parse re-ranker (Ratnaparkhi et al., 1994; Collins and Koo, 2005; Charniak and Johnson, 2005).

While our main objective is to use web-scale lexical statistics to create an accurate classifier for base NP-internal constituents, we do produce a parse tree for evaluation purposes. The probability of a parse tree is defined as the product of the probabilities of all the spans (constituents) in the tree. The most probable tree is computed with the CYK algorithm.

6 Features

Over the course of development experiments, we discovered that the more position-specific our features were, the more effectively we could parse NPs. We define a word's position as its distance from the right of the full NP, as the semantic head of NPs is most often the right-most word. Ultimately, we decided to conjoin each feature with the position of the proposed bracketing. Since the features for differing proposed bracketings are now disjoint, this is equivalent to scoring bracketings with different classifiers, with each classifier chosen according to the bracketing position. We now outline the feature types that are common, but weighted differently, in each proposed bracketing's feature set.

6.1 N-gram Features

All of the features described in this section require estimates of the probability of specific words or sequences of words. All probabilities are computed using **Google V2** (Section 4).

6.1.1 PMI

Recall that the adjacency model for the three-word task uses the associations of the two pairs of adjacent words, while the dependency model uses the associations of the two pairs of attachment sites for the initial noun. We generalize the adjacency and dependency models by including the pointwise mutual information (Church and Hanks, 1990) between *all* pairs of words in the NP:

$$\mathrm{PMI}(x, y) = \log \frac{p(\text{“x y”})}{p(\text{“x”})p(\text{“y”})} \qquad (1)$$

For NPs of length n, for each proposed bracketing, we include separate features for the PMI between all $\binom{n}{2}$ pairs of words in the NP. For NPs including conjunctions, we include additional PMI features (Section 6.1.2).

Since these features are also tied to the proposed bracketing positions (as explained above), this allows us to learn relationships between various associations within the NP and each potential bracketing. For example, consider a proposed bracketing from word 4 to word 5. We learn that a high association of words inside a bracketing (here, a high association between word 4 and word 5) indicates a bracketing is likely, while a high association between words that cross a proposed bracketing (e.g., a high association between word 3 and word 4) indicates the bracketing is unlikely.

The value of these features is the PMI, if it is defined. If the PMI is undefined, we include one of two binary features:
$p(\text{“x y”}) = 0$ or $p(\text{“x”}) \vee p(\text{“y”}) = 0$.

We illustrate the PMI features with an example. In deciding whether (*movie producers*) is a reasonable bracketing within "*French television and movie producers*," the classifier weighs features for all of:

PMI(*French, television*)
PMI(*French, and*)
...
PMI(*television, producers*)
PMI(*and, producers*)
PMI(*movie, producers*)

6.1.2 Conjunctions

Properly handling NPs containing conjunctions (NP+conj) requires special statistical features. For example, *television* and *movie* are commonly conjoined, but the relevant statistics that suggest placing brackets around the phrase "*television and movie*" are not provided by the above PMI features (i.e., this is not clear from PMI(*television, and*), PMI(*television, movie*), nor PMI(*and, movie*)). Rather, we want to know if the full phrase "television and movie" is common.

We thus have additional NP+conj features that consider the PMI association across the word *and*:

$$\text{PMI}_{and}(x, y) = \log \frac{p(\text{“}x\ and\ y\text{”})}{p(\text{“}x\ and\text{”})p(\text{“}and\ y\text{”})} \quad (2)$$

When PMI$_{and}$ between a pair of words is high, they are likely to be the constituents of a conjunction.

Let $NP{=}(w_1 \ldots w_{i-1}, \text{‘}and\text{’}, w_{i+1} \ldots w_n)$ be an NP+conj. We include the PMI$_{and}$ features between w_{i-1} and all $w \in w_{i+1} \ldots w_n$. In the example "*French television and movie producers*," we would include features PMI$_{and}$(*television, movie*) and PMI$_{and}$(*television, producers*).

In essence, we are assuming w_{i-1} is the head of one of the items being conjoined, and we score the likelihood of each of the words to the right of the *and* being the head for the other item. In our running example, the conjunction has narrow scope, and PMI$_{and}$(*television, movie*) is greater than PMI$_{and}$(*television, producers*), indicating to our classifier that (*television and movie*) is a good bracketing. In other examples the conjunction will join heads that are further apart, as in *((French TV) and (British radio)) stars*, where both of the following hold:

PMI$_{and}$(TV, radio) $>$ PMI$_{and}$(TV, British)
PMI$_{and}$(TV, radio) $>$ PMI$_{and}$(TV, stars)

6.2 Lexical

We include a binary feature to indicate the presence of a particular word at each position in the NP. We learn that, for instance, the word *Inc.* in names tends to occur outside of brackets.

6.3 Shape

Previous work on NP bracketing has used gold-standard named entity tags (Vadas and Curran, 2007b) as features. We did not want to use any gold-standard features in our experiments, however NER information is helpful in separating premodifiers from names, i.e. *(news reporter) (Walter Cronkite).*

As an expedient way to get both NER information and useful information from hyphenated adjectives, abbreviations, and other punctuation, we normalize each string using the following regular expressions:

[A-Z]+ → A   [a-z]+ → a

We use this normalized string as an indicator feature. E.g. the word "Saudi-born" will fire the binary feature "Aa-a."

6.4 Position

We also include the position of the proposed bracketing as a feature. This represents the prior of a particular bracketing, regardless of the actual words.

7 Experiments

7.1 Experimental Details

We use Vadas and Curran (2007a)'s annotations (Section 3) to create training, development and testing data for base NPs, using standard splits of the Penn Treebank (Table 1). We consider all non-trivial base NPs, i.e., those longer than two words.

For training, we expand each NP in our training set into independent examples corresponding to all the possible internal NP-bracketings, and represent these examples as feature vectors (Section 5). Each example is positively labeled if it is

Data Set	Train	Dev	Test	Chance
PTB Section	2-22	24	23	
Length=3	41353	1428	2498	50%
Length=4	12067	445	673	20%
Length=5	3930	148	236	7%
Length=6	1272	34	81	2%
Length>6	616	29	34	< 1%
Total NPs	59238	2084	3522	

Table 1: Breakdown of the PTB base NPs used in our experiments. Chance = 1/Catalan(length).

Features	All NPs	NP+conj	NP-conj
All features	**95.4**	**89.7**	**95.7**
-N-grams	94.0	84.0	94.5
-lexical	92.2	87.4	92.5
-shape	94.9	**89.7**	95.2
-position	95.3	**89.7**	95.6
Right bracketing	72.6	58.3	73.5

Table 2: Accuracy (%) of base NPs parsing; ablation of different feature classes.

consistent with the gold-standard bracketing, otherwise it is a negative example.

We train using LIBLINEAR, an efficient linear Support Vector Machine (SVM).[3] We use an L2-loss function, and optimize the regularization parameter on the development set (reaching an optimum at C=1). We converted the SVM output to probabilities.[4] Perhaps surprisingly, since SVMs are not probabilistic, performance on the development set with these SVM-derived probabilities was higher than using probabilities from the LIBLINEAR logistic regression solver.

At test time, we again expand the NPs and calculate the probability of each constituent, inserting the score into a chart. We run the CYK algorithm to find the most probable parse of the entire NP according to the chart. Our evaluation metric is **Accuracy**: the proportion of times our proposed parse of the NP exactly matches the gold standard.

8 Results

8.1 Base NPs

Our method improves substantially over the baseline of assuming a completely right-branching structure, 95.4% versus 72.6% (Table 2). The accuracy of the constituency classifier itself (before the CYK parser is used) is 96.1%.

The lexical features are most important, but all feature classes are somewhat helpful. In particular, including N-gram PMI features significantly improves the accuracy, from 94.0% to 95.4%.[5] Correctly parsing more than 19 base NPs out of 20 is an exceptional level of accuracy, and provides a strong new standard on this task. The most comparable result is by Vadas and Curran (2007b), who achieved 93.0% accuracy on a different set of PTB noun phrases (see footnote 2), but their classifier used features based on gold-standard part-of-speech and named-entity information.

Exact match is a tough metric for parsing, and the difficulty increases as the length of the NP increases (because there are more decisions to make correctly). At three word NPs, our accuracy is 98.5%; by six word NPs, our accuracy drops to 79.0% (Figure 1). Our method's accuracy decreases as the length of the NP increases, but much less rapidly than a right-bracketing or chance baseline.

8.2 Base NPs with Conjunctions

N-gram PMI features help more on NP+conj than on those that do not contain conjunctions (NP-conj) (Table 2). N-gram PMI features are the most important features for NP+conj, increasing accuracy from 84.0% to 89.7%, a 36% relative reduction in error.

8.3 Effect of Thresholding N-gram data

We now address two important related questions: 1) how does our parser perform as the amount of unlabeled auxiliary data varies, and 2) what is the effect of thresholding an N-gram corpus? The second question is of widespread relevance as

[3] www.csie.ntu.edu.tw/~cjlin/liblinear/

[4] Following instructions in http://www.csie.ntu.edu.tw/~cjlin/liblinear/FAQ.html

[5] McNemar's test, $p < 0.05$

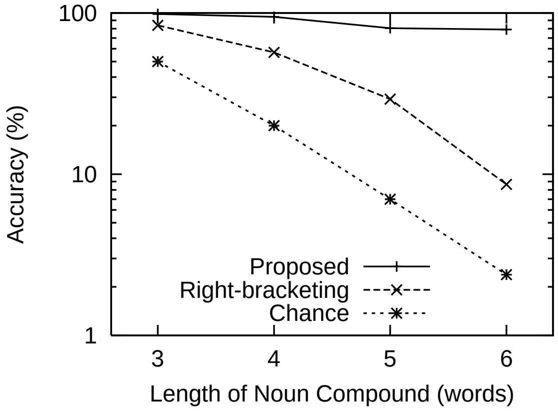


Figure 1: Accuracy (log scale) over different NP lengths, of our method, the right-bracketing baseline, and chance (1/Catalan(length)).

thresholded N-gram corpora are now widely used in NLP. Without thresholds, web-scale N-gram data can be unmanageable.

While we cannot lower the threshold after creating the N-gram corpus, we can raise it, filtering more N-grams, and then measure the relationship between threshold and performance.

Threshold	Unique N-grams	Accuracy
10	4,145,972,000	95.4%
100	391,344,991	95.3%
1,000	39,368,488	95.2%
10,000	3,924,478	94.8%
100,000	386,639	94.8%
1,000,000	37,567	94.4%
10,000,000	3,317	94.0%

Table 3: There is no data like more data. Accuracy improves with the number of parameters (unique N-grams).

We repeat the parsing experiments while including in our PMI features only N-grams with a count $\geq$10 (the whole data set), $\geq$100, $\geq$1000, . . ., $\geq 10^7$. All other features (lexical, shape, position) remain unchanged. The N-gram data almost perfectly exhibits Zipf's power law: raising the threshold by a factor of ten decreases the number of unique N-grams by a factor of ten (Table 3). The improvement in accuracy scales log-linearly with the number of unique N-grams. From a practical standpoint, we see a trade-off between storage and accuracy. There are consistent improvements in accuracy from lowering the threshold and increasing the amount of auxiliary data. If for some application it is necessary to reduce storage by several orders of magnitude, then one can easily estimate the resulting impact on performance.

Corpus	# of tokens	τ	# of types
NEWS	3.2 B	1	3.7 B
Google V1	1,024.9 B	40	3.4 B
Google V2	207.4 B	10	4.1 B

Table 4: N-gram data, with total number of words (*tokens*) in the original corpus (in billions, B), frequency threshold used to filter the data, τ, and total number of unique N-grams (*types*) remaining in the data after thresholding.

We repeat the thresholding experiments using two other N-gram sources:

NEWS: N-gram data created from a large set of news articles including the Reuters and Gigaword (Graff, 2003) corpora, not thresholded.

Google V1: The original web-scale N-gram corpus (Section 4).

Details of these sources are given in Table 4.

For a given number of unique N-grams, using any of the three sources does about the same (Figure 2). It does not matter that the source corpus for Google V1 is about five times larger than the source corpus for Google V2, which in turn is sixty-five times larger than NEWS (Table 4). Accuracies increase linearly with the log of the number of *types* in the auxiliary data set.

Google V1 is the one data source for which the relationship between accuracy and number of N-grams is not monotonic. After about 100 million unique N-grams, performance starts decreasing. This drop shows the need for Google V2. Since Google V1 contains duplicated web pages and sentences, mistakes that should be rare can appear to be quite frequent. Google V2, which comes from the same snapshot of the web as Google V1, but has only unique sentences, does not show this drop.

We regard the results in Figure 2 as a companion to Banko and Brill (2001)'s work on exponentially increasing the amount of labeled training data. Here we see that varying the amount of

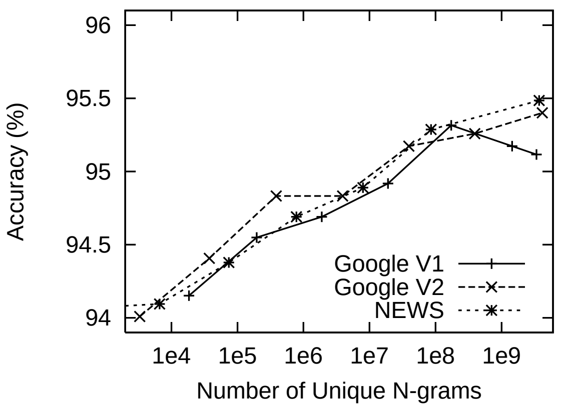


Figure 2: There is no data like more data. Accuracy improves with the number of parameters (unique N-grams). This trend holds across three different sources of N-grams.

unlabeled data can cause an equally predictable improvement in classification performance, without the cost of labeling data.

Suzuki and Isozaki (2008) also found a log-linear relationship between unlabeled data (up to a billion words) and performance on three NLP tasks. We have shown that this trend continues well beyond Gigaword-sized corpora. Brants et al. (2007) also found that more unlabeled data (in the form of input to a language model) leads to improvements in BLEU scores for machine translation.

Adding noun phrase parsing to the list of problems for which there is a “bigger is better” relationship between performance and unlabeled data shows the wide applicability of this principle. As both the amount of text on the web and the power of computer architecture continue to grow exponentially, collecting and exploiting web-scale auxiliary data in the form of N-gram corpora should allow us to achieve gains in performance linear in time, without any human annotation, research, or engineering effort.

9 Conclusion

We used web-scale N-grams to produce a new standard in performance of base NP parsing: 95.4%. The web-scale N-grams substantially improve performance, particularly in long NPs that include conjunctions. There is no data like more data. Performance improves log-linearly with the number of parameters (unique N-grams). One can increase performance with larger models, e.g., increasing the size of the unlabeled corpora, or by decreasing the frequency threshold. Alternatively, one can decrease storage costs with smaller models, e.g., decreasing the size of the unlabeled corpora, or by increasing the frequency threshold. Either way, the log-linear relationship between accuracy and model size makes it easy to estimate the trade-off between performance and storage costs.

Acknowledgments

We gratefully acknowledge the Center for Language and Speech Processing at Johns Hopkins University for hosting the workshop at which this research was conducted.

References

Atterer, M. and H. Schütze. 2007. Prepositional phrase attachment without oracles. *Computational Linguistics*, 33(4):469–476.

Banko, M. and E. Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *ACL*.

Barker, K. 1998. A trainable bracketer for noun modifiers. In *Twelfth Canadian Conference on Artificial Intelligence (LNAI 1418)*.

Barr, C., R. Jones, and M. Regelson. 2008. The linguistic structure of English web-search queries. In *EMNLP*.

Bergsma, S. and Q.I. Wang. 2007. Learning noun phrase query segmentation. In *EMNLP-CoNLL*.

Brants, T. and A. Franz. 2006. The Google Web 1T 5-gram Corpus Version 1.1. LDC2006T13.

Brants, T., A.C. Popat, P. Xu, F.J. Och, and J. Dean. 2007. Large language models in machine translation. In *EMNLP*.

Charniak, E. and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *ACL*.

Church, K.W. and P. Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Church, K. and R. Patil. 1982. Coping with syntactic ambiguity or how to put the block in the box on the table. *Computational Linguistics*, 8(3-4):139–149.

Church, K.W. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *ANLP*.

Collins, M. and T. Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

Finkel, J.R., A. Kleeman, and C.D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *ACL*.

Graff, D. 2003. English Gigaword. LDC2003T05.

Guo, J., G. Xu, H. Li, and X. Cheng. 2008. A unified and discriminative model for query refinement. In *SIGIR*.

Jones, R., B. Rey, O. Madani, and W. Greiner. 2006. Generating query substitutions. In *WWW*.

Kudo, T. and Y. Matsumoto. 2001. Chunking with support vector machines. In *NAACL*.

Lapata, M. and F. Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1):1–31.

Lauer, M. 1995. Corpus statistics meet the noun compound: some empirical results. In *ACL*.

Liberman, M. and R. Sproat. 1992. The stress and structure of modified noun phrases in English. *Lexical matters*, pages 131–181.

Lin, D., K. Church, H. Ji, S. Sekine, D. Yarowsky, S. Bergsma, K. Patil, E. Pitler, R. Lathbury, V. Rao, K. Dalwani, and S. Narsale. 2010. New tools for web-scale n-grams. In *LREC*.

Marcus, M.P., B. Santorini, and M.A. Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Marcus, M.P. 1980. *Theory of Syntactic Recognition for Natural Languages*. MIT Press, Cambridge, MA, USA.

Nakov, P. and M. Hearst. 2005. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *CoNLL*.

Pustejovsky, J., P. Anick, and S. Bergler. 1993. Lexical semantic techniques for corpus analysis. *Computational Linguistics*, 19(2):331–358.

Ramshaw, L.A. and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *3rd ACL Workshop on Very Large Corpora*.

Ratnaparkhi, A., S. Roukos, and R.T. Ward. 1994. A maximum entropy model for parsing. In *Third International Conference on Spoken Language Processing*.

Resnik, P. 1993. *Selection and information: a class-based approach to lexical relationships*. Ph.D. thesis, University of Pennsylvania.

Suzuki, J. and H. Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *ACL*.

Tan, B. and F. Peng. 2008. Unsupervised query segmentation using generative language models and Wikipedia. In *WWW*.

Taskar, B., D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *EMNLP*.

Vadas, D. and J.R. Curran. 2007a. Adding noun phrase structure to the Penn Treebank. In *ACL*.

Vadas, D. and J.R. Curran. 2007b. Large-scale supervised models for noun phrase bracketing. In *PACLING*.

Zhai, C. 1997. Fast statistical parsing of noun phrases for document indexing. In *ANLP*.

Citation Summarization Through Keyphrase Extraction

Vahed Qazvinian
Department of EECS
University of Michigan
vahed@umich.edu

Dragomir R. Radev
School of Information and
Department of EECS
University of Michigan
radev@umich.edu

Arzucan Özgür
Department of EECS
University of Michigan
ozgur@umich.edu

Abstract

This paper presents an approach to summarize single scientific papers, by extracting its contributions from the set of citation sentences written in other papers. Our methodology is based on extracting significant keyphrases from the set of citation sentences and using these keyphrases to build the summary. Comparisons show how this methodology excels at the task of single paper summarization, and how it out-performs other multi-document summarization methods.

1 Introduction

In recent years statistical physicists and computer scientists have shown great interest in analyzing complex adaptive systems. The study of such systems can provide valuable insight on the behavioral aspects of the involved agents with potential applications in economics and science. One such aspect is to understand what motivates people to provide the $n + 1^{st}$ review of an artifact given that they are unlikely to add something significant that has not already been said or emphasized. Citations are part of such complex systems where articles use citations as a way to mention different contributions of other papers, resulting in a collective system.

The focus of this work is on the corpora created based on citation sentences. A citation sentence is a sentence in an article containing a citation and can contain zero or more *nuggets* (i.e., non-overlapping contributions) about the cited article. For example the following sentences are a few citation sentences that appeared in the NLP literature in past that talk about Resnik's work.

> *The STRAND system* **(Resnik, 1999)**, *for example, uses structural markup information from the pages, without looking at their content, to attempt to align them.*

> **Resnik (1999)** *addressed the issue of language identification for finding Web pages in the languages of interest.*

> *Mining the Web for bilingual text* **(Resnik, 1999)** *is not likely to provide sufficient quantities of high quality data..*

The set of citations is important to analyze because human summarizers have put their effort collectively but independently to read the target article and cite its important contributions. This has been shown in other work too (Elkiss et al., 2008; Nanba et al., 2004; Qazvinian and Radev, 2008; Mei and Zhai, 2008; Mohammad et al., 2009). In this work, we introduce a technique to summarize the set of citation sentences and cover the major contributions of the target paper. Our methodology first finds the set of keyphrases that represent important information units (i.e., nuggets), and then finds the best set of k sentences to cover more, and more important nuggets.

Our results confirm the effectiveness of the method and show that it outperforms other state of the art summarization techniques. Moreover, as shown in the paper, this method does not need to calculate the full cosine similarity matrix for a document cluster, which is the most time consuming part of the mentioned baseline methods.

1.1 Related Work

Previous work has used citations to produce summaries of scientific work (Qazvinian and Radev,

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 895–903,
Beijing, August 2010

2008; Mei and Zhai, 2008; Elkiss et al., 2008). Other work (Bradshaw, 2003; Bradshaw, 2002) benefits from citations to determine the content of articles and introduce “Reference Directed Indexing” to improve the results of a search engine.

In other work, (Nanba and Okumura, 1999) analyze citation sentences and automatically categorize citations into three groups using 160 predefined phrase-based rules to support a system for writing a survey. Previous research has shown the importance of the citation summaries in understanding what a paper contributes. In particular, (Elkiss et al., 2008) performed a large-scale study on citation summaries and their importance. Results from this experiment confirmed that the “Self Cohesion” (Elkiss et al., 2008) of a citation summary of an article is consistently higher than the that of its abstract and that citations contain additional information that does not appear in abstracts.

Kan et al. (2002) use annotated bibliographies to cover certain aspects of summarization and suggest using metadata and critical document features as well as the prominent content-based features to summarize documents. Kupiec et al. (1995) use a statistical method and show how extracts can be used to create summaries but use no annotated metadata in summarization.

Siddharthan and Teufel describe a new task to decide the scientific attribution of an article (Siddharthan and Teufel, 2007) and show high human agreement as well as an improvement in the performance of Argumentative Zoning (Teufel, 2005). Argumentative Zoning is a rhetorical classification task, in which sentences are labeled as one of Own, Other, Background, Textual, Aim, Basis, Contrast according to their role in the author’s argument. These all show the importance of citation summaries and the vast area for new work to analyze them to produce a summary for a given topic.

The Maximal Marginal Relevance (MMR) summarization method, which is based on a greedy algorithm, is described in (Carbonell and Goldstein, 1998). MMR uses the full similarity matrix to choose the sentences that are the least similar to the sentences already selected for the summary. We selected this method as one of our baseline methods, which we have explained in more details in Section 4.

Fact	Occurrences
f_1: “ Supervised Learning”	5
f_2: “ instance/concept relations”	3
f_3: “Part-of-Speech tagging”	3
f_4: “filtering QA results”	2
f_5: “lexico-semantic information”	2
f_6: “hyponym relations”	2

Table 2: Nuggets of P03-1001 extracted by annotators.

2 Data

In order to evaluate our method, we use the ACL Anthology Network (AAN), which is a collection of papers from the Computational Linguistics journal and proceedings from ACL conferences and workshops and includes more than $13,000$ papers (Radev et al., 2009). We use 25 manually annotated papers from (Qazvinian and Radev, 2008), which are highly cited articles in AAN. Table 1 shows the ACL ID, title, and the number of citation sentences for these papers.

The annotation guidelines asked a number of annotators to read the citation summary of each paper and extract a list of the main contributions of that paper. Each item on the list is a non-overlapping contribution (nugget) perceived by reading the citation summary. The annotation strictly instructed the annotators to focus on the citing sentences to do the task and not their own background on the topic. Then, extracted nuggets are reviewed and those nuggets that have only been mentioned by 1 annotator are removed. Finally, the union of the rest is used as a set of nuggets representing each paper.

Table 2 lists the nuggets extracted by annotators for P03-1001.

3 Methodology

Our methodology assumes that each citation sentence covers 0 or more nuggets about the cited papers, and tries to pick sentences that maximize nugget coverage with respect to summary length.

These nuggets are essentially represented using keyphrases. Therefore, we try to extract significant keyphrases in order to represent nuggets each sentence contains. Here, the keyphrases are ex-

ACL-ID	Title	# citations
N03-1017	Statistical Phrase-Based Translation	180
P02-1006	Learning Surface Text Patterns For A Question Answering System	74
P05-1012	On-line Large-Margin Training Of Dependency Parsers	71
C96-1058	Three New Probabilistic Models For Dependency Parsing: An Exploration	66
P05-1033	A Hierarchical Phrase-Based Model For Statistical Machine Translation	65
P97-1003	Three Generative, Lexicalized Models For Statistical Parsing	55
P99-1065	A Statistical Parser For Czech	54
J04-4002	The Alignment Template Approach To Statistical Machine Translation	50
D03-1017	Towards Answering Opinion Questions: Separating Facts From Opinions ...	42
P05-1013	Pseudo-Projective Dependency Parsing	40
W00-0403	Centroid-Based Summarization Of Multiple Documents: Sentence Extraction, ...	31
P03-1001	Offline Strategies For Online Question Answering: Answering Questions Before They Are Asked	27
N04-1033	Improvements In Phrase-Based Statistical Machine Translation	24
A00-2024	Cut And Paste Based Text Summarization	20
W00-0603	A Rule-Based Question Answering System For Reading Comprehension Tests	19
A00-1043	Sentence Reduction For Automatic Text Summarization	19
C00-1072	The Automated Acquisition Of Topic Signatures For Text Summarization	19
W05-1203	Measuring The Semantic Similarity Of Texts	17
W03-0510	The Potential And Limitations Of Automatic Sentence Extraction For Summarization	15
W03-0301	An Evaluation Exercise For Word Alignment	14
A00-1023	A Question Answering System Supported By Information Extraction	13
D04-9907	Scaling Web-Based Acquisition Of Entailment Relations	12
P05-1014	The Distributional Inclusion Hypotheses And Lexical Entailment	10
H05-1047	A Semantic Approach To Recognizing Textual Entailment	8
H05-1079	Recognising Textual Entailment With Logical Inference	9

Table 1: List of papers chosen from AAN for evaluation together with the number of sentences citing each.

	unique	all	max freq
unigrams	229,631	7,746,792	437,308
bigrams	2,256,385	7,746,791	73,957
3-grams	5,125,249	7,746,790	3,600
4-grams	6,713,568	7,746,789	2,408

Table 3: Statistics on the abstract corpus in AAN used as the background data

pressed using N-grams, and thus these building units are the key elements to our summarization. For each citation sentence d_i, our method first extracts a set of important keyphrases, D_i, and then tries to find sentences that have a larger number of important and non-redundant keyphrases. In order to take the first step, we extract statistically significantly frequent N-grams (up to $N = 4$) from each citing sentence and use them as the set of representative keyphrases for that citing sentence.

3.1 Automatic Keyphrase Extraction

A list of keyphrases for each citation sentence can be generated by extracting N-grams that occur significantly frequently in that sentence compared to a large corpus of such N-grams. Our method for such an extraction is inspired by the previous work by Tomokiyo and Hurst (Tomokiyo and Hurst, 2003).

A language model, $\mathcal{M}$, is a statistical model that assigns probabilities to a sequence of N-grams. Every language model is a probability distribution over all N-grams and thus the probabilities of all N-grams of the same length sum up to 1. In order to extract keyphrases from a text using statistical significance we need two language models. The first model is referred to as the *Background Model ($\mathcal{BM}$)* and is built using a large text corpus. Here we build the BM using the text of all the paper abstracts provided in AAN[1]. The second language model is called the *Foreground Model ($\mathcal{FM}$)* and is the model built on the text from which keyphrases are being extracted. In this work, the set of all citation sentences that cite a particular target paper are used to build a foreground language model.

Let g^i be an N-gram of size i and $C_{\mathcal{M}}(g^i)$ denote the count of g^i in the model $\mathcal{M}$. First, we extract the counts of each N-grams in both the background ($\mathcal{BM}$) and the foreground corpora ($\mathcal{FM}$).

[1] http://chernobog.si.umich.edu/clair/anthology/index.cgi

$$
\begin{aligned}
M_{\mathcal{BM}} &= \sum_{g^i \in \{\mathcal{BM} \cup \mathcal{FM}\}} 1 \\
N_{\mathcal{BM}} &= \sum_{g^i \in \{\mathcal{BM} \cup \mathcal{FM}\}} C_{\mathcal{BM}}(g^i) \\
N_{\mathcal{FM}} &= \sum_{g^i \in \mathcal{FM}} C_{\mathcal{FM}}(g^i) \\
\hat{p}_{\mathcal{FM}}(g^i) &= C_{\mathcal{FM}}(g^i)/N_{\mathcal{FM}} \\
\hat{p}_{\mathcal{BM}}(g^i) &= (C_{\mathcal{BM}}(g^i)+1)/(M_{\mathcal{BM}}+N_{\mathcal{BM}})
\end{aligned}
$$

The last equation is also known as Laplace smoothing (Manning and Schutze, 2002) and handles the N-grams in the foreground corpus that have a 0 occurrence frequency in the background corpus. Next, we extract N-grams from the foreground corpus that have significant frequencies compared to the frequency of the same N-grams in the background model and its individual terms in the foreground model.

To measure how randomly a set of consecutive terms are forming an N-gram, Tomokiyo and Hurst (Tomokiyo and Hurst, 2003) use pointwise divergence. In particular, for an N-gram of size i, $g^i = (w_1 w_2 \cdots w_i)$,

$$\delta_{g^i}(\mathcal{FM}^i \| \mathcal{FM}^1) = \hat{p}_{\mathcal{FM}}(g^i) \log\left(\frac{\hat{p}_{\mathcal{FM}}(g^i)}{\prod_{j=1}^{i} \hat{p}_{\mathcal{FM}}(w_j)}\right)$$

This equation shows the extent to which the terms forming g^i have occurred together randomly. In other words, it indicates the extent of information that we lose by assuming independence of each word by applying the unigram model, instead of the N-gram model.

In addition, to measure how randomly a sequence of words appear in the foreground model with respect to the background model, we use pointwise divergence as well. Here, pointwise divergence defines how much information we lose by assuming that g^i is drawn from the background model instead of the foreground model:

$$\delta_{g^i}(\mathcal{FM}^i \| \mathcal{BM}^i) = \hat{p}_{\mathcal{FM}}(g^i) \log\left(\frac{\hat{p}_{\mathcal{FM}}(g^i)}{\hat{p}_{\mathcal{BM}}(g^i)}\right)$$

(Corley and Mihalcea, 2005) applied or utilized lexical based word overlap measures.
{overlap measures, word overlap, lexical based, utilized lexical}

Table 4: Example: citation sentence for W05-1203 written by D06-1621, and its extracted bigrams.

We set the criteria of choosing a sequence of words as significant to be whether it has positive pointwise divergence with respect to both the background model, and individual terms of the foreground model. In other words we extract all g^i from $\mathcal{FM}$ for which the both properties are positive:

$$
\begin{aligned}
\delta_{g^i}(\mathcal{FM}^i \| \mathcal{BM}^i) &> 0 \\
\delta_{g^i}(\mathcal{FM}^i \| \mathcal{FM}^1) &\geq 0
\end{aligned}
$$

The equality condition in the second equation is specifically set to handle unigrams, in which $\hat{p}_{\mathcal{FM}}(g^i) = \prod_{j=1}^{i} \hat{p}_{\mathcal{FM}}(w_j)$.

In order to handle the text corpora and building the language models, we have used the CMU-Cambridge Language Model toolkit (Clarkson and Rosenfeld, 1997). We use the set of citation sentences for each paper to build foreground language models. Furthermore, we employ this tool and make the background model using nearly 11,000 abstracts from AAN. Table 3 summarizes some of the statistics about the background data.

Once keyphrases (significant N-grams) of each sentence are extracted, we remove all N-grams in which more than half of the terms are stopwords. For instance, we remove all stopword unigrams, if any, and all bigrams with at least one stopword in them. For 3-grams and 4-grams we use a threshold of 2 and 3 stopwords respectively. After that, the set of remaining N-grams is used to represent each sentence and to build summaries. Table 4 shows an example of a citation sentence from D06-1621 citing W05-1203 (Corley and Mihalcea, 2005), and its extracted bigrams.

3.2 Sentence Selection

After extracting the set of keyphrases for each sentence, d_i, the sentence is represented using its set of N-grams, denoted by D_i. Then, the goal is to pick sentences (sets) for each paper that cover more important and non-redundant keyphrases. Essentially, keyphrases that have been repeated in more sentences are more important and could represent more important nuggets. Therefore, sentences that contain more frequent keyphrases are more important. Based on this intuition we define the reward of building a summary comprising a set of keyphrases S as

$$f(S) = |S \cap A|$$

where A is the set of all keyphrases from sentences not in the summary.

The set function f has three main properties. First, it is non-negative. Second, it is monotone (i.e., For every set v we have $f(S+v) \geq f(S)$). Third, f is sub-modular. The submodularity means that for a set v and two sets $S \subseteq T$ we have

$$f(S+v) - f(S) \geq f(T+v) - f(T)$$

Intuitively, this property implies that adding a set v to S will increase the reward at least as much as it would to a larger set T. In the summarization setting, this means that adding a sentence to a smaller summary will increase the reward of the summary at least as much as adding it to a larger summary that subsumes it. The following theorem formalizes this and is followed by a proof.

Theorem 1 *The reward function f is submodular.*

Proof
We start by defining a gain function $\mathcal{G}$ of adding sentence (set) D_i to $\mathcal{S}_{k-1}$ where $\mathcal{S}_{k-1}$ is the set of keyphrases in a summary built using $k-1$ sentences, and D_i is a candidate sentence to be added:

$$\mathcal{G}(D_i, \mathcal{S}_{k-1}) = f(\mathcal{S}_{k-1} \cup D_i) - f(\mathcal{S}_{k-1})$$

Simple investigation through a Venn diagram proof shows that $\mathcal{G}$ can be re-written as

$$\mathcal{G}(D_i, \mathcal{S}_{k-1}) = |D_i \cap (\cup_{j \neq i} D_j) - \mathcal{S}_{k-1}|$$

Let's denote $D_i \cap (\cup_{j \neq i} D_j)$ by $\cap_i$. The following equations prove the theorem.

$$\begin{aligned}
&\mathcal{S}_{k-1} \subseteq \mathcal{S}_k \\
&\mathcal{S}'_{k-1} \supseteq \mathcal{S}'_k \\
&\cap_i \cap \mathcal{S}'_{k-1} \supseteq \cap_i \cap \mathcal{S}'_k \\
&\cap_i - \mathcal{S}_{k-1} \supseteq \cap_i - \mathcal{S}_k \\
&| \cap_i - \mathcal{S}_{k-1}| \geq | \cap_i - \mathcal{S}_k| \\
&\mathcal{G}(D_i, \mathcal{S}_{k-1}) \geq \mathcal{G}(D_i, \mathcal{S}_k) \\
&f(\mathcal{S}_{k-1} \cup D_i) - f(\mathcal{S}_{k-1}) \geq f(\mathcal{S}_k \cup D_i) - f(\mathcal{S}_k)
\end{aligned}$$

Here, $\mathcal{S}'_k$ is the set of all N-grams in the vocabulary that are not present in $\mathcal{S}_k$. The gain of adding a sentence, D_i, to an empty summary is a non-negative value.

$$\mathcal{G}(D_i, \mathcal{S}_0) = C \geq 0$$

By induction, we will get

$$\mathcal{G}(D_i, \mathcal{S}_0) \geq \mathcal{G}(D_i, \mathcal{S}_1) \geq \cdots \geq \mathcal{G}(D_i, \mathcal{S}_k) \geq 0$$

□

Theorem 1 implies the general case of submodularity:

$$\forall m, n, 0 \leq m \leq n \leq |D| \Rightarrow \mathcal{G}(D_i, \mathcal{S}_m) \geq \mathcal{G}(D_i, \mathcal{S}_n)$$

Maximizing this submodular function is an NP-hard problem (Khuller et al., 1999). A common way to solve this maximization problem is to start with an empty set, and in each iteration pick a set that maximizes the gain. It has been shown before in (Kulik et al., 2009) that if f is a submodular, nondecreasing set function and $f(\emptyset) = 0$, then such a greedy algorithm finds a set $\mathcal{S}$, whose gain is at least as high as $(1 - 1/e)$ of the best possible solution. Therefore, we can optimize the keyphrase coverage as described in Algorithm 1.

4 Experimental Setup

We use the annotated data described in Section 2. In summary, the annotation consisted of two parts: nugget extraction and nugget distribution analysis. Five annotators were employed to annotate the sentences in each of the 25 citation summaries and write down the nuggets (non-overlapping contributions) of the target paper. Then using these

Summary generated using bigram-based keyphrases	
ID	Sentence
P06-1048:1	Ziff-Davis Corpus Most previous work (Jing 2000; Knight and Marcu 2002; Riezler et al 2003; Nguyen et al 2004a; Turner and Charniak 2005; McDonald 2006) has relied on automatically constructed parallel corpora for training and evaluation purposes.
J05-4004:18	Between these two extremes, there has been a relatively modest amount of work in sentence simplification (Chandrasekar, Doran, and Bangalore 1996; Mahesh 1997; Carroll et al 1998; Grefenstette 1998; Jing 2000; Knight and Marcu 2002) and document compression (Daume III and Marcu 2002; Daume III and Marcu 2004; Zajic, Dorr, and Schwartz 2004) in which words, phrases, and sentences are selected in an extraction process.
A00-2024:9	The evaluation of sentence reduction (see (Jing, 2000) for details) used a corpus of 500 sentences and their reduced forms in human-written abstracts.
N03-1026:17	To overcome this problem, linguistic parsing and generation systems are used in the sentence condensation approaches of Knight and Marcu (2000) and Jing (2000).
P06-2019:5	Jing (2000) was perhaps the first to tackle the sentence compression problem.

Table 5: Bigram-based summary generated for A00-1043.

Algorithm 1 The greedy algorithm for summary generation

```
$k \leftarrow$ the number of sentences in the summary
$D_i \leftarrow$ keyphrases in $d_i$
$\mathcal{S} \leftarrow \emptyset$
**for** $l$ = 1 to $k$ **do**
    $s_l \leftarrow \arg\max_{D_i \in D} |D_i \cap (\cup_{j \neq i} D_j)|$
    $\mathcal{S} \leftarrow \mathcal{S} \cup s_l$
    **for** $j$ = 1 to $|D|$ **do**
        $D_j \leftarrow D_j - s_l$
    **end for**
**end for**
**return** $\mathcal{S}$
```

nugget sets, each sentence was annotated with the nuggets it contains. This results in a sentence-fact matrix that helps with the evaluation of the summary. The summarization goal and the intuition behind the summarizing system is to select a few (5 in our experiments) sentences and cover as many nuggets as possible. Each sentence in a citation summary may contain 0 or more nuggets and not all nuggets are mentioned an equal number of times. Covering some nuggets (contributions) is therefore more important than others and should be weighted highly.

To capture this property, the pyramid score seems the best evaluation metric to use. We use the pyramid evaluation method (Nenkova and Passonneau, 2004) at the sentence level to evaluate the summary created for each set. We benefit from the list of annotated nuggets provided by the annotators as the ground truth of the summarization evaluation. These annotations give the list of nuggets covered by each sentence in each citation summary, which are equivalent to the *summarization content unit (SCU)* as described in (Nenkova and Passonneau, 2004).

The pyramid score for a summary is calculated as follows. Assume a pyramid that has n tiers, T_i, where tier $T_i > T_j$ if $i > j$ (i.e., T_i is not below T_j, and that if a nugget appears in more sentences, it falls in a higher tier.). Tier T_i contains nuggets that appeared in i sentences, and thus has weight i. Suppose $|T_i|$ shows the number of nuggets in tier T_i, and Q_i is the size of a subset of T_i whose members appear in the summary. Further suppose Q shows the sum of the weights of the facts that are covered by the summary. $Q = \sum_{i=1}^{n} i \times Q_i$. In addition, the optimal pyramid score for a summary with X facts, is

$$Max = \sum_{i=j+1}^{n} i \times |T_i| + j \times (X - \sum_{i=j+1}^{n} |T_i|)$$

where $j = \max_i(\sum_{t=i}^{n} |T_t| \geq X)$. The pyramid score for a summary is then calculated as follows.

$$P = \frac{Q}{Max}$$

This score ranges from 0 to 1, and a high score shows the summary contains more heavily weighted facts.

4.1 Baselines and Gold Standards

To evaluate the quality of the summaries generated by the greedy algorithm, we compare its pyramid score in each of the 25 citation summaries with those of a gold standard, a random summary, and four other methods. The gold standards are summaries created manually using 5 sentences. The 5 sentences are manually selected in a way to cover as many nuggets as possible with higher priority for the nuggets with higher frequencies. We also created random summaries using Mead (Radev et al., 2004). These summaries are basically a random selection of 5 sentences from the pool of sentences in the citation summary. Generally we expect the summaries created by the greedy method to be significantly better than random ones.

In addition to the gold and random summaries, we also used 4 baseline state of the art summarizers: LexRank, the clustering C-RR and C-LexRank, and Maximal Marginal Relevance (MMR). LexRank (Erkan and Radev, 2004) works based on a random walk on the cosine similarity of sentences and prints out the most frequently visited sentences. Said differently, LexRank first builds a network in which nodes are sentences and edges are cosine similarity values. It then uses the eigenvalue centralities to find the most central sentences. For each set, the top 5 sentences on the list are chosen for the summary.

The clustering methods, C-RR and C-LexRank, work by clustering the cosine similarity network of sentences. In such a network, nodes are sentences and edges are cosine similarity of node pairs. Clustering would intuitively put nodes with similar nuggets in the same clusters as they are more similar to each other. The C-RR method as described in (Qazvinian and Radev, 2008) uses a round-robin fashion to pick sentences from each cluster, assuming that the clustering will put the sentences with similar facts into the same clusters. Unlike C-RR, C-LexRank uses LexRank to find the most salient sentences in each cluster, and prints out the most central nodes of each cluster as summary sentences.

Finally, MMR uses the full cosine similarity matrix and greedily chooses sentences that are the least similar to those already selected for the summary (Carbonell and Goldstein, 1998). In particular,

$$MMR = \arg \min_{d_i \in D-A} \left[\max_{d_j \in A} Sim(d_i, d_j) \right]$$

where A is the set of sentences in the summary, initially set to $A = \emptyset$. This method is different from ours in that it chooses the least similar sentence to the summary in each iteration.

4.2 Results and Discussion

As mentioned before, we use the text of the abstracts of all the papers in AAN as the background, and each citation set as a separate foreground corpus. For each citation set, we use the method described in Section 3.1 to extract significant N-grams of each sentence. We then use the keyphrase set representation of each sentence to build the summaries using Algorithm 1. For each of the 25 citation summaries, we build 4 different summaries using unigrams, bigrams, 3-grams, and 4-grams respectively. Table 5 shows a 5-sentence summary created using algorithm 1 for the paper A00-1043 (Jing, 2000).

The pyramid scores for different methods are reported in Figure 1 together with the scores of gold standards, manually created to cover as many nuggets as possible in 5 sentences, as well as summary evaluations of the 4 baseline methods described above. This Figure shows how the keyphrase based summarization method when employing N-grams of size 3 or smaller, outperforms other baseline systems significantly. More importantly, Figure 1 also indicates that this method shows more stable results and low variation in summary quality when keyphrases of size 3 or smaller are employed. In contrast, MMR shows high variation in summary qualities making summaries that obtain pyramid scores as low as 0.15.

Another important advantage of this method is that we do not need to calculate the cosine similarity of the pairs of sentences, which would add a running time of $O(|D|^2|V|)$ in the number of documents, $|D|$, and the size of the vocabulary $|V|$ to the algorithm.

5 Conclusion and Future Work

This paper presents a summarization methodology that employs keyphrase extraction to find important contributions of scientific articles. The summarization is based on citation sentences and picks sentences to cover nuggets (represented by keyphrases) or contributions of the target papers. In this setting the best summary would have as few sentences and at the same time as many nuggets as possible. In this work, we use pointwise KL-divergence to extract statistically significant N-grams and use them to represent nuggets. We then apply a new set function for the task of summarizing scientific articles. We have proved that this function is submodular and concluded that a

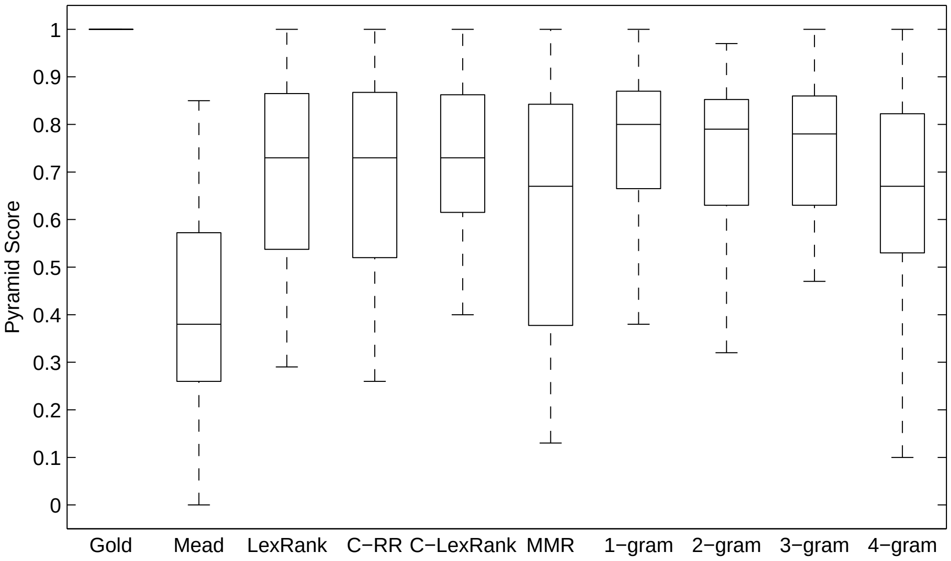


Figure 1: Evaluation Results (summaries with 5 sentences): The median pyramid score over 25 datasets using different methods.

greedy algorithm will result in a near-optimum set of covered nuggets using only 5 sentences. Our experiments in this paper confirm that the summaries created based on the presented algorithm are better than randomly generated summary, and also outperform other state of the art summarization methods in most cases. Moreover, we show how this method generates more stable summaries with lower variation in summary quality when N-grams of size 3 or smaller are employed.

A future direction for this work is to perform post-processing on the summaries and re-generate sentences that cover the extracted nuggets. However, the ultimate goal is to eventually develop systems that can produce summaries of entire research areas, summaries that will enable researchers to easily and quickly switch between fields of research.

One future study that will help us generate better summaries is to understand how nuggets are generated by authors. In fact, modeling the nugget coverage behavior of paper authors will help us identify more important nuggets and discover some aspects of the paper that would otherwise be too difficult by just reading the paper itself.

6 Acknowledgements

This work is in part supported by the National Science Foundation grant “iOPENER: A Flexible Framework to Support Rapid Learning in Unfamiliar Research Domains”, jointly awarded to University of Michigan and University of Maryland as IIS 0705832, and in part by the NIH Grant U54 DA021519 to the National Center for Integrative Biomedical Informatics.

Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the supporters.

References

Bradshaw, Shannon. 2002. *Reference Directed Indexing: Indexing Scientific Literature in the Context of Its Use*. Ph.D. thesis, Northwestern University.

Bradshaw, Shannon. 2003. Reference directed indexing: Redeeming relevance for subject search in citation indexes. In *Proceedings of the 7th European*

Conference on Research and Advanced Technology for Digital Libraries.

Carbonell, Jaime G. and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR'98*, pages 335–336.

Clarkson, PR and R Rosenfeld. 1997. Statistical language modeling using the cmu-cambridge toolkit. *Proceedings ESCA Eurospeech*, 47:45–148.

Elkiss, Aaron, Siwei Shen, Anthony Fader, Güneş Erkan, David States, and Dragomir R. Radev. 2008. Blind men and elephants: What do citation summaries tell us about a research article? *Journal of the American Society for Information Science and Technology*, 59(1):51–62.

Erkan, Güneş and Dragomir R. Radev. 2004. Lexrank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research*.

Jing, Hongyan. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the sixth conference on Applied natural language processing*, pages 310–315, Morristown, NJ, USA. Association for Computational Linguistics.

Kan, Min-Yen, Judith L. Klavans, and Kathleen R. McKeown. 2002. Using the Annotated Bibliography as a Resource for Indicative Summarization. In *Proceedings of LREC 2002*, Las Palmas, Spain.

Khuller, Samir, Anna Moss, and Joseph (Seffi) Naor. 1999. The budgeted maximum coverage problem. *Inf. Process. Lett.*, 70(1):39–45.

Kulik, Ariel, Hadas Shachnai, and Tami Tamir. 2009. Maximizing submodular set functions subject to multiple linear constraints. In *SODA '09*, pages 545–554.

Kupiec, Julian, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *SIGIR '95*, pages 68–73, New York, NY, USA. ACM.

Manning, Christopher D. and Hirich Schutze. 2002. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, London, England.

Mei, Qiaozhu and ChengXiang Zhai. 2008. Generating impact-based summaries for scientific literature. In *Proceedings of ACL '08*, pages 816–824.

Mohammad, Saif, Bonnie Dorr, Melissa Egan, Ahmed Hassan, Pradeep Muthukrishan, Vahed Qazvinian, Dragomir Radev, and David Zajic. 2009. Using citations to generate surveys of scientific paradigms. In *NAACL 2009*, pages 584–592, June.

Nanba, Hidetsugu and Manabu Okumura. 1999. Towards multi-paper summarization using reference information. In *IJCAI1999*, pages 926–931.

Nanba, Hidetsugu, Noriko Kando, and Manabu Okumura. 2004. Classification of research papers using citation links and citation types: Towards automatic review article generation. In *Proceedings of the 11th SIG Classification Research Workshop*, pages 117–134, Chicago, USA.

Nenkova, Ani and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. *Proceedings of the HLT-NAACL conference*.

Qazvinian, Vahed and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *COLING 2008*, Manchester, UK.

Radev, Dragomir, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. 2004. MEAD - a platform for multidocument multilingual text summarization. In *LREC 2004*, Lisbon, Portugal, May.

Radev, Dragomir R., Pradeep Muthukrishnan, and Vahed Qazvinian. 2009. The ACL anthology network corpus. In *ACL workshop on Natural Language Processing and Information Retrieval for Digital Libraries*.

Siddharthan, Advaith and Simone Teufel. 2007. Whose idea was this, and why does it matter? attributing scientific work to citations. In *Proceedings of NAACL/HLT-07*.

Teufel, Simone. 2005. Argumentative Zoning for Improved Citation Indexing. *Computing Attitude and Affect in Text: Theory and Applications*, pages 159–170.

Tomokiyo, Takashi and Matthew Hurst. 2003. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions*, pages 33–40.

2D Trie for Fast Parsing

Xian Qian, Qi Zhang, Xuanjing Huang, Lide Wu
Institute of Media Computing
School of Computer Science, Fudan University
{xianqian, qz, xjhuang, ldwu}@fudan.edu.cn

Abstract

In practical applications, decoding speed is very important. Modern structured learning technique adopts template based method to extract millions of features. Complicated templates bring about abundant features which lead to higher accuracy but more feature extraction time. We propose Two Dimensional Trie (2D Trie), a novel efficient feature indexing structure which takes advantage of relationship between templates: feature strings generated by a template are prefixes of the features from its extended templates. We apply our technique to Maximum Spanning Tree dependency parsing. Experimental results on Chinese Tree Bank corpus show that our 2D Trie is about 5 times faster than traditional Trie structure, making parsing speed 4.3 times faster.

1 Introduction

In practical applications, decoding speed is very important. Modern structured learning technique adopts template based method to generate millions of features. Such as shallow parsing (Sha and Pereira, 2003), named entity recognition (Kazama and Torisawa,), dependency parsing (McDonald et al., 2005), etc.

The problem arises when the number of templates increases, more features generated, making the extraction step time consuming. Especially for maximum spanning tree (MST) dependency parsing, since feature extraction requires quadratic time even using a first order model. According to Bohnet's report (Bohnet, 2009), a fast feature extraction beside of a fast parsing algorithm is important for the parsing and training speed. He takes 3 measures for a 40X speedup, despite the same inference algorithm. One important measure is to store the feature vectors in file to skip feature extraction, otherwise it will be the bottleneck.

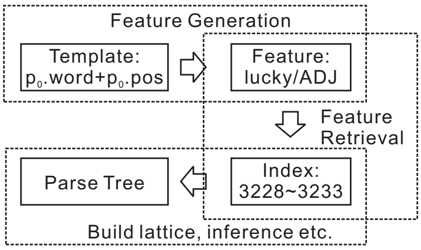


Figure 1: Flow chart of dependency parsing. p_0.*word*, p_0.*pos* denotes the word and POS tag of parent node respectively. Indexes correspond to the features conjoined with dependency types, e.g., *lucky/ADJ/OBJ*, *lucky/ADJ/NMOD*, etc.

Now we quickly review the feature extraction stage of structured learning. Typically, it consists of 2 steps. First, features represented by strings are generated using templates. Then a feature indexing structure searches feature indexes to get corresponding feature weights. Figure 1 shows the flow chart of MST parsing, where p_0.*word*, p_0.*pos* denote the word and POS tag of parent node respectively.

We conduct a simple experiment to investigate decoding time of MSTParser, a state-of-the-art java implementation of dependency parsing [1]. Chinese Tree Bank 6 (CTB6) corpus (Palmer and

[1] http://sourceforge.net/projects/mstparser

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 904–912,
Beijing, August 2010

Step	Feature Generation	Index Retrieval	Other	Total
Time	300.27	61.66	59.48	421.41

Table 1: Time spent of each step (seconds) of MSTParser on CTB6 standard test data (2660 sentences). Details of the hardware and corpus are described in section 5

Xue, 2009) with standard train/development/test split is used for evaluation. Experimental results are shown in Table 1. The observation is that time spent of inference is trivial compared with feature extraction. Thus, speeding up feature extraction is critical especially when large template set is used for high accuracy.

General indexing structure such as Hash and Trie does not consider the relationships between templates, therefore they could not speed up feature generation, and are not completely efficient for searching feature indexes. For example, feature string s_1 generated by template "p$_0$.*word*" is prefix of feature s_2 from template "p$_0$.*word* + c$_0$.*word*" (word pair of parent and child), hence index of s_1 could be used for searching s_2. Further more, if s_1 is not in the feature set, then s_2 must be absent, its generation can be skipped.

We propose Two Dimensional Trie (2D Trie), a novel efficient feature indexing structure which takes advantage of relationship between templates. We apply our technique to Maximum Spanning Tree dependency parsing. Experimental results on CTB6 corpus show that our 2D Trie is about 5 times faster than traditional Trie structure, making parsing speed 4.3 times faster.

The paper is structured as follows: in section 2, we describe template tree which represents relationship between templates; in section 3, we describe our new 2D Trie structure; in section 4, we analyze the complexity of the proposed method and general string indexing structures for parsing; experimental results are shown in section 5; we conclude the work in section 6.

2 Template tree

2.1 Formulation of template

A template is a set of template units which are manually designed: $T = \{t_1, \dots, t_m\}$. For convenience, we use another formulation: $T = t_1 + \dots + t_m$. All template units appearing in this paper are described in Table 2, most of them are widely used. For example, "T = p$_0$.*word* + c$_0$.*word*$|^l$ " denotes the word pair of parent and child nodes, conjoined with their distance.

Unit	Meaning
p$_{-i}$/p$_i$	the i^{th} node left/right to parent node
c$_{-i}$/c$_i$	the i^{th} node left/right to child node
r$_{-i}$/r$_i$	the i^{th} node left/right to root node
n.*word*	word of node n
n.*pos*	POS tag of node n
n.*length*	word length of node n
$\|^l$	conjoin current feature with linear distance between child node and parent node
$\|_d$	conjoin current feature with direction of dependency (left/right)

Table 2: Template units appearing in this paper

2.2 Template tree

In the rest of the paper, for simplicity, let s_i be a feature string generated by template T_i.

We define the relationship between templates: T_1 is the **ancestor** of T_2 if and only $T_1 \subset T_2$, and T_2 is called the **descendant** of T_1. Recall that, feature string s_1 is prefix of feature s_2. Suppose $T_3 \subset T_1 \subset T_2$, obviously, the most efficient way to look up indexes of s_1, s_2, s_3 is to search s_3 first, then use its index id_3 to search s_1, and finally use id_1 to search s_2. Hence the relationship between T_2 and T_3 can be neglected.

Therefore we define **direct ancestor** of T_1: T_2 is a direct ancestor of T_1 if $T_2 \subset T_1$, and there is no template T' such that $T_2 \subset T' \subset T_1$. Correspondingly, T_1 is called the **direct descendant** of T_2.

Template graph $G = (V, E)$ is a directed graph that represents the relationship between templates, where $V = \{T_1, \dots, T_n\}$ is the template set, $E = \{e_1, \dots, e_N\}$ is the edge set. Edge from T_i to T_j exists, if and only if T_i is the direct ancestor of T_j. For templates having no ancestor, we add an empty template as their common direct ancestor, which is also the root of the graph.

The left part of Figure 2 shows a template graph for templates T_1 =p$_0$.*word*, T_2 =p$_0$.*pos* , T_3 =p$_0$.*word* + p$_0$.*pos*. In this example, T_3 has 2 direct ancestors, but in fact s_3 has only one prefix

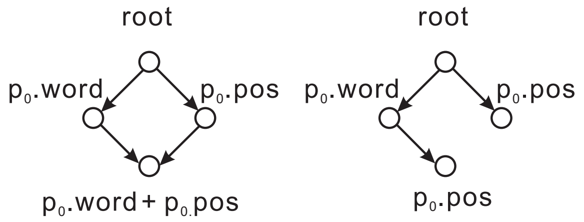


Figure 2: Left graph shows template graph for T_1 =p$_0$.*word*, T_2 =p$_0$.*pos* , T_3 =p$_0$.*word* + p$_0$.*pos*. Right graph shows the corresponding template tree, where each vertex saves the subset of template units that do not belong to its father

which depends on the order of template units in generation step. If $s_3 = s_1 + s_2$, then its prefix is s_1, otherwise its prefix is s_2. In this paper, we simply use the breadth-first tree of the graph for disambiguation, which is called **template tree**. The only direct ancestor T_1 of T_2 in the tree is called **father** of T_2, and T_2 is a **child** of T_1. The right part of Figure 2 shows the corresponding template tree, where each vertex saves the subset of template units that do not belong to its father.

2.3 Virtual vertex

Consider the template tree in the left part of Figure 3, red vertex and blue vertex are partially overlapped, their intersection is p$_0$.*word*, if string s from template T =p$_0$.*word* is absent in feature set, then both nodes can be neglected. For efficiently pruning candidate templates, each vertex in template tree is restricted to have exactly one template unit (except root). Another important reason for such restriction will be given in the next section.

To this end, virtual vertexes are created for multi-unit vertexes. For efficient pruning, the new virtual vertex should extract the most common template unit. A natural goal is to minimize the creation number. Here we use a simple greedy strategy, for the vertexes sharing a common father, the most frequent common unit is extracted as new vertex. Virtual vertexes are iteratively created in this way until all vertexes have one unit. The final template tree is shown in the right part of Figure 3, newly created virtual vertexes are shown in dashed circle.

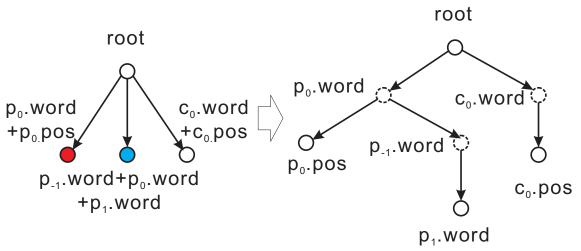


Figure 3: Templates that are partially overlapped: $T_{red} \cap T_{blue}$ =p$_0$.*word*, virtual vertexes shown in dashed circle are created to extract the common unit

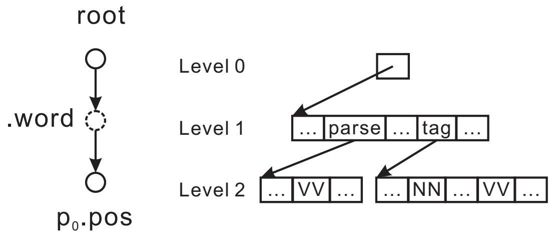


Figure 4: 2D Trie for single template, alphabets at level 1 and level 2 are the word set, POS tag set respectively

3 2D Trie

3.1 Single template case

Trie stores strings over a fixed alphabet, in our case, feature strings are stored over several alphabets, such as word list, POS tag list, etc. which are extracted from training corpus.

To illustrate 2D Trie clearly, we first consider a simple case, where only one template used. The template tree degenerates to a sequence, we could use a Trie like structure for feature indexing, the only difference from traditional Trie is that nodes at different levels could have different alphabets. One example is shown in Figure 4. There are 3 feature strings from template "p$_0$.*word* + p$_0$.*pos*": {*parse/VV, tag/VV, tag/VV*}. Alphabets at level 1 and level 2 are the word set, POS tag set respectively, which are determined by corresponding template vertexes.

As mentioned before, each vertex in template tree has exactly one template unit, therefore, at each level, we look up an index of a word or POS

He	PRP	2648	21
had	VBD	2731	27
been	VBN	1121	28
a	DT	0411	04
sales	NNS	5064	13
and	CC	0631	01
marketing	NN	3374	12
executive	NN	1923	12
with	IN	6023	06
Chrysler	NNP	1560	13
for	IN	2203	06
20	CD	0056	02
years	NNS	6778	14

Figure 5: Look up indexes of words and POS tags beforehand.

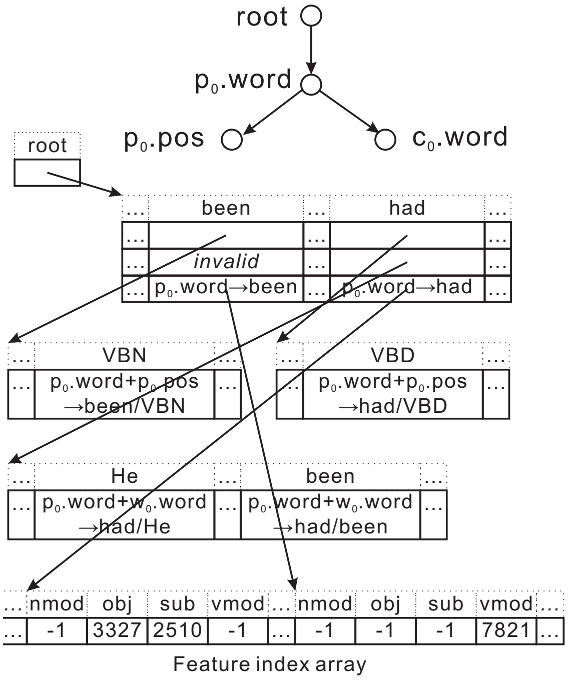


Figure 6: 2D trie for a general template tree. Dashed boxes are keys of columns, which are not stored in the structure

tag in sentence, not their combinations. Hence the number of alphabets is limited, and all the indexes could be searched beforehand for reuse, as shown in Figure 5, the token table is converted to a index table. For example, when generating features at position i of a sentence, template "r_0.*word* + r_1.*word*" requires index of $i+1^{th}$ word in the sentence, which could be reused for generation at position $i+1$.

3.2 General case

Generally, for vertex in template tree with K children, children of corresponding Trie node are arranged in a matrix of K rows and L columns, L is the size of corresponding alphabet. If the vertex is not virtual, i.e., it generates features, one more row is added at the bottom to store feature indexes. Figure 6 shows the 2D Trie for a general template tree.

3.3 Feature extraction

When extracting features for a pair of nodes in a sentence, template tree and 2D Trie are visited in breath first traversal order. Each time, an alphabet and a token index j from index table are selected according to current vertex. For example, POS tag set and the index of the POS tag of parent node are selected as alphabet and token index respectively for vertex "p_0.*pos*". Then children in the j^{th} column of the Trie node are visited, valid children and corresponding template vertexes are saved for further retrieval or generate feature indexes if the child is at the bottom and current Trie node is not virtual. Two queues are maintained to save the valid children and Trie nodes. Details of feature extraction algorithm are described in Algorithm 1.

3.4 Implementation

When feature set is very large, space complexity of 2D Trie is expensive. Therefore, we use Double Array Trie structure (Aoe, 1989) for implementation. Since children of 2D Trie node are arranged in a matrix, not an array, so each element of the base array has a list of bases, not one base in standard structure. For children that store features, corresponding bases are feature indexes. One example is shown in Figure 7. The root node has 3 bases that point to three rows of the child matrix of vertex "p_0.*word*" respectively. Number of bases in each element need not to be stored, since it can be obtained from template vertex in extraction procedure.

Building algorithm is similarly to Double Array Trie, when inserting a Trie node, each row of the child matrix is independently insert into base and check arrays using brute force strategy. The inser-

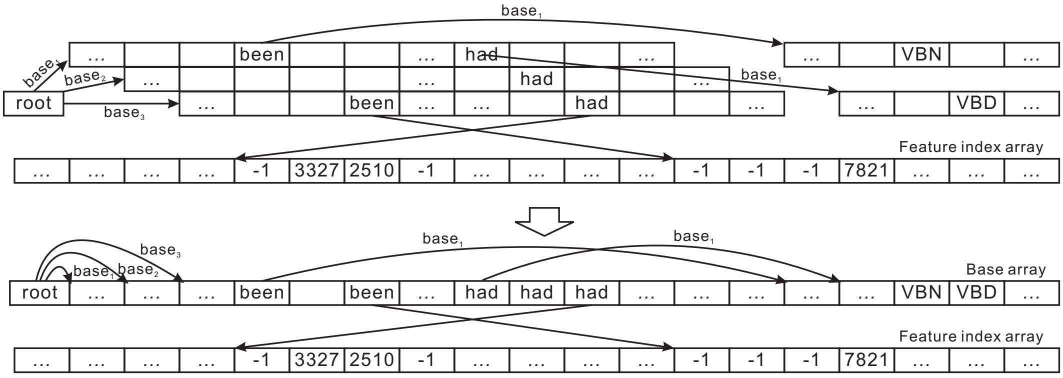


Figure 7: Build base array for 2D Trie in Figure 6. String in the box represents the key of the child. Blank boxes are the invalid children. The root node has 3 bases that point to three rows of the child matrix of vertex "p$_0$.*word*" respectively

```
Algorithm 1 Feature extraction using 2D Trie
Input: 2D Trie that stores features, template tree, template graph, a table storing token indexes, parent and child positions
Output: Feature index set $S$ of dependency from parent to child.
Create template vertex queue $Q_1$ and Trie node queue $Q_2$. Push roots of template tree and Trie into $Q_1, Q_2$ respectively. $S = \emptyset$
while $Q_1$ is not empty, do
    Pop a template vertex $T$ from $Q_1$ and a Trie node $N$ from $Q_2$. Get token index $j$ from index table according to $T$.
    for $i = 1$ to child number of $T$
        if child of $N$ at row $i$ column $j$ is valid,
            push it into $Q_2$ and push the $i^{th}$ child of $T$ into $Q_1$.
        else
            remove decedents of $i^{th}$ child of $T$ from template tree
        end if
    end for
    if $T$ is not virtual and the last child of $N$ in column $j$ is valid
            Enumerate dependency types, add valid feature indexes to $S$
    end if
end while
Return $S$.
```

tion repeats recursively until all features stored.

4 Complexity analysis

Let

- $|T|$ = number of templates
- $|t|$ = number of template units
- $|V|$ = number of vertexes in template tree, i.e, $|t|+$ number of virtual vertexes
- $|F|$ = number of features
- l = length of sentence
- $|f|$ = average length of feature strings

The procedure of 2D Trie for feature extraction consists of 2 steps: tokens in string table are mapped to their indexes, then Algorithm 1 is carried out for all node pairs of sentence. In the first step, we use double array Trie for efficient mapping. In fact, time spent is trivial compared with step 2 even by binary search. The main time spent of Algorithm 1 is the traversal of the whole template tree, in the worst case, no vertexes removed, so the time complexity of a sentence is $l^2|V|$, which is proportional to $|V|$. In other words, minimizing the number of virtual vertexes is important for efficiency.

For other indexing structures, feature generation is a primary step of retrieval. For each node

Structure	Generation	Retrieval
2D Trie	$l^2\|V\|$	
Hash / Trie	$l^2\|t\|$	$l^2\|f\|\|T\|$
Binary Search	$l^2\|t\|$	$l^2\|T\|\log\|F\|$

Table 3: Time complexity of different indexing structures.

pair of sentence, $|t|$ template units are processed, including concatenations of tokens and split symbols (split tokens in feature strings), boundary check (e.g, p$_{-1}$.*word* is out of boundary for beginning node of sentence). Thus the generation requires $l^2|t|$ processes. Notice that, time spent of each process varies on the length of tokens.

For feature string s with length $|s|$, if perfect hashing technique is adopted for index retrieval, it takes $|s|$ calculations to get hash value and a string comparison to check the string at the calculated position. So the time complexity is proportional to $|s|$, which is the same as Trie. Hence the total time for a sentence is $l^2|f||T|$. If binary search is used instead, $\log|F|$ string comparisons are required, complexity for a sentence is $l^2|T|\log|F|$.

Time complexity of these structures is summarized in Table 3.

5 Experiments

5.1 Experimental settings

We use Chinese Tree Bank 6.0 corpus for evaluation. The constituency structures are converted to dependency trees by Penn2Malt [2] toolkit and the standard training/development/test split is used. 257 sentences that failed in the conversion were removed, yielding 23316 sentences for training, 2060 sentences for development and 2660 sentences for testing respectively.

Since all the dependency trees are projective, a first order projective MST parser is naturally adopted. Online Passive Aggressive algorithm (Crammer et al., 2006) is used for fast training, 2 parameters, i.e, iteration number and C, are tuned on development data. The quality of the parser is measured by the labeled attachment score (LAS), i.e., the percentage of tokens with correct head and dependency type.

Group	IDs	#Temp.	#Vert.	#Feat.	LAS
1	1-2	72	91	$3.23M$	79.55%
2	1-3	128	155	$10.4M$	81.38%
3	1-4	240	275	$25.0M$	81.97%
4	1-5	332	367	$34.8M$	82.44%

Table 5: Parsing accuracy and number of templates, vertexes in template tree, features in decoding stage (zero weighted features are excluded) of each group.

We compare the proposed structure with Trie and binary search. We do not compare with perfect hashing, because it has the same complexity as Trie, and is often used for large data base retrieval, since it requires only one IO operation. For easy comparison, all feature indexing structures and the parser are implemented with C++. All experiments are carried out on a 64bit linux platform (CPU: Intel(R) Xeon(R) E5405, 2.00GHz, Memory: 16G Bytes). For each template set, we run the parser five times on test data and the averaged parsing time is reported.

5.2 Parsing speed comparison

To investigate the scalability of our method, rich templates are designed to generate large feature sets, as shown in Table 4. All templates are organized into 4 groups. Each row of Table 5 shows the details of a group, including parsing accuracy and number of templates, vertexes in template tree, and features in decoding stage (zero weighted features are excluded).

There is a rough trend that parsing accuracy increases as more templates used. Though such trend is not completely correct, the clear conclusion is that, abundant templates are necessary for accurate parsing.

Though algorithm described in section 2.3 for minimizing the number of virtual vertexes is heuristic, empirical results are satisfactory, number of newly created vertexes is only 10% as original templates. The reason is that complex templates are often extended from simple ones, their differences are often one or two template units.

Results of parsing time comparison are shown in Table 6. We can see that though time complexity of dynamic programming is cubic, parsing time of all systems is consistently dominated

[2] http://w3.msi.vxu.se/ nivre/research/Penn2Malt.html

ID	Templates			
1	$\mathrm{p}_i.\mathit{word}$	$\mathrm{p}_i.\mathit{pos}$	$\mathrm{p}_i.\mathit{word}+\mathrm{p}_i.\mathit{pos}$	
	$\mathrm{c}_i.\mathit{word}$	$\mathrm{c}_i.\mathit{pos}$	$\mathrm{c}_i.\mathit{word}+\mathrm{c}_i.\mathit{pos}$	$(\vert i\vert \leq 2)$
	$\mathrm{p}_i.\mathit{length}$	$\mathrm{p}_i.\mathit{length}+\mathrm{p}_i.\mathit{pos}$		
	$\mathrm{c}_i.\mathit{length}$	$\mathrm{c}_i.\mathit{length}+\mathrm{c}_i.\mathit{pos}$		$(\vert i\vert \leq 1)$
	$\mathrm{p}_0.\mathit{length}+\mathrm{c}_0.\mathit{length}\vert^l_d$	$\mathrm{p}_0.\mathit{length}+\mathrm{c}_0.\mathit{length}+\mathrm{c}_0.\mathit{pos}\vert^l_d$	$\mathrm{p}_0.\mathit{length}+\mathrm{p}_0.\mathit{pos}+\mathrm{c}_0.\mathit{length}\vert^l_d$	
	$\mathrm{p}_0.\mathit{length}+\mathrm{p}_0.\mathit{pos}+\mathrm{c}_0.\mathit{pos}\vert^l_d$	$\mathrm{p}_0.\mathit{pos}+\mathrm{c}_0.\mathit{length}+\mathrm{c}_0.\mathit{pos}\vert^l_d$	$\mathrm{p}_0.\mathit{length}+\mathrm{p}_0.\mathit{pos}+\mathrm{c}_0.\mathit{length}+\mathrm{c}_0.\mathit{pos}\vert^l_d$	
	$\mathrm{p}_i.\mathit{length}+\mathrm{p}_j.\mathit{length}+\mathrm{c}_k.\mathit{length}+\mathrm{c}_m.\mathit{length}\vert^l_d$			$(\vert i\vert + \vert j\vert + \vert k\vert + \vert m\vert \leq 2)$
	$\mathrm{r}_0.\mathit{word}$	$\mathrm{r}_{-1}.\mathit{word}+\mathrm{r}_0.\mathit{word}$	$\mathrm{r}_0.\mathit{word}+\mathrm{r}_1.\mathit{word}$	
	$\mathrm{r}_0.\mathit{pos}$	$\mathrm{r}_{-1}.\mathit{pos}+\mathrm{r}_0.\mathit{pos}$	$\mathrm{r}_0.\mathit{pos}+\mathrm{r}_1.\mathit{pos}$	
2	$\mathrm{p}_i.\mathit{pos}+\mathrm{c}_j.\mathit{pos}\vert_d$	$\mathrm{p}_i.\mathit{word}+\mathrm{c}_j.\mathit{word}\vert_d$	$\mathrm{p}_i.\mathit{pos}+\mathrm{c}_j.\mathit{word}+\mathrm{c}_j.\mathit{pos}\vert_d$	
	$\mathrm{p}_i.\mathit{word}+\mathrm{p}_i.\mathit{pos}+\mathrm{c}_j.\mathit{pos}\vert_d$	$\mathrm{p}_i.\mathit{word}+\mathrm{p}_i.\mathit{pos}+\mathrm{c}_j.\mathit{word}\vert_d$	$\mathrm{p}_i.\mathit{word}+\mathrm{c}_j.\mathit{word}+\mathrm{c}_j.\mathit{pos}\vert_d$	
	$\mathrm{p}_i.\mathit{word}+\mathrm{p}_i.\mathit{pos}+\mathrm{c}_j.\mathit{word}+\mathrm{c}_j.\mathit{pos}\vert_d$			$(\vert i\vert + \vert j\vert = 0)$
	Conjoin templates in the row above with $\vert^l$			
3	Similar with 2 $\vert i\vert + \vert j\vert = 1$			
4	Similar with 2 $\vert i\vert + \vert j\vert = 2$			
5	$\mathrm{p}_i.\mathit{word}$ + $\mathrm{p}_j.\mathit{word}$ + $\mathrm{c}_k.\mathit{word}\vert_d$	$\mathrm{p}_i.\mathit{word}$ + $\mathrm{c}_j.\mathit{word}$ + $\mathrm{c}_k.\mathit{word}\vert_d$		
	$\mathrm{p}_i.\mathit{pos}$ + $\mathrm{p}_j.\mathit{pos}$ + $\mathrm{c}_k.\mathit{pos}\vert_d$	$\mathrm{p}_i.\mathit{pos}$ + $\mathrm{c}_j.\mathit{pos}$ + $\mathrm{c}_k.\mathit{pos}\vert_d$		$(\vert i\vert + \vert j\vert + \vert k\vert \leq 2)$
	Conjoin templates in the row above with $\vert^l$			
	$\mathrm{p}_i.\mathit{word}$ + $\mathrm{p}_j.\mathit{word}$ + $\mathrm{p}_k.\mathit{word}$ + $\mathrm{c}_m.\mathit{word}\vert_d$		$\mathrm{p}_i.\mathit{word}$ + $\mathrm{p}_j.\mathit{word}$ + $\mathrm{c}_k.\mathit{word}$ + $\mathrm{c}_m.\mathit{word}\vert_d$	
	$\mathrm{p}_i.\mathit{word}$ + $\mathrm{c}_j.\mathit{word}$ + $\mathrm{c}_k.\mathit{word}$ + $\mathrm{c}_m.\mathit{word}\vert_d$			
	$\mathrm{p}_i.\mathit{pos}$ + $\mathrm{p}_j.\mathit{pos}$ + $\mathrm{p}_k.\mathit{pos}$ + $\mathrm{c}_m.\mathit{pos}\vert_d$		$\mathrm{p}_i.\mathit{pos}$ + $\mathrm{p}_j.\mathit{pos}$ + $\mathrm{c}_k.\mathit{pos}$ + $\mathrm{c}_m.\mathit{pos}\vert_d$	
	$\mathrm{p}_i.\mathit{pos}$ + $\mathrm{c}_j.\mathit{pos}$ + $\mathrm{c}_k.\mathit{pos}$ + $\mathrm{c}_m.\mathit{pos}\vert_d$			$(\vert i\vert + \vert j\vert + \vert k\vert + \vert m\vert \leq 2)$
	Conjoin templates in the row above with $\vert^l$			

Table 4: Templates used in Chinese dependency parsing.

by feature extraction. When efficient indexing structure adopted, i.e, Trie or Hash, time index retrieval is greatly reduced, about 4-5 times faster than binary search. However, general structures search features independently, their results could not guide feature generation. Hence, feature generation is still time consuming. The reason is that processing each template unit includes a series of steps, much slower than one integer comparison in Trie search.

On the other hand, 2D Trie greatly reduces the number of feature generations by pruning the template graph. In fact, no string concatenation occurs when using 2D Trie, since all tokens are converted to indexes beforehand. The improvement is significant, 2D Trie is about 5 times faster than Trie on the largest feature set, yielding 13.4 sentences per second parsing speed, about 4.3 times faster.

Space requirement of 2D Trie is about 2.1 times as binary search, and 1.7 times as Trie. One possible reason is that column number of 2D Trie (e.g. size of words) is much larger than standard double array Trie, which has only 256 children, i.e, range of a byte. Therefore, inserting a 2D Trie node is more strict, yielding sparser double arrays.

5.3 Comparison against state-of-the-art

Recent works on dependency parsing speedup mainly focus on inference, such as expected linear time non-projective dependency parsing (Nivre, 2009), integer linear programming (ILP) for higher order non-projective parsing (Martins et al., 2009). They achieve 0.632 seconds per sentence over several languages. On the other hand, Goldberg and Elhadad proposed splitSVM (Goldberg and Elhadad, 2008) for fast low-degree polynomial kernel classifiers, and applied it to transition based parsing (Nivre, 2003). They achieve 53 sentences per second parsing speed on English corpus, which is faster than our results, since transition based parsing is linear time, while for graph based method, complexity of feature extraction is quadratic. Xavier Lluís et al. (Lluís et al., 2009) achieve 8.07 seconds per sentence speed on CoNLL09 (Hajič et al., 2009) Chinese Tree Bank test data with a second order graphic model. Bernd Bohnet (Bohnet, 2009) also uses second order model, and achieves 610 minutes on CoNLL09 English data (2399 sentences, 15.3 second per sentence). Although direct comparison of parsing time is difficult due to the differences in data, models, hardware and implementations,

Group	Structure	Total	Generation	Retrieval	Other	Memory	sent/sec
1	Trie	87.39	63.67	10.33	13.39	$402M$	30.44
	Binary Search	127.84	62.68	51.52	13.64	$340M$	20.81
	2D Trie	39.74	26.29		13.45	$700M$	66.94
2	Trie	264.21	205.19	39.74	19.28	$1.3G$	10.07
	Binary Search	430.23	212.50	198.72	19.01	$1.2G$	6.18
	2D Trie	72.81	53.95		18.86	$2.5G$	36.53
3	Trie	620.29	486.40	105.96	27.93	$3.2G$	4.29
	Binary Search	982.41	484.62	469.44	28.35	$2.9G$	2.71
	2D Trie	146.83	119.56		27.27	$5.9G$	18.12
4	Trie	854.04	677.32	139.70	37.02	$4.9G$	3.11
	Binary Search	1328.49	680.36	609.70	38.43	$4.1G$	2.00
	2D Trie	198.31	160.38		37.93	$8.6G$	13.41

Table 6: Parsing time of 2660 sentences (seconds) on a 64bit linux platform (CPU: Intel(R) Xeon(R) E5405, 2.00GHz, Memory: 16G Bytes). Title “Generation” and “Retrieval” are short for feature generation and feature index retrieval steps respectively.

System	sec/sent
(Martins et al., 2009)	0.63
(Goldberg and Elhadad, 2008)	0.019
(Lluís et al., 2009)	8.07
(Bohnet, 2009)	15.3
(Galley and Manning, 2009)	15.6
ours group1	0.015
ours group2	0.027
ours group3	0.055
ours group4	0.075

Table 7: Comparison against state of the art, direct comparison of parsing time is difficult due to the differences in data, models, hardware and implementations.

these results demonstrate that our structure can actually result in a very fast implementation of a parser. Moreover, our work is orthogonal to others, and could be used for other learning tasks.

6 Conclusion

We proposed 2D Trie, a novel feature indexing structure for fast template based feature extraction. The key insight is that feature strings generated by a template are prefixes of the features from its extended templates, hence indexes of searched features can be reused for further extraction. We applied 2D Trie to dependency parsing task, experimental results on CTB corpus demonstrate the advantages of our technique, about 5 times faster than traditional Trie structure, yielding parsing speed 4.3 times faster, while using only 1.7 times as much memory.

7 Acknowledgments

The author wishes to thank the anonymous reviewers for their helpful comments. This work was partially funded by 973 Program (2010CB327906), The National High Technology Research and Development Program of China (2009AA01A346), Shanghai Leading Academic Discipline Project (B114), Doctoral Fund of Ministry of Education of China (200802460066), and Shanghai Science and Technology Development Funds (08511500302).

References

Aoe, Jun’ichi. 1989. An efficient digital search algorithm by using a double-array structure. *IEEE Transactions on software andengineering*, 15(9):1066–1077.

Bohnet, Bernd. 2009. Efficient parsing of syntactic and semantic dependency structures. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 67–72, Boulder, Colorado, June. Association for Computational Linguistics.

Crammer, Koby, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. In *JMLR 2006*.

Galley, Michel and Christopher D. Manning. 2009. Quadratic-time dependency parsing for machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 773–781, Suntec, Singapore, August. Association for Computational Linguistics.

Goldberg, Yoav and Michael Elhadad. 2008. splitsvm: Fast, space-efficient, non-heuristic, polynomial kernel computation for nlp applications. In *Proceedings of ACL-08: HLT, Short Papers*, pages 237–240, Columbus, Ohio, June. Association for Computational Linguistics.

Hajič, Jan, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June. Association for Computational Linguistics.

Kazama, Jun'ichi and Kentaro Torisawa. A new perceptron algorithm for sequence labeling with non-local features. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 315–324.

Lluís, Xavier, Stefan Bott, and Lluís Màrquez. 2009. A second-order joint eisner model for syntactic and semantic dependency parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 79–84, Boulder, Colorado, June. Association for Computational Linguistics.

Martins, Andre, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 342–350, Suntec, Singapore, August. Association for Computational Linguistics.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 91–97. Association for Computational Linguistics.

Nivre, Joakim. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 11th International Conference on Parsing Techniques*, pages 149–160.

Nivre, Joakim. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore, August. Association for Computational Linguistics.

Palmer, Martha and Nianwen Xue. 2009. Adding semantic roles to the Chinese Treebank. *Natural Language Engineering*, 15(1):143–172.

Sha, Fei and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 134–141, May.

The Bag-of-Opinions Method for Review Rating Prediction from Sparse Text Patterns

Lizhen Qu
Max-Planck Institute
for Informatics
lqu@mpii.mpg.de

Georgiana Ifrim
Bioinformatics Research
Centre
ifrim@birc.au.dk

Gerhard Weikum
Max-Planck Institute
for Informatics
weikum@mpii.mpg.de

Abstract

The problem addressed in this paper is to predict a user's numeric rating in a product review from the text of the review. Unigram and n-gram representations of text are common choices in opinion mining. However, unigrams cannot capture important expressions like "could have been better", which are essential for prediction models of ratings. N-grams of words, on the other hand, capture such phrases, but typically occur too sparsely in the training set and thus fail to yield robust predictors. This paper overcomes the limitations of these two models, by introducing a novel kind of bag-of-opinions representation, where an opinion, within a review, consists of three components: a root word, a set of modifier words from the same sentence, and one or more negation words. Each opinion is assigned a numeric score which is learned, by ridge regression, from a large, domain-independent corpus of reviews. For the actual test case of a domain-dependent review, the review's rating is predicted by aggregating the scores of all opinions in the review and combining it with a domain-dependent unigram model. The paper presents a constrained ridge regression algorithm for learning opinion scores. Experiments show that the bag-of-opinions method outperforms prior state-of-the-art techniques for review rating prediction.

1 Introduction

1.1 Motivation

Opinion mining and sentiment analysis has become a hot research area (Pang and Lee, 2008). There is ample work on analyzing the sentiments of online-review communities where users comment on products (movies, books, consumer electronics, etc.), implicitly expressing their *opinion polarities* (positive, negative, neutral), and also provide numeric *ratings* of products (Titov and McDonald, 2008b; Lerman et al., 2009; Hu and Liu, 2004; Titov and McDonald, 2008a; Pang and Lee, 2005; Popescu and Etzioni, 2005a). Although ratings are more informative than polarities, most prior work focused on classifying text fragments (phrases, sentences, entire reviews) by polarity. However, a product receiving mostly 5-star reviews exhibits better customer purchase behavior compared to a product with mostly 4-star reviews. In this paper we address the learning and prediction of numerical ratings from review texts, and we model this as a metric *regression* problem over an appropriately defined feature space.

Formally, the input is a set of rated documents (i.e., reviews), $\{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i$ is a sequence of word-level unigrams $(w_1, ..., w_l)$ and $y_i \in \mathbb{R}$ is a rating. The goal is to learn a function $f(\mathbf{x})$ that maps the word vector $\mathbf{x}$ into a numerical rating $\hat{y}$, which indicates both the polarity and strength of the opinions expressed in a document.

Numerical review rating prediction is harder than classifying by polarity. Consider the following example from Amazon book reviews:

The organization of the book is hard to follow and the chapter titles are not very helpful, so going back and trying to find information is quite *difficult*.

We note that there are many subjective words (*hard, helpful, difficult*) modified by opinion modifiers such as (*very, quite*) and negation words like (*not*). For rating prediction, considering opinion modifiers is crucial; *very helpful* is a much stronger sentiment than *helpful*. Negation words also need attention. As pointed out by Liu and Seneff (2009) we cannot simply reverse the polarity. For example, if we assign a higher positive score to *very helpful* than to *helpful*, simply reversing the sign of the scores would incorrectly suggest that *not helpful* is less negative than *not very helpful*.

The widely used unigram (bag-of-words) model (Pang and Lee, 2005; Snyder and Barzilay, 2007; Goldberg and Zhu, 2006; Ganu et al., 2009) cannot properly capture phrase patterns. Consider the following example: *not so helpful* vs. *not so bad*. In a unigram-based regression model each unigram gets a weight indicating its polarity and strength. High positive/negative weights are strongly positive/negative clues. It is reasonable to assign a positive weight to *helpful* and a negative weight to *bad*. The fundamental problem of unigrams arises when assigning a weight to *not*. If *not* had a strongly negative weight, the positive weight of *helpful* would be strongly reduced while the negative weight of *bad* would be amplified (by combining weights). This clearly fails to capture the true intentions of the opinion phrases. The same problem holds for *so*, which is an intensifier that should keep the same sign as the word it modifies. We refer to this limitation of the unigram model as **polarity incoherence**.

A promising way of overcoming this weakness is to include *n-grams*, generalizing the bag-of-words model into a bag-of-phrases model (Baccianella et al., 2009; Pang and Lee, 2008). However, regression models over the feature space of all n-grams (for either fixed maximal n or variable-length phrases) are computationally expensive in their training phase. Moreover and most importantly for our setting, including n-grams in the model results in a very high dimensional feature space: many features will then occur only very rarely in the training data. Therefore, it is difficult if not impossible to reliably learn n-gram weights from limited-size training sets. We refer to this problem as the **n-gram sparsity bottleneck**. In our experiments we investigate the effect of using bigrams and variable-length ngrams for improving review rating prediction.

1.2 Contribution

To overcome the above limitations of unigram and n-gram features, we have developed a novel kind of *bag-of-opinions* model, which exploits domain-independent corpora of opinions (e.g., all Amazon reviews), but is finally applied for learning predictors on domain-specific reviews (e.g., movies as rated in IMDB or Rottentomatoes). A document is represented as a bag of opinions each of which has three components: a root word, a set of modifier words and one or more negation words. In the phrase *not very helpful*, the opinion root is *helpful*, one (of potentially many) opinion modifier(s) is *very*, and a negation word is *not*. We enforce polarity coherence by the design of a learnable function that assigns a score to an opinion.

Our approach generalizes the cumulative linear offset model (CLO) presented in (Liu and Seneff, 2009). The CLO model makes several restrictive assumptions, most notably, that all opinion scores within one document are the same as the overall document rating. This assumption does not hold in practice, not even in reviews with extremely positive/negative ratings. For example, in a 5-star Amazon review the phrases *most impressive book* and *it helps explain* should receive different scores. Otherwise, the later transfer step to different domains would yield poor predictions. Due to this restriction, CLO works well on particular types of reviews that have pro/con entries listing characteristic major opinions about the object under review. For settings with individual reviews whose texts do not exhibit any specific structure, the CLO model faces its limitations.

In our bag-of-opinions method, we address the learning of opinion scores as a constrained ridge regression problem. We consider the opinion scores in a given review to be drawn from an unknown probability distribution (so they do not have to be the same within a document). We estimate the review rating based on a set of statistics (e.g., expectation, variance, etc.) derived from the scores of opinions in a document. Thus, our method has a sound statistical foundation and can be applied to arbitrary reviews with mixed opinion polarities and strengths. We avoid the n-gram sparsity problem by the limited-size structured feature space of *(root,modifiers,negators)* opinions.

We treat domain-independent and domain-dependent opinions differently in our system. In the first step we learn a bag-of-opinions model on a large dataset of online reviews to obtain scores for domain-independent opinions. Since the polarity of opinions is not bound to a topic, one can learn opinion scores from a pooled corpus of reviews for various categories, e.g., movies, books, etc., and then use these scored opinions for predicting the ratings of reviews belonging to a particular category. In order to also capture domain-dependent information (possibly complementary to the opinion lexicon used for learning domain-independent opinions), we combine the bag-of-opinions model with an unigram model trained on the domain-dependent corpus. Since domain-dependent training is typically limited, we model it using unigram models rather than bag-of-opinions. By combining the two models, even if an opinion does not occur in the domain-dependent training set but it occurs in a test review, we can still accurately predict the review rating based on the globally learned opinion score. In some sense our combined learning scheme is similar to smoothing in standard learning techniques, where the estimate based on a limited training set is smoothed using a large background corpus (Zhai and Lafferty, 2004).

In summary, the contributions of this paper are the following:

1. We introduce the bag-of-opinions model, for capturing the influence of n-grams, but in a structured way with root words, modifiers, and negators, to avoid the explosion of the feature space caused by explicit n-gram models.

2. We develop a constrained ridge regression method for learning scores of opinions from domain-independent corpora of rated reviews.

3. For transferring the regression model to newly given domain-dependent applications, we derive a set of statistics over opinion scores in documents and use these as features, together with standard unigrams, for predicting the rating of a review.

4. Our experiments with Amazon reviews from different categories (books, movies, music) show that the bag-of-opinions method outperforms prior state-of-the-art techniques.

2 Bag-of-Opinions Model

In this section we first introduce the bag-of-opinions model, followed by the method for learning (domain-independent) model parameters. Then we show how we annotate opinions and how we adapt the model to domain-dependent data.

2.1 Model Representation

We model each document as a bag-of-opinions $\{op_k\}_{k=1}^{K}$, where the number of opinions K varies among documents. Each opinion op_k consists of an opinion root w_r, $r \in S_R$, a set of opinion modifiers $\{w_m\}_{m=1}^{M}$, $m \in S_M$ and a set of negation words $\{w_z\}_{z=1}^{Z}$, $z \in S_Z$, where the sets S_R, S_M, S_Z are component index sets of opinion roots, opinion modifiers and negation words respectively. The union of these sets forms a global component index set $S \in \mathbb{N}^d$, where d is the dimension of the index space. The opinion root determines the prior polarity of the opinion. Modifiers intensify or weaken the strength of the prior polarity. Negation words strongly reduce or reverse the prior polarity. For each opinion, the set of negation words consists of at most a negation valence shifter like *not* (Kennedy and Inkpen, 2006) and its intensifiers like capitalization of the valence shifter. Each opinion component is associated with a score. We assemble the scores of opinion elements into an opinion-score by using a score function. For example, in the opinion *not very helpful*, the opinion root *helpful* determines the prior polarity positive say with a score 0.9, the modifier *very* intensifies the polarity say with a score 0.5. The prior polarity is further strongly reduced by the negation word *not* with e.g., a score -1.2. Then we sum up the scores to get a score of 0.2 for the opinion *not very helpful*.

Formally, we define the function $score(op)$ as a linear function of opinion components, which takes the form

$$\begin{aligned} score(op) &= sign(r)\beta_r x_r \\ &+ \sum_{m=1}^{M} sign(r)\beta_m x_m \\ &+ \sum_{z=1}^{Z} sign(r)\beta_z x_z \end{aligned} \quad (1)$$

where $\{x_z, x_m, x_r\}$ are binary variables denoting the presence or absence of negation words, modifiers and opinion root. $\{\beta_z, \beta_m, \beta_r\}$ are weights of each opinion elements. $sign(r) : w_r \rightarrow \{-1, 1\}$ is the opinion polarity function of the opinion root w_r. It assigns a value 1/-1 if an opinion root is positive/negative. Due to the semantics of opinion elements, we have constraints that $\beta_r \geq 0$ and $\beta_z \leq 0$. The sign of β_m is determined in the learning phase, since we have no prior knowledge whether it intensifies or weakens the prior polarity.

Since a document is modeled as a bag-of-opinions, we can simply consider the expectation of opinion scores as the document rating. If we assume the scores are uniformly distributed, the prediction function is then $f(\mathbf{x}) = \frac{1}{K}\sum_{k=1}^{K} score(op_k)$ which assigns the average of opinion scores to the document **x**.

2.2 Learning Regression Parameters

We assume that we can identify the opinion roots and negation words from a subjectivity lexicon. In this work we use MPQA (Wilson et al., 2005). In addition, the lexicon provides the prior polarity of the opinion roots. In the training phase, we are given a set of documents with ratings $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$, and our goal is to find an optimal function f^* whose predictions $\{\hat{y}_i\}_{i=1}^{N}$ are as close as possibile to the original ratings $\{y_i\}_{i=1}^{N}$. Formally, we aim to minimize the following loss function:

$$L = \frac{1}{2N}\sum_{i=1}^{N}(f(\mathbf{x}_i) - y_i)^2 \quad (2)$$

where $f(\mathbf{x}_i)$ is modeled as the average score of opinions in review $\mathbf{x}_i$.

First, we rewrite $score(op)$ as the dot product $\langle \boldsymbol{\beta}, \mathbf{p}\rangle$ between a weight vector $\boldsymbol{\beta} = [\boldsymbol{\beta}_z, \boldsymbol{\beta}_m, \beta_r]$ and a feature vector $\mathbf{p} = [sign(r)\mathbf{x}_z, sign(r)\mathbf{x}_m, sign(r)x_r]$. In order to normalize the vectors, we rewrite the weight and feature vectors in the d dimensional vector space of all root words, modifiers and negation words. Then $\boldsymbol{\beta} = [..,\boldsymbol{\beta}_z, 0, .., \boldsymbol{\beta}_m, 0, .., \beta_r, 0..] \in R^d$ and $\mathbf{p} = [sign(r)\mathbf{x}_z, 0, .., sign(r)\mathbf{x}_m, 0, .., sign(r)x_r, ...] \in R^d$. The function $f(\mathbf{x_i})$ can then be written as the dot product $\langle \boldsymbol{\beta}, \mathbf{v_i}\rangle$, where $\mathbf{v_i} = \frac{1}{K_i}\sum_{k=1}^{K_i}\mathbf{p}_k$, with K_i the number of opinions in review $\mathbf{x_i}$. By using this feature representation, the learning problem is equivalent to:

$$\min_{\beta} \; L(\boldsymbol{\beta}) = \frac{1}{2N}\sum_{i=1}^{N}(\langle \boldsymbol{\beta}, \mathbf{v}_i\rangle + \beta_0 - y_i)^2$$

$s.t.$

$$\begin{aligned} &\beta_z \leq 0 \quad z \in S_Z \\ &\beta_r \geq 0 \quad r \in S_R \end{aligned} \quad (3)$$

where $\boldsymbol{\beta} \in R^d$, $\boldsymbol{\beta} = [\boldsymbol{\beta}_z, \boldsymbol{\beta}_m, \boldsymbol{\beta}_r]$. β_0 is the intercept of the regression function, which is estimated as the mean of the ratings in the training set. We define a new variable $\tilde{y}_i = y_i - \beta_0$.

In order to avoid overfitting, we add an $l2$ norm regularizer to the loss function with the parameter $\lambda > 0$.

$$LR(\boldsymbol{\beta}) = \frac{1}{2N}\sum_{i=1}^{N}(\langle \boldsymbol{\beta}, \mathbf{v}_i\rangle - \tilde{y}_i)^2 + \frac{\lambda}{2} \parallel \boldsymbol{\beta} \parallel_2^2$$

$s.t.$

$$\begin{aligned} &\beta_z \leq 0 \quad z \in S_Z \\ &\beta_r \geq 0 \quad r \in S_R \end{aligned} \quad (4)$$

We solve the above optimization problem by Algorithm 1 using coordinate descent. The procedure starts with $\boldsymbol{\beta}^0 = 0$, $\boldsymbol{\beta}^0 \in R^d$. Then it updates iteratively every coordinate of the vector $\boldsymbol{\beta}$ until convergence. Algorithm 1 updates every coordinate $\beta_j, j \in \{1, 2, ..., d\}$ of $\boldsymbol{\beta}$ by solving the following one-variable sub-problem:

$$min_{l_j \leq \beta_j \leq c_j} LR(\beta_1, ..., \beta_j, ..., \beta_d)$$

where l_j and c_j denote the lower and upper bounds of β_j. If $j \in S_Z$, $l_j = -\infty$ and $c_j = 0$. If $j \in S_R$, $l_j = 0$ and $c_j = \infty$. Otherwise both bounds are infinity.

According to (Luo and Tseng, 1992), the solution of this one-variable sub-problem is

$$\hat{\beta}_j = max\{l_j, min\{c_j, g_j\}\}$$

where

$$g_j = \frac{\frac{1}{N}\sum_{i=1}^{N} v_{ij}(\tilde{y}_i - \sum_{l \neq j} \beta_l v_l)}{\frac{1}{N}\sum_{i=1}^{N} v_{ij}^2 + \lambda}$$

Here g_j is the close form solution of standard ridge regression at coordinate j (for details see (Friedman et al., 2008)). We prove the convergence of Algorithm 1, by the following theorem using techniques in (Luo and Tseng, 1992).

Theorem 1 *A sequence of $\boldsymbol{\beta}$ generated by Algorithm 1 globally converges to an optimal solution $\beta^* \in \chi^*$ of problem (4), where χ^* is the set of optimal solutions.*

Proof: *Luo and Tseng (1992) show that coordinate descent for constrained quadratic functions in the following form converges to one of its global optimal solutions.*

$$\begin{array}{ll} min_{\boldsymbol{\beta}} & h(\boldsymbol{\beta}) = \langle \boldsymbol{\beta}, \mathbf{Q}\boldsymbol{\beta} \rangle / 2 + \langle \mathbf{q}, \boldsymbol{\beta} \rangle \\ s.t. & \mathbf{E}^T \boldsymbol{\beta} \geq \mathbf{b} \end{array}$$

where $\mathbf{Q}$ is a $d \times d$ symmetric positive-definite matrix, $\mathbf{E}$ is a $d \times d$ matrix having no zero column, $\mathbf{q}$ is a d-vector and $\mathbf{b}$ is a d-vector.

We rewrite LR in matrix form as

$$\begin{aligned} & \frac{1}{2N}(\tilde{\mathbf{y}} - \mathbf{V}\boldsymbol{\beta})^T(\tilde{\mathbf{y}} - \mathbf{V}\boldsymbol{\beta}) + \frac{\lambda}{2}\boldsymbol{\beta}^T\boldsymbol{\beta} \\ = & \frac{1}{2N}(\mathbf{V}\boldsymbol{\beta})^T(\mathbf{V}\boldsymbol{\beta}) + \frac{\lambda}{2}\boldsymbol{\beta}^T\boldsymbol{\beta} - \frac{1}{2N}((\mathbf{V}\boldsymbol{\beta})^T\tilde{\mathbf{y}} \\ & -\frac{1}{2N}\tilde{\mathbf{y}}^T(\mathbf{V}\boldsymbol{\beta})) + \frac{1}{2N}\tilde{\mathbf{y}}^T\tilde{\mathbf{y}} \\ = & \langle \boldsymbol{\beta}, \mathbf{Q}\boldsymbol{\beta} \rangle / 2 + \langle \mathbf{q}, \boldsymbol{\beta} \rangle + constant \end{aligned}$$

where

$$\mathbf{Q} = \mathbf{B}^T\mathbf{B}, \mathbf{B} = \begin{bmatrix} \sqrt{\frac{1}{N}}\mathbf{V} \\ \sqrt{\lambda}\mathbf{I}^{d \times d} \end{bmatrix}, \mathbf{q} = \frac{-1}{N}(\mathbf{V}^T\tilde{\mathbf{y}})$$

where $\mathbf{I}^{d \times d}$ is the identity matrix. Because $\lambda > 0$, all columns of $\mathbf{B}$ are linearly independent. As $\mathbf{Q} = \mathbf{B}^T\mathbf{B}$ and symmetric, $\mathbf{Q}$ is positive definite.

We define $\mathbf{E}$ as a $d \times d$ diagonal matrix with all entries on the main diagonal equal to 1 except $e_{ii} = -1, i \in S_Z$ and $\mathbf{b}$ is a d-vector with all entries equal to $-\infty$ except $b_i = 0$, for $i \in S_Z$ or $i \in S_R$.

Because the almost cyclic rule is applied to generate the sequence $\{\boldsymbol{\beta}^t\}$, the algorithm converges to a solution $\boldsymbol{\beta}^* \in \chi^*$.

Algorithm 1 Constrained Ridge Regression

```
Input: $\lambda$ and $\{\mathbf{v}_n, \tilde{y}_n\}_{n=1}^N$
Output: optimal $\boldsymbol{\beta}$
repeat
    for $j = 1, ..., d$ do
        $g_j = \frac{\frac{1}{N}\sum_{i=1}^{N} v_{ij}(\tilde{y}_i - \sum_{l \neq j} \beta_l v_l)}{\frac{1}{N}\sum_{i=1}^{N} v_{ij}^2 + \lambda}$
        $\hat{\beta}_j = \begin{cases} 0, & if\, j \in S_R\, and\, g_j < 0 \\ 0, & if\, j \in S_Z\, and\, g_j > 0 \\ g_j, & else \end{cases}$
    end for
until Convergence condition is satisfied
```

2.3 Annotating Opinions

The MPQA lexicon contains separate lexicons for subjectivity clues, intensifiers and valence shifters (Wilson et al., 2005), which are used for identifying opinion roots, modifiers and negation words. Opinion roots are identified as the positive and negative subjectivity clues in the subjectivity lexicon. In the same manner, intensifiers and valence shifters of the type {negation, shiftneg} are mapped to modifiers and negation words. Other modifier candidates are adverbs, conjunctions and modal verbs around opinion roots. We consider non-words modifiers as well, e.g., punctuations, capitalization and repetition of opinion roots. If the opinion root is a noun, adjectives are also included into modifier sets.

The automatic opinion annotation starts with locating the continous subjectivity clue sequence. Once we find such a sequence and at least one of the subjectivity clue is positive or negative, we search to the left up to 4 words for negation words and modifier candidates, and stop if encountering another opinion root. Similarly, we search to the right up to 3 unigrams for modifiers and stop if we find negation words or any other opinion roots. The prior polarity of the subjectivity sequence is determined by the polarity of the last subjectivity clue with either positive or negative polarity in the sequence. The other subjectivity clues in the same sequence are treated as modifiers.

2.4 Adaptation to Domain-Dependent Data

The adaptation of the learned (domain-independent) opinion scores to the target domain and the integration of domain-dependent unigrams is done in a second ridge-regression task. Note that this is a simpler problem than typical domain-adaptation, since we already know from the sentiment lexicon which are the domain-independent features. Additionally, its relatively easy to obtain a large mixed-domain corpus for reliable estimation of domain-independent opinion scores (e.g., use all Amazon product reviews). Furthermore, we need a domain-adaptation step since domain-dependent and domain-independent data have generally different rating distributions. The differences are mainly reflected in the intercept of the regression function (estimated as the mean of the ratings). This means that we need to scale the positive/negative mean of the opinion scores differently before using it for prediction on domain-dependent reviews. Moreover, other statistics further characterize the opinion score distribution. We use the variance of opinion scores to capture the reliability of the mean, multiplied by the negative sign of the mean to show how much it strengthens/weakens the estimation of the mean. The mean score of the dominant polarity (*major exp*) is also used to reduce the influence of outliers. Because positive and negative means should be scaled differently, we represent positive and negative values of the mean and *major exp* as 4 different features. Together with variance, they are the 5 statistics of the opinion score distribution. The second learning step on opinion score statistics and domain-dependent unigrams as features, re-weights the importance of domain-independent and domain-dependent information according to the target domain bias.

3 Experimental Setup

We performed experiments on three target domains of Amazon reviews: books, movies (DVDs), and music (CDs). For each domain, we use ca. 8000 Amazon reviews for evaluation; an additional set of ca. 4000 reviews are withheld for parameter tuning (regularization parameter, etc.). For learning weights for domain-independent opinions, we use a mixed-domain corpus of ca. 350,000 reviews from Amazon (electronics, books, dvds, etc.); this data is disjoint from the test sets and contains no reviews from the music domain. In order to learn unbiased scores, we select about the same number of positive and negative reviews (where reviews with more/less than 3 stars are regarded as positive/negative). The regularization parameters used for this corpus are tuned on withheld data with ca. 6000 thematically mixed reviews.[1].

We compare our method, subsequently referred to as *CRR-BoO* (Constrained Ridge Regression for Bag-of-Opinions), to a number of alternative state-of-the-art methods. These competitors are varied along two dimensions: 1) feature space, and 2) training set. Along the first dimension, we consider a) unigrams coined *uni*, b) unigrams and bigrams together, coined *uni+bi*, c) variable-length n-grams coined *n-gram*, d) the opinion model by (Liu and Seneff, 2009) coined *CLO* (cumulative linear offset model). As learning procedure, we use ridge regression for a), b), and d), and bounded cyclic regression, coined *BCR*, for c). Along the second - orthogonal - dimension, we consider 3 different training sets: i) domain-dependent training set coined *DD*, ii) the large mixed-domain training set coined *MD*, iii) domain-dependent training set and the large mixed-domain training set coined *DD+MD*. For the *DD+MD* training set, we apply our two stage approach for *CRR-BoO* and *CLO*, i.e., we use the mixed-domain corpus for learning the opinion scores in the first stage, and integrate unigrams from *DD* in a second domain-adaptation stage. We train the remaining feature models directly on the combination of the whole mixed-domain cor-

[1] All datasets are available from `http://www.mpi-inf.mpg.de/∼lqu`

feature models		uni	uni+bi	n-gram	CLO	**CRR-BoO**
DD	book	1.004	0.961	0.997	1.469	0.942
	dvd	1.062	1.018	1.054	1.554	0.946
	music	0.686	0.672	0.683	0.870	0.638
MD	book	1.696	1.446	1.643	1.714	1.427
	dvd	1.919	1.703	1.858	1.890	1.565
	music	2.395	2.160	2.340	2.301	1.731
DD+MD	book	1.649	1.403	1.611	1.032	**0.884**
	dvd	1.592	1.389	1.533	1.086	**0.928**
	music	1.471	1.281	1.398	0.698	**0.627**

Table 1: Mean squared error for rating prediction methods on Amazon reviews.

pus and the training part of *DD*.

The CLO model is adapted as follows. Since bags-of-opinions generalize CLO, adjectives and adverbs are mapped to opinion roots and modifiers, respectively; negation words are treated the same as CLO. Subsequently we use our regression technique. As Amazon reviews do not contain pro and con entries, we learn from the entire review.

For BCR, we adapt the variable-length n-grams method of (Ifrim et al., 2008) to elastic-net-regression (Friedman et al., 2008) in order to obtain a fast regularized regression algorithm for variable-length n-grams. We search for significant n-grams by incremental expansion in backward direction (e.g., expand *bad* to *not bad*). BCR pursues a dense solution for unigrams and a sparse solution for n-grams. Further details on the BCR learning algorithm will be found on a subsequent technical report.

As for the regression techniques, we show only results with ridge regression (for all feature and training options except BCR). It outperformed ϵ-*support vector regression* (SVR) of libsvm (Chang and Lin, 2001), lasso (Tibshirani, 1996), and elastic net (Zou and Hastie, 2005) in our experiments.

4 Results and Discussion

Table 1 shows the mean square error (MSE) from each of the three domain-specific test sets. The error is defined as $MSE = \frac{1}{N}\sum_{i=1}^{N}(f(\mathbf{x}_i) - y_i)^2$. The right most two columns of the table show results for the full-fledge two-stage learning for our method and CLO, with domain-dependent weight learning and the domain adaptation step. The other models are trained directly on the given training sets. For the DD and DD+MD training sets, we use five-fold cross-validation on the domain-specific sets. For the MD training set, we take the domain-specific test sets as hold-out data for evaluation.

Table 1 clearly shows that our *CRR-BoO* method outperforms all alternative methods by a significant margin. Most noteworthy is the music domain, which is not covered by the mixed-domain corpus. As expected, unigrams only perform poorly, and adding bigrams leads only to marginal improvements. BCR pursues a dense solution for unigrams and a sparse solution for variable-length n-grams, but due to the sparsity of occurence of long n-grams, it filters out many interesting-but-infrequent ngrams and therefore performs worse than the dense solution of the *uni+bi* model. The CLO method of (Liu and Seneff, 2009) shows unexpectedly poor performance. Its main limitation is the assumption that opinion scores are identical within one document. This does not hold in documents with mixed opinion polarities. It also results in conflicts for opinion components that occur in both positive and negative documents. In contrast, *CRR-BoO* naturally captures the mixture of opinions as a bag of positive/negative scores. We only require that the mean of opinion scores equals the overall document rating.

The right most column of Table 1 shows that our method can be improved by learning opinion scores from the large mixed-domain corpus. How-

opinion	score
good	0.18
recommend	1.64
most difficult	-1.66
but it gets very good!	2.37
would highly recommend	2.73
would not recommend	-1.93

Table 2: Example opinions learned from the Amazon mixed-domain corpus.

ever, the high error rates of the models learned directly on the MD corpus show that direct training on the mixed-domain data can introduce a significant amount of noise into the prediction models. Although the noise can be reduced by learning from MD and DD together, the performance is still worse than when learning directly from the domain-dependent corpora. Additionally, when the domain is not covered by the mixed-domain corpus (e.g., music), the results are even worse. Thus, the two stages of our method (learning domain-independent opinion scores plus domain-adaptation) are decisive for a good performance, and the sentiment-lexicon-based BoO model leads to robust learning of domain-independent opinion scores.

Another useful property of BoO is its high interpretability. Table 2 shows example opinion scores learned from the mixed-domain corpus. We observe that the scores corelate well with our intuitive interpretation of opinions.

Our *CRR-BoO* method is highly scalable. Excluding the preprocessing steps (same for all methods), the learning of opinion component weights from the ca. 350,000 domain-independent reviews takes only 11 seconds.

5 Related Work

Rating prediction is modeled as an ordinal regression problem in (Pang and Lee, 2005; Goldberg and Zhu, 2006; Snyder and Barzilay, 2007). They simply use the bag-of-words model with regression algorithms, but as seen previously this cannot capture the expressive power of phrases. The resulting models are not highly interpretable. Baccianella et al. (2009) restrict the n-grams to the ones having certain POS patterns. However, the long n-grams matching the patterns still suffer from sparsity. The same seems to hold for sparse n-gram models (BCR in this paper) in the spirit of Ifrim et al. (2008). Although sparse n-gram models can explore arbitrarily large n-gram feature spaces, they can be of little help if the n-grams of interests occur sparsely in the datasets.

Since our approach can be regarded as learning a domain-independent sentiment lexicon, it is related to the area of automatically building domain-independent sentiment lexicons (Esuli and Sebastiani, 2006; Godbole et al., 2007; Kim and Hovy, 2004). However, this prior work focused mainly on the opinion polarity of opinion words, neglecting the opinion strength. Recently, the lexicon based approaches were extended to learn domain-dependent lexicons (Kanayama and Nasukawa, 2006; Qiu et al., 2009), but these approaches also neglect the aspect of opinion strength. Our method requires only the prior polarity of opinion roots and can thus be used on top of those methods for learning the scores of domain-dependent opinion components. The methods proposed in (Hu and Liu, 2004; Popescu and Etzioni, 2005b) can also be categorized into the lexicon based framework because their procedure starts with a set of seed words whose polarities are propagated to other opinion bearing words.

6 Conclusion and Future Work

In this paper we show that the bag-of-opinions (BoO) representation is better suited for capturing the expressive power of n-grams while at the same time overcoming their sparsity bottleneck. Although in this paper we use the BoO representation to model domain-independent opinions, we believe the same framework can be extended to domain-dependent opinions and other NLP applications which can benefit from modelling n-grams (given that the n-grams are decomposable in some way). Moreover, the learned model can be regarded as a domain-independent opinion lexicon with each entry in the lexicon having an associated score indicating its polarity and strength. This in turn has potential applications in sentiment summarization, opinionated information retrieval and opinion extraction.

References

Baccianella, S., A. Esuli, and F. Sebastiani. 2009. Multi-facet rating of product reviews. In *ECIR*. Springer.

Chang, C.C. and C.J. Lin, 2001. *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Esuli, A. and F. Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *LREC*, pages 417–422.

Friedman, J., T. Hastie, and R. Tibshirani. 2008. Regularization paths for generalized linear models via coordinate descent. Technical report, Technical Report, Available at http://www-stat. stanford. edu/jhf/ftp/glmnet. pdf.

Ganu, G., N. Elhadad, and A. Marian. 2009. Beyond the stars: Improving rating predictions using review text content. In 12^{th} *International Workshop on the Web and Databases*.

Godbole, Namrata, Manjunath Srinivasaiah, and Steven Skiena. 2007. Large-scale sentiment analysis for news and blogs. In *ICWSM*.

Goldberg, A. B. and X.J. Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *HLT-NAACL 2006 Workshop on Textgraphs: Graph-based Algorithms for Natural Language Processing*.

Hu, M.Q. and B. Liu. 2004. Mining and summarizing customer reviews. In *CIKM*, pages 168–177. ACM New York,USA.

Ifrim, G., G. Bakir, and G. Weikum. 2008. Fast logistic regression for text categorization with variable-length n-grams. In *KDD*, pages 354–362, New York,USA. ACM.

Kanayama, H. and T. Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *EMNLP*, pages 355–363.

Kennedy, A. and D. Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110–125.

Kim, S.M. and E. Hovy. 2004. Determining the sentiment of opinions. In *COLING*, pages 1367–1373.

Lerman, K., S. Blair-Goldensohn, and R. McDonald. 2009. Sentiment summarization: Evaluating and learning user preferences. In *EACL*, pages 514–522. ACL.

Liu, J.J. and S. Seneff. 2009. Review Sentiment Scoring via a Parse-and-Paraphrase Paradigm. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 161–169. ACL.

Luo, Z.Q. and Q. Tseng. 1992. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35.

Pang, B. and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, page 124. ACL.

Pang, B. and L. Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Popescu, A.M. and O. Etzioni. 2005a. Extracting product features and opinions from reviews. In *HLT/EMNLP*, volume 5, pages 339–346. Springer.

Popescu, A.M. and O. Etzioni. 2005b. Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP*, volume 5, pages 339–346. Springer.

Qiu, G., B. Liu, J.J. Bu, and C. Chen. 2009. Expanding Domain Sentiment Lexicon through Double Propagation. In *IJCAI*.

Snyder, B. and R. Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *NAACL/HLT*, pages 300–307.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288.

Titov, I. and R. McDonald. 2008a. A joint model of text and aspect ratings for sentiment summarization. In *HLT/ACL*, pages 308–316.

Titov, I. and R. McDonald. 2008b. Modeling online reviews with multi-grain topic models. In *WWW*, pages 111–120. ACM.

Wilson, T., J. Wiebe, and P. Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/ACL*, pages 347–354.

Zhai, C. X. and J. Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214.

Zou, H. and T. Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B(Statistical Methodology)*, 67(2):301–320.

An Exploration of Features for Recognizing Word Emotion

Changqin Quan
Faculty of Engineering
University of Tokushima
quan-c@is.tokushima-u.ac.jp

Fuji Ren
Faculty of Engineering
University of Tokushima
ren@is.tokushima-u.ac.jp

Abstract

Emotion words have been well used as the most obvious choice as feature in the task of textual emotion recognition and automatic emotion lexicon construction. In this work, we explore features for recognizing word emotion. Based on Ren-CECps (an annotated emotion corpus) and MaxEnt (Maximum entropy) model, several contextual features and their combination have been experimented. Then PLSA (probabilistic latent semantic analysis) is used to get semantic feature by clustering words and sentences. The experimental results demonstrate the effectiveness of using semantic feature for word emotion recognition. After that, "word emotion components" is proposed to describe the combined basic emotions in a word. A significant performance improvement over contextual and semantic features was observed after adding word emotion components as feature.

1 Introduction

Textual emotion analysis is becoming increasingly important due to augmented communication via computer mediated communication (CMC). A possible application of textual emotion recognition is online chat system. An emotion feedback system can recognize users' emotion and give appropriate responses. Another application example is weblog emotion recognition and prediction. Blogspace consists of millions of users who maintain their online diaries, containing frequently-updated views and personal remarks about a range of issues. An emotion recognition and prediction system can understand the public's reaction to some social issues and predict emotion changes. It would be helpful for solving some psychological problems or giving early warnings, such as suicide or terrorism.

Textual emotion analysis also can improve the accuracy of other nonverbal modalities like speech or facial emotion recognition, and to improve human computer interaction systems. However, automatic recognition of emotion meaning from texts presents a great challenge. One of the reasons is the manifoldness of expressed emotions in words.

Emotion words have been well used as the most obvious choice as feature in the task of textual emotion recognition and automatic emotion lexicon construction (Virginia and Pablo, 2006; Tokuhisa et al., 2008, etc.). And there are many lexical resources developed for these tasks, such as GI (Stone et al., 1966), WordNet-Affect (Strapparava and Valitutti, 2004), NTU Sentiment Dictionary (Ku et al., 2006), Hownet (Dong and Dong, 2003), SentiWordnet (Esuli and Sebastiani, 2006). In these sentimental or affective lexicons, the words usually bear direct emotions or opinions, such as happy or sad, good or bad. Although they play a role in some applications, several problems of emotion expression in words have been ignored.

Firstly, there are a lot of sentences can evoke emotions without direct emotion words. For example,

(1) 春天在孩子们的眼里、在孩子们的心里。(Spring is in children's eyes, and in their hearts.)

In sentence (1), we may feel joy, love or expect delivered by the writer. But there are no direct emotion words can be found from lexicons. As Ortony (1987) indicates, besides words directly referring to emotion states (e.g., "fear", "cheerful") and for which an appropriate lexicon would help, there are words that act only as an indirect reference to emotions depending on the context. Strapparava et al. (2006) also address this issue. The authors believed that all words can potentially convey affective meaning, and they distinguished between words directly referring to emotion states (direct affective words) and those having only an indirect reference that depends on the context (indirect affective words).

The second problem is emotion ambiguity of words. The same word in different contexts may reflect different emotions. For example,

(2) 这是目前我**唯一**能做的。(This is currently the only thing I can do.)

(3) 他是我的**唯一**。(He is my only one.)

In sentence (2), the word "唯一 (only)" may express the emotion of anxiety or expect; but in sentence (3), the word "唯一 (only)" may express the emotion of love or expect. The emotion categories can not be determined without their certain contexts especially for the words with emotion ambiguity.

In addition, some words can express multiple emotions, such as "悲喜交加 (mingled feelings of joy and sorrow)". Statistics on an annotated emotion corpus (Ren-CECps [1], Chinese emotion corpus developed by Ren-lab) showed that 84.9% of all emotion words have one emotion, 15.1% have more than one emotions (Quan and Ren, 2010). Multi-emotion words are indispensable for expressing complex feelings in use of language.

In this work, we explore features for recognizing word emotion in sentences. Based on Ren-CECps and MaxEnt model, several contextual features and their combination have been experimented. Then PLSA (probabilistic latent semantic analysis) is used to get semantic feature by clustering word and sentence. The experimental results demonstrate the effectiveness of using semantic feature for word emotion recognition. After that, the notion of "word emotion components" is proposed to describe the combined basic emotions in a word. A significant performance improvement over only using contextual and semantic features was observed after adding word emotion components as feature and output in MaxEnt based model.

This paper is organized as follows. In section 2, based on Ren-CECps and MaxEnt, an exploration of using contextual feature for Chinese word emotion recognition is described. In section 3, using PLSA technique, the performance of adding semantic feature is presented. In section 4, the notion of "word emotion components" is proposed and the performance of using encoding feature is presented. In section 5, the discussions are described. Section 6 is conclusions.

2 Chinese Word Emotion Recognition

2.1 Related Works

There are many researches concerning computing semantics of words, while the researches on computing emotions of words are relatively less. Computing word emotions is a challenge task because the inherent of emotion is ambiguous and natural language is very rich in emotion terminology. Using the textual emotion information, several methods have been explored for computing lexical emotions. Wilson et al. (2009) proposed a two-step approach to classify word polarity out of context firstly, and then to classify word polarity in context with a wide variety of features. Strapparava et al. (2007) implemented a variation of Latent Semantic Analysis (LSA) to measure the similarities between direct affective terms and generic terms. Lee and Narayanan (2005) proposed a method of computing mutual information between a specific word and emotion category to measure how much information a word provides about a given emotion category (emotion salience). Based on structural similarity, Bhowmick (2008) computed the structural similarity of words in WordNet to distinguish the emotion words from the non-emotion words. Kazemzadeh (2008) measured similarity between word and emotion category based on interval type-2 fuzzy logic method. Takamura (2005) used a spin model to extract emotion polarity of words.

Different from the above researches, in this work, we explore which features are effective for word emotion recognition. The features include contextual feature, semantic feature and encoding feature.

[1] http://a1-www.is.tokushima-u.ac.jp/member /ren/Ren-CECps1.0/Ren-CECps1.0.html

2.2 Ren-CECps and MaxEnt based Chinese Word Emotion Recognition

Ren-CECps is constructed based on a relative fine-grained annotation scheme, annotating emotion in text at three levels: document, paragraph, and sentence. The all dataset consisted of 1,487 blog articles published at sina blog, sciencenet blog, etc. There are 11,255 paragraphs, 35,096 sentences, and 878,164 Chinese words contained in this corpus (more details can be found in (Quan and Ren, 2010)).

In the emotion word annotation scheme of Ren-CECps, direct emotion words and indirect emotion words in a sentence are all annotated. For example, in sentence (1) “春天 (spring)” and “孩子们 (the children)” are labeled. An emotion keyword or phrase is represented as a vector to record its intensities of the eight basic emotion classes (expect, joy, love, surprise, anxiety, sorrow, angry and hate). For instance, the emotion vector for the word “春天 (spring)” $\overrightarrow{w} = (0.1, 0.3, 0.3, 0.0, 0.0, 0.0, 0.0, 0.0)$ indicates the emotions of weak expect, joy and love. In this work, we focus on if a word contains some emotion(s) in a certain context. The analysis on emotion intensity of emotion words is included in our future work.

As word emotion is subjective entity, a word in a certain context may evoke multiple emotions in different people’s mind. A part of documents in Ren-CECps have been annotated by three annotators independently to measure agreement on the annotation of this corpus, which include 26 documents with a total of 805 sentences, 19,738 words. This part of corpus is used as testing corpus to evaluate the experimental results. (Section 5.1 shows the analysis on the annotation agreement on word emotion.)

MaxEnt modeling provides a framework for integrating information from many heterogeneous information sources for classification (Manning, 1999). MaxEnt principle is a well used technique provides probability of belongingness of a token to a class. In word emotion recognition, the MaxEnt estimation process produces a model in which each feature f_i is assigned a weight α_i. The deterministic model produces conditional probability (Berger, 1996), see equation (1) and (2). In experiments, we have used a Java based open-nlp MaxEnt toolkit [2].

$$p(e|context) = \frac{1}{Z(context)} \prod_i \alpha_i^{f_i(context,e)} \quad (1)$$

$$Z(context) = \sum \prod_i \alpha_i^{f_i(context,e)} \quad (2)$$

2.3 Contextual Features

The contextual features used in MaxEnt for Chinese word emotion recognition are described as follows:

Word Feature (WF): Word itself to be recognized.

N-words Feature (NF): To know the relationship between word emotion and its context, the surrounding words of length n for the word (w_i) to be recognized are used as feature: $(w_{i-n}...w_i...w_{i+n})$.

POS Feature (POSF): The part of speech of the current word and surrounding words are used as feature. We have used a Chinese segmentation and POS tagger (Ren-CMAS) developed by Ren-lab, which has an accuracy about 97%. The set of POS includes 35 classes.

Pre-N-words Emotion Feature (PNEF): The emotions of the current word may be influenced by the emotions of its previous words. So the emotions of previous n words are used as feature. The value of this feature for a word (w_i) is obtained only after the computation of the emotions for its previous words.

Pre-is-degree-word Feature (PDF), Pre-is-negative-word Feature (PNF), Pre-is-conjunction Feature (PCF): To determine if the previous word is a degree word, a negative word, or a conjunction may be helpful to identify word emotions. The degree word list (contains 1,039 words), negative word list (contains 645 words), and conjunction list (contains 297 words) extracted from Ren-CECps have been used.

2.4 The Performance of Using Contextual Feature

We use the documents in Ren-CECps that have been annotated by three annotators independently

[2] http://maxent.sourceforge.net/

as testing corpus. An output of word emotion(s) will be regarded as a correct result if it is in agreement with any one item of word emotion(s) provided by the three annotators. The numbers of training and testing corpus are shown in table 1. The accuracies are measured by F-value.

Table 1: Number of training and testing corpus

Number	Training	Testing
Documents	1,450	26
Sentences	33,825	805
Words	813,507	19,738
Emotion words	99,571	$2,271^*$

(*) At least agreed by two annotators.

Table 2 gives the results of F-value for different contextual features in the MaxEnt based Chinese word emotion recognition. The results of F-value include: (a) recognize emotion and unemotion words; (b) recognize the eight basic emotions for emotion words (complete matching); (c) recognize the eight basic emotions for emotion words (single emotion matching).

As shown in table 2, when we only use Word Feature(WF), the F-value of task (a) achieved a high value (96.3). However, the F-values of task (b) and (c) are relative low, that means the problem of recognizing the eight basic emotions for emotion words is a lot more difficult than the problem of recognizing emotion and unemotion words, so we focus on task (b) and (c).

When we experiment with Word Feature(WF) and N-words Feature (NF), we have observed that word feature (w_i) and a window of previous and next word (w_{i-1}, w_i, w_{i+1}) give the best results (a=96.5, b=50.4, c=69.0). Compared with (w_{i-1}, w_i, w_{i+1}), a larger window of previous and next two words ($w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$) reduces the F-value. This demonstrates that w_i and w_{i-1}, w_i, w_{i+1} are effective features for word emotion recognition.

When POS Feature (POSF) is added, the F-value is increased. Especially the F-value is increased to (a=97.1, b=51.9, c=72.0) when pos_i and $pos_{i-1}, pos_i, pos_{i+1}$ are added.

We also find that Pre-N-words Emotion Feature (PNEF) ($pre\_e_0, ..., pre\_e_{i-1}$) increases the F-value, but previous one word emotion can not increases the F-value.

As can be seen from table 2, when only contextual features are used, the highest F-value is (a=97.1, b=53.0, c=72.7) when Pre-is-degree-word Feature (PDF), Pre-is-negative-word Feature (PNF), Pre-is-conjunction Feature (PCF) are added.

3 Semantic Feature

To know if semantic information is useful for emotion recognition, we have used probabilistic latent semantic analysis (PLSA) (Hofmann, 1999) to cluster words and sentences. PLSA clusters documents based on the term-document cooccurrence which results in semantic decomposition of the term-document matrix into a lower dimensional latent space. PLSA can be defined as:

$$P(s,w) = \sum_{z \in Z} P(z)P(s|z)P(w|z) \qquad (3)$$

where $p(s,w)$ is the probability of word w and sentence s co-occurrence, $P(s|z)$ is the probability of a sentence given a semantic class z, and $P(w|z)$ is the probability of a word given a semantic class z.

For word clustering, We made the assignment based on the maximum $p(z|w)$, if $p(z^{'}|w) =$ max $p(z|w)$, then w was assigned to $z^{'}$. Sentence clustering is similar to word clustering. Word clustering and sentence clustering are run separately. The word class_id and sentence class_id are used as semantic feature (SF), which including sentence class feature (SCF) and word class feature (WCF). PeenAspect implementation of PLSA has been used for our expriments [3].

Table 3 gives the results of F-value for combined all contextual features and semantic feature in the MaxEnt based Chinese word emotion recognition.

As can be seen from table 3, when SCF is used, the best result is obtained when the cluster number is 100; when WCF is used, the best result is obtained when the cluster number is 100 or 160. The results demonstrate the effectiveness of using SCF is a little higher than using WCF.

[3] http://www.cis.upenn.edu/datamining/software_dist/PennAspect/

Table 2: F-value for different contextual features in the MaxEnt based Chinese word emotion recognition

(a) recognize emotion or unemotion words
(b) recognize the eight basic emotions for emotion words (complete matching)
(c) recognize the eight basic emotions for emotion words (single emotion matching)

Feature type	Features	F-value		
		(a)	(b)	(c)
WF	$f1 = w_i$	96.3	45.9	63.0
NF	$f1 = w_{i-1}, w_i, w_{i+1}$	94.8	44.8	60.7
	$f1 = w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2}$	92.4	28.4	40.3
WF+NF	$f1 = w_i$; $f2 = w_{i-1}, w_i, w_{i+1}$	96.5	50.4	69.0
WF+NF +POSF	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$	96.8	51.5	71.1
	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_{i-1}, pos_i, pos_{i+1}$	97.0	51.7	71.6
	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$ $f4 = pos_{i-1}, pos_i, pos_{i+1}$	97.1	51.9	72.0
WF+NF +POSF	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$ $f4 = pos_{i-1}, pos_i, pos_{i+1}$ $f5 = pre\_e_{i-1}$	97.1	51.9	72.0
+PNEF	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$ $f4 = pos_{i-1}, pos_i, pos_{i+1}$ $f5 = pre\_e_0, ..., pre\_e_{i-1}$	97.1	52.4	72.2
WF+NF +POSF +PNEF +PDF +PNF +PCF	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$ $f4 = pos_{i-1}, pos_i, pos_{i+1}$ $f5 = pre\_e_0, ..., pre\_e_{i-1}$ $f6 = ?(w_{i-1}\ is\ a\ degree\ word)$ $f7 = ?(w_{i-1}\ is\ a\ negative\ word)$ $f8 = ?(w_{i-1}\ is\ a\ conjunction)$	97.1	53.0	72.7

4 Encoding Feature: Emotion Components of Word

Researches on the psychology of concepts show that categories in the human mind are not simply sets with clearcut boundaries (Murphy, 2002; Hampton, 2007). Word emotions are certainly related to mental concepts. As for emotion states, most theorists appear to take a combinatorial view. Plutchik (1962), for example, talks about "mixed states", "dyads" and "triads" of primary emotions. Similarly, Averill (1975) argues for compound emotions based on more elementary ones. And one model, suggested by Ekman (1982) (emotion blends) and Plutchik (mixed states), is that emotions mix (Ortony, 1988). According to these researches, we use an encoding feature: emotion components of word.

"Emotion components of word" describes the combined basic emotions in a word, which is represented by eight binary digits, and each digit corresponding to a basic emotion class respectively. For example, the word "喜欢 (like)", its possible emotion components in a certain context is "01100000", which expresses the combined emotions by joy and love.

With the expression of emotion components of word, it is possible to distinguish direct emotion words and indirect emotion words. Those words always demonstrate similar emotion components in different contexts can be regarded as direct emotion words, accordingly, those words demonstrate different emotion components in different contexts can be regarded as indirect emotion words. With the expression of emotion components in word, the problem of expressing emotion ambiguity in words can be solved. The same word in different contexts may reflect different emotions, which can be expressed by different emotion components. The emotions of words with multiple emotions also can be expressed by emotion components.

Table 3: F-value for combined contextual features (CF) and semantic feature (SF) (including sentence class feature (SCF) and word class feature (WCF))

Feature type	Cluster number	F-value		
		(a)	(b)	(c)
CF+SCF	20	97.0	53.1	72.8
	40	97.0	53.4	72.7
	60	97.0	53.5	72.8
	80	97.0	52.9	72.5
	100	97.0	**53.6**	**73.1**
	120	97.0	53.1	72.7
	150	97.0	53.2	72.9
	180	97.0	53.4	73.1
CF+WCF	40	97.0	53.1	72.8
	100	97.0	**53.4**	**72.9**
	160	97.0	**53.4**	**72.9**
	220	97.0	53.3	72.9
	280	97.0	53.2	72.8
	370	97.0	53.1	72.8

The statistics of word emotion components in Ren-CECps show that there are a total of 68 emotion components in all of 22,095 annotated emotion words without repetitions. Figure 1 shows the growth curve of word emotion components number with emotion word number increase.

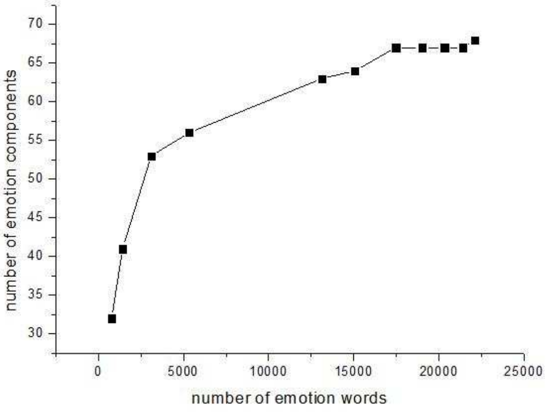


Figure 1: The growth curve of word emotion components

As can be seen from figure 1, the number increase of word emotion components shows a very slow growth rate with the number increase of emotion words. We can conclude that the space of word emotion components is a relatively small space.

In the model of MaxEnt based Chinese word emotion recognition, the Pre-N-words Emotion Feature (PNEF) and emotion output can be encoded to emotion components.

Pre-N-words Emotion Components Feature (PNECF): The emotion components of its previous words for a word (w_i). The value of this feature is obtained only after the computation of the emotion components for its previous words.

Table 4 gives the results of F-value for the combined contextual features and encoding feature.

As can be seen in table 4, when Pre-N-words Emotion Feature (PNEF) is replaced by Pre-N-words Emotion Components Feature (PNECF), and emotion components are output as results, F-value is increased up to (a=97.3, b=57.3, c=73.3). Then based on this result, we firstly trained a word emotion based model, then the word emotion outputs of this model are used as Pre-N-words Emotion Feature (PNEF) for the word emotion components based model. A significant F-value improvement of task (b) and (c) (b=62.5, c=73.7) over only using contextual and semantic features was observed after adding the combined word emotion and word emotion components as feature.

5 Discussion

5.1 Word Emotion Agreement on People's Judgments

The final aim of a human-computer interaction recognition system is to get the result close to people's judgments. As word emotion is inherently uncertain and subjective, here we report the annotation agreement on word emotion of Ren-CECps, which can be taken as an evaluation criteria for a algorithm.

To measure the annotation agreement of Ren-CECps, three annotators independently annotated 26 documents with a total of 805 sentences, 19,738 words. We use the following two metrics to measure agreement on word emotion annotation.

(1) Kappa coefficient of agreement (Carletta, 1996). It is a statistic adopted by the computa-

Table 4: F-value for the combined contextual features and encoding feature

Feature type	Features	F-value		
		(a)	(b)	(c)
WF+NF+POSF+PNECF +PDF+PNF+PCF	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$ $f4 = pos_{i-1}, pos_i, pos_{i+1}$ $f5 = pre\_es_0, ..., pre\_es_{i-1}$ $f6 = ?(w_{i-1}\ is\ a\ degree\ word)$ $f7 = ?(w_{i-1}\ is\ a\ negative\ word)$ $f8 = ?(w_{i-1}\ is\ a\ conjunction)$	97.3	57.3	73.3
WF+NF+POSF+PNEF +PNECF+PDF+PNF+PCF	$f1 = w_i$ $f2 = w_{i-1}, w_i, w_{i+1}$ $f3 = pos_i$ $f4 = pos_{i-1}, pos_i, pos_{i+1}$ $f5 = pre\_e_0, ..., pre\_e_{i-1}$ $f6 = pre\_es_0, ..., pre\_es_{i-1}$ $f7 = ?(w_{i-1}\ is\ a\ degree\ word)$ $f8 = ?(w_{i-1}\ is\ a\ negative\ word)$ $f9 = ?(w_{i-1}\ is\ a\ conjunction)$	97.3	62.5	73.7

tional linguistics community as a standard measure.

(2) Voting agreement. It is used to measure how much intersection there is between the sets of word emotions identified by the annotators. It includes majority-voting agreement ($Agreement_{MV}$) and all-voting agreement ($Agreement_{AV}$). $Agreement_{MV}$ is defined as follows. Let A, B and C be the sets of word emotion components annotated by annotators a, b and c respectively. The expert coder is the set of expressions that agreed by at least two annotators, see equation (4).

$$Agreement_{MV} = Avg(\frac{count(t_i = e_j)}{count(t_i)}) \quad (4)$$

In which, $t_i \in T$, $e_j \in E$, $T = A \bigcup B \bigcup C$, $E = (A \bigcap B) \bigcup (A \bigcap C) \bigcup (B \bigcap C)$.

Accordingly, the expert coder of $Agreement_{AV}$ is the set of expressions that agreed by all annotators.

The above two metrics are used to measure the agreements on: (a) determine if a word is an emotion or unemotion word; (b) determine the eight basic emotions for emotion words (complete emotion matching); (c) determine the eight basic emotions for emotion words (single matching). (b) and (c) are provided that at least two people to believe the word is an emotion word. Table 5 shows the agreements measured by the two metrics.

As shown in table 5, it is easier for annotators to agree at if a word contains emotion, but it is more difficult to agree on emotions or emotion components of a word. Compared with the agreement on people's judgments, our experiments gave promising results.

Table 5: Agreement of word emotion annotation measured by Kappa, Majority-voting (MV), and All-voting (AV)

Measure	Kappa	MV	AV
(a)	84.3	98.5	95.1
(b)	66.7	70.3	26.2
(c)	77.5	100	84.9

5.2 Error Analysis

Conducting an error analysis, we find that a lot of errors occur due to the recognition on multi-emotion words and indirect emotion words, especially in short sentences because the features can be extracted are too few. So more features should be considered from larger contexts, such as the topic emotion of paragraph or document.

There are some errors occur due to more than one emotion holders exist in one sentence, for example of sentence (4).

(4) 我发现女儿正看着她感兴趣的玩具。(I found that daughter was looking at the toys of her interest.)

In sentence (4), three annotators all agree that the emotion components of the word “感兴趣 (interest)” is “00000000” since they believe that this word is an unemotion word from the view of the writer. But our system give a result of “00100000” because the emotion holder “女儿 (daughter)” of the emotion word “感兴趣 (interest)” has not been considered in our algorithm. Therefore, the recognition of emotion holder is indispensable for an accurate emotion analysis system.

In addition, Chinese segmentation mistakes and phrasing error also cause errors.

6 Conclusions

Automatically perceive the emotions from text has potentially important applications in CMC (computer-mediated communication) that range from identifying emotions from online blogs to enabling dynamically adaptive interfaces. Therein words play important role in emotion expressions of text.

In this paper we explored features for recognizing word emotions in sentences. Different from previous researches on textual emotion recognition that based on affective lexicons, we believe that besides obvious emotion words referring to emotions, there are words can potentially convey emotions act only as an indirect reference. Also, quite often words that bear emotion ambiguity and multiple emotions are difficult to be recognized depending on emotion lexicons. Emotion of a word should be determined with its context.

Based on Ren-CECps (an annotated emotion corpus) and MaxEnt (Maximum entropy) model, we have experimented several contextual features and their combination, then using PLSA (probabilistic latent semantic analysis), semantic feature are demonstrated the effectiveness for word emotion recognition. A significant performance improvement over only using contextual and semantic features was observed after adding encoding feature (word emotion components). Determining intensity of word emotion and recognizing emotion of sentence or document based on word emotion are included in our future work.

Acknowledgments

This research has been partially supported by Ministry of Education, Science, Sprots and Culture, Grant-in-Aid for Challenging Exploratory Research, 21650030. We also wish to acknowledge the anonymous reviewer’s insightful comments and suggestions.

References

J. R. Averill. 1975. A semantic atlas of emotional concepts. *JSAS Catalog of Selected Documents in Psychology*.

Adam Berger, Vincent Della Pietra and Stephen A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistic* 22(1), pages 39 – 71.

Plaban Kumar Bhowmick, Animesh Mukherjee, Aritra Banik, Pabitra Mitra, Anupam Basu. 2008. A comparative study of the properties of emotional and non-Emotional words in the Wordnet: A complex network approach. In *Proceedings of International conference on natural language processing (ICON 2008)*.

Jean Carletta. 1996. Assessing agreement on classification tasks: the Kappa statistic. *Computational Linguistics*. 22(2): 249-254.

Z. Dong and Q. Dong. 2003. HowNet—a hybrid language and knowledge resource. In *Proceedings of Int’l Conf. Natural Language Processing and Knowledge Eng.*, pages 820 – 824.

Paul Ekman. 1982. *Emotion in the human face*. Cambridge University Press.

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: A publicly available lexical resource for opinion mining. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, pages 417-422.

James A. Hampton. 2007. Typicality, graded membership, and vagueness. *Cognitive Science* 31:355 – 384.

Thomas Hofmann. 1999. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI’99).*

Abe Kazemzadeh, Sungbok Lee, and Shrikanth Narayanan. 2008. An interval type-2 fuzzy logic system to translate between emotion-related ocabularies. In *Proceedings of Interspeech.*

Lun-Wei Ku, Yu-Ting Liang and Hsin-Hsi Chen. 2006. Tagging heterogeneous evaluation corpora for opinionated tasks. In *Proceedings of Conference on Language Resources and Evaluation (LREC 2006)*, pages 667-670.

Chul Min Lee, Shrikanth S. Narayanan. 2005. Toward detecting emotions in spoken dialogs. *Journal of the American Society for Information Science. IEEE Trans. on Speech and Audio Processing* 13(2):293-303.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of statistical natural language processing*. Cambridge, MA: MIT Press.

Gregory L. Murphy. 2002. *The Big Book of Concepts*. Cambridge, MA: MIT Press.

Andrew Ortony. Gerald l. Clore. Mark A. Foss. 1987. The referential structure of the affective lexicon. *Cognitive Science* 11:341-364.

Andrew Ortony, Gerald L. Clore, Allan Collins. 1988. *The Cognitive Structure of Emotions*. Cambridge University Press.

Robert Plutchik. 1962. *The emotions: Facts, theories, and a new model*. New York: Random House.

Changqin Quan and Fuji Ren. 2010. A blog emotion corpus for emotional expression analysis in Chinese. *Computer Speech & Language*, 24(4):726 – 749.

Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, and Daniel M. Ogilvie. 1966. *The General Inquirer: A computer approach to content analysis.* The MIT Press.

Carlo Strapparava and Alessandro Valitutti. 2004. Wordnet-affect: an affective extension of wordnet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pages 1083-1086.

Carlo Strapparava, Alessandro Valitutti, and Oliviero Stock. 2006. The affective weight of lexicon. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC 2006)*, pages 423-426.

Carlo Strapparava, Alessandro Valitutti, Oliviero Stock. 2007. Dances with words. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1719 – 1724.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. Extracting emotional polarity of words using spin model. 2005. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 133 – 140.

Ryoko Tokuhisa, Kentaro Inui, Yuji Matsumoto. 2008. Emotion classification using massive examples extracted from the web. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. pages 881 – 888.

Francisco Virginia and Gervás Pablo. 2006. Exploring the compositionality of emotions in text: word emotions, sentence emotions and sutomated Tagging. In *Proceedings of the AAAI-06 Workshop on Computational Aesthetics: Artificial Intelligence Approaches to Beauty and Happiness*, pages 16 – 20.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2009. Recognizing Contextual Polarity: an exploration of features for phrase-level sentiment analysis. *Computational Linguistics* 35(3): 1 – 34.

Inducing Fine-Grained Semantic Classes via Hierarchical and Collective Classification

Altaf Rahman and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
{altaf,vince}@hlt.utdallas.edu

Abstract

Research in named entity recognition and mention detection has typically involved a fairly small number of semantic classes, which may not be adequate if semantic class information is intended to support natural language applications. Motivated by this observation, we examine the under-studied problem of semantic subtype induction, where the goal is to automatically determine which of a set of 92 fine-grained semantic classes a noun phrase belongs to. We seek to improve the standard supervised approach to this problem using two techniques: hierarchical classification and collective classification. Experimental results demonstrate the effectiveness of these techniques, whether or not they are applied in isolation or in combination with the standard approach.

1 Introduction

Semantic class determination refers to the task of classifying a noun phrase (NP), be it a name or a nominal, as one of a set of pre-defined semantic classes. A semantic class classifier is a basic text-processing component in many high-level natural language processing (NLP) applications, including information-extraction (IE) systems and question-answering (QA) systems. In recent years, supervised semantic class determination has been tackled primarily in the context of (1) *coreference resolution* (e.g., Ng (2007), Huang et al. (2009)), where semantic classes are induced and subsequently used to disallow coreference between semantically incompatible NPs, and (2) the *mention detection* task in the ACE evaluations (e.g., Florian et al. (2004; 2006)), where the goal is to identify the boundary of a *mention* (i.e., a noun phrase that belongs to one of the pre-defined ACE semantic classes), its mention type (e.g., pronoun, name), and its semantic class. The output of a mention detector is then used by downstream IE components, which typically include a coreference resolution system and a relation extraction system. Owing in part to its potentially large influence on downstream IE components, accurate semantic class determination is crucial.

Over the years, NLP researchers have focused on a relatively small number of semantic classes in both NE recognition and mention detection: seven classes in the MUC-6 and MUC-7 NE recognition task, four classes in the CoNLL 2002 and 2003 NE recognition shared task, and seven classes in the ACE 2005 mention detection task. Given that one of the uses of semantic class information is to support NLP applications, it is questionable whether this purpose can be adequately served by such a small number of semantic classes. For example, given the question "Which city was the first Olympic Games held in?", it would be helpful for a QA system to know which NEs are cities. However, virtually all of the existing NE recognizers and mention detectors can only determine whether an NE is a location or not.

Our goal in this paper is to tackle the under-studied problem of determining fine-grained semantic classes (henceforth *semantic subtypes*). More specifically, we aim to classify an NP as one of the 92 fine-grained, domain-independent semantic classes that are determined to be useful for supporting the development of QA and

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 931–939,
Beijing, August 2010

IE systems in the ACE and AQUAINT programs. These 92 semantic subtypes have been used to manually annotate the NPs in the *BBN Entity Type Corpus* (Weischedel and Brunstein, 2005). Given the availability of this semantic subtype-annotated corpus, we adopt a supervised machine learning approach to semantic subtype determination. Specifically, given (the boundary of) an NP, we train a classification model to determine which of the 92 semantic subtypes it belongs to.

More importantly, we seek to improve the standard approach to semantic subtype induction described above by proposing two techniques. The first technique, collective classification, aims to address a common weakness in the standard supervised learning paradigm, where a classifier classifies each instance independently of the others and is unable to exploit any relational information between a pair (or a subset) of the instances that may be helpful for classification. The second technique, hierarchical classification, exploits the observation that these 92 semantic subtypes can be grouped into a smaller number of coarse-grained semantic types (henceforth semantic supertypes). With this two-level hierarchy, learning can proceed in a sequential fashion: given an NP, we first determine its semantic supertype and then classify it as one of the semantic subtypes that fall under the predicted supertype in the hierarchy. Empirical results show that these two techniques, when applied in isolation to the standard learning approach to subtype induction, can significantly improve its accuracy, and the best result is achieved when they are applied in combination.

The rest of the paper is organized as follows. Section 2 provides an overview of the 92 semantic subtypes and the evaluation corpus. In Section 3, we present our baseline semantic subtype classification system. Sections 4 and 5 introduce collective classification and hierarchical classification respectively, and describe how these two techniques can be used to improve the baseline semantic subtype classifier. We show evaluation results in Section 6 and conclude in Section 7.

2 Semantic Subtypes

As noted before, each name and nominal in the *BBN Entity Type Corpus* is annotated with one of the 92 semantic subtypes. In our experiments, we use all the 200 Penn Treebank Wall Street Journal articles in the corpus, yielding 17,292 NPs that are annotated with their semantic subtypes.

Table 1 presents an overview of these subtypes. Since they have been manually grouped into 29 supertypes, we also show the supertypes in the table. More specifically, the first column shows the supertypes, the second column contains a brief description of a supertype, and the last column lists the subtypes that correspond to the supertype in the first column. In cases where a supertype contains only one subtype (e.g., PERSON), the supertype is not further partitioned into different subtypes; for classification purposes, we simply treat the subtype as identical to its supertype (and hence the two always have the same name). A detailed description of these supertypes and subtypes can be found in Weischedel and Brunstein (2005). Finally, we show the class distribution: the parenthesized number after each subtype is the percentage of the 17,292 NPs annotated with the subtype.

3 Baseline Classification Model

We adopt a supervised machine learning approach to train our baseline classifier for determining the semantic subtype of an NP. This section describes the details of the training process.

Training corpus. As mentioned before, we use the Wall Street Journal articles in the BBN Entity Type Corpus for training the classifier.

Training instance creation. We create one training instance for each annotated NP, NP_i, which is either a name or a nominal, in each training text. The classification of an instance is its annotated semantic subtype value, which is one of the 92 semantic subtypes. Each instance is represented by a set of 33 features[1], as described below.

1. Mention String (3): Three features are derived from the string of NP_i. Specifically, we employ the NP string as a feature. If NP_i contains more than one token, we create one feature for each of its constituent tokens. Finally, to distinguish the different senses of a nominal, we create

[1] As we will see, since we employ an exponential model, an instance may be represented by fewer than 33 features.

Supertype	Brief Description	Subtypes
PERSON	Proper names of people.	Person (9.2).
PERSON DESC	Any head word of a common noun referring to a person or group of people.	Person Desc (16.8).
NORP	This type is named after its subtypes: nationality, religion, political, etc.	Nationality (2.9), Religion (0.1), Political (0.6), Other (0.1).
FACILITY	Names of man-made structures, including infrastructure, buildings, monuments, camps, farms, mines, ports, etc.	Building (0.1), Bridge (0.02), Airport (0.01), Attraction (0.01), Highway Street (0.05), Other (0.1).
FACILITY DESC	Head noun of a noun phrase describing buildings, bridges, airports, etc.	Building (0.5), Bridge (0.05), Airport (0.01), Highway Street (0.2), Attraction (0.02), Other (0.5).
ORGANIZATION	Names of companies, government agencies, educational institutions and other institutions.	Government (3.6), Corporation (8.3), Political (0.5), Educational (0.3), Hotel (0.04), City (0.01), Hospital (0.01), Religious (0.1), Other (0.7).
ORG DESC	Heads of descriptors of companies, educational institutions and other governments, government agencies, etc.	Government (2.1), Corporation (4.3), Political (0.2), Educational (0.1), Religious (0.1), Hotel (0.1), City (0.01), Hospital (0.02), Other (0.7).
GPE	Names of countries, cities, states, provinces, municipalities, boroughs.	Country (4.2), City (3.2), State Province (1.4), Other (0.1).
GPE DESC	Heads of descriptors of countries, cities, states, provinces, municipalities.	Country (0.8), City (0.3), State Province (0.3), Other (0.1).
LOCATION	Names of locations other than GPEs. E.g., mountain ranges, coasts, borders, planets, geo-coordinates, bodies of water.	River (0.03), Lake Sea Ocean (0.05), Region (0.2), Continent (0.1), Other (0.2).
PRODUCT	Name of any product. It does not include the manufacturer).	Food (0.01), Weapon (0.02), Vehicle (0.2), Other (0.2).
PRODUCT DESC	Descriptions of weapons and vehicles only. Cars, buses, machine guns, missiles, bombs, bullets, etc.	Food (0.01), Weapon (0.2), Vehicle (0.97), Other (0.02).
DATE	Classify a reference to a date or period.	Date (7.99), Duration (1.9), Age (0.5), Other (0.4).
TIME	Any time ending with A.M. or P.M.	Time (0.5).
PERCENT	Percent symbol or the actual word percent.	Percent (2.07).
MONEY	Any monetary value.	Money (2.9).
QUANTITY	Used to classify measurements. E.g., 4 miles, 4 grams, 4 degrees, 4 pounds, etc.	1D (0.11), 2D (0.08), 3D (0.1), Energy (0.01), Speed (0.01), Weight (0.1), Other (0.04).
ORDINAL	All ordinal numbers. E.g., First, fourth.	Ordinal (0.6).
CARDINAL	Numerals that provide a count or quantity.	Cardinal (5.1).
EVENT	Named hurricanes, battles, wars, sports events, and other named events.	War (0.03), Hurricane (0.1), Other (0.24).
PLANT	Any plant, flower, tree, etc.	Plant (0.2).
ANIMAL	Any animal class or proper name of an animal, real or fictional.	Animal (0.7).
SUBSTANCE	Any chemicals, elements, drugs, and foods. E.g., boron, penicillin, plutonium.	Food (1.1), Drug (0.46), Chemical (0.23), Other (0.9).
DISEASE	Any disease or medical condition.	Disease (0.6).
LAW	Any document that has been made into a law. E.g., Bill of Rights, Equal Rights.	Law (0.5).
LANGUAGE	Any named language.	Language (0.2).
CONTACT INFO	Address, phone.	Address (0.01), Phone (0.04).
GAME	Any named game.	Game (0.1).
WORK OF ART	Titles of books, songs and other creations.	Book (0.16), Play (0.04), Song (0.03), Painting (0.01), Other (0.4).

Table 1: The 92 semantic subtypes and their corresponding supertypes.

a feature whose value is the concatenation of the head of NP_i and its WordNet sense number.[2]

2. Verb String (3): If NP_i is governed by a verb, the following three features are derived from the governing verb. First, we employ the string of the governing verb as a feature. Second, we create a feature whose value is the semantic role of the

[2] We employ the sense number that is manually annotated for each NP in the WSJ corpus as part of the OntoNotes project (Hovy et al., 2006).

governing verb.[3] Finally, to distinguish the different senses of the governing verb, we create a feature whose value is the concatenation of the verb and its WordNet sense number.

3. Semantic (5): We employ five semantic features. First, if NP_i is an NE, we create a feature whose value is the NE label of NP_i, as determined by the Stanford CRF-based NE recognizer (Finkel et al., 2005). However, if NP_i is a nominal, we create a feature that encodes the WordNet semantic class of which it is a hyponym, using the manually determined sense of NP_i.[4] Moreover, to improve generalization, we employ a feature whose value is the WordNet synset number of the head noun of a nominal. If NP_i has a governing verb, we also create a feature whose value is the WordNet synset number of the verb. Finally, if NP_i is a nominal, we create a feature based on its *WordNet equivalent concept*. Specifically, for each entity type defined in ACE 2005[5], we create a list containing all the word-sense pairs in WordNet (i.e., synsets) whose glosses are compatible with that entity type.[6] Then, given NP_i and its sense, we use these lists to determine if it belongs to any ACE 2005 entity type. If so, we create a feature whose value is the corresponding entity type.

4. Morphological (8). If NP_i is a nominal, we create eight features: prefixes and suffixes of length one, two, three, and four.

5. Capitalization (4): We create four capitalization features to determine whether NP_i `IsAllCap`, `IsInitCap`, `IsCapPeriod`, and `IsAllLower` (see Bikel et al. (1999)).

6. Gazetteers (8): We compute eight gazetteer-based features, each of which checks whether NP_i is in a particular gazetteer. The eight dictionaries contain pronouns (77 entries), common words and words that are not names (399.6k), person names (83.6k), person titles and honorifics (761), vehicle words (226), location names (1.8k), company names (77.6k), and nouns extracted from WordNet that are hyponyms of PERSON (6.3k).

7. Grammatical (2): We create a feature that encodes the part-of-speech (POS) sequence of NP_i obtained via the Stanford POS tagger (Toutanova et al., 2003). In addition, we have a feature that determines whether NP_i is a nominal or not.

We employ maximum entropy (MaxEnt) modeling[7] for training the baseline semantic subtype classifier. MaxEnt is chosen because it provides a *probabilistic* classification for each instance, which we will need to perform collective classification, as described in the next section.

4 Collective Classification

One weakness of the baseline classification model is that it classifies each instance independently. In particular, the model cannot take into account relationships between them that may be helpful for improving classification accuracy. For example, if two NPs are the same string in a given document, then it is more likely than not that they have the same semantic subtype according to the "one sense per discourse" hypothesis (Gale et al., 1992). Incorporating this kind of *relational* information into the feature set employed by the baseline system is not an easy task, since each feature characterizes only a single NP.

To make use of the relational information, one possibility is to design a new learning procedure. Here, we adopt a different approach: we perform collective classification, or joint probabilistic inference, on the output of the baseline model. The idea is to treat the output for each NP, which is a probability distribution over the semantic subtypes, as its *prior* label/class distribution, and convert it into a *posterior* label/class distribution by exploiting the available relational information as an additional piece of evidence. For this purpose, we will make use of *factor graphs*. In this section, we first give a brief overview of factor graphs[8], and show how they can be used to perform joint

[3] We also employ the semantic role that is manually annotated for each NP in the WSJ corpus in OntoNotes.

[4] The semantic classes we considered are person, location, organization, date, time, money, percent, and object.

[5] The ACE 2005 entity types include person, organization, GPE, facility, location, weapon, and vehicle.

[6] Details of how these lists are constructed can be found in Nicolae and Nicolae (2006).

[7] We use the MaxEnt implementation available at http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html

[8] See Bunescu and Mooney (2004) and Loeliger (2004) for a detailed introduction to factor graphs.

inference for semantic subtype determination.

4.1 Factor Graphs

Factor graphs model optimization problems of an objective function g, which is a real-valued function of n random variables $X_1, ..., X_n$. We assume that g can be decomposed into a product of m *factors*. In other words, $g(X_1, ..., X_n) = f_1(s_1(X_1, ..., X_n)) ... f_m(s_m(X_1, ..., X_n))$, where each factor f_k is a real-valued function of some subset of $X_1, ..., X_n$, denoted as $s_k(X_1, ..., X_n)$. Each f_k can be thought of as a feature function that computes the *compatibility* of an assignment of values to the variables in $s_k(X_1, ..., X_n)$ with respect to a user-defined feature. Hence, a larger function value is more desirable, as it corresponds to a more compatible assignment of values to the variables involved.

A factor graph consists of two types of nodes: variable nodes and factor nodes. Each random variable X_i is represented by a variable node, and each factor f_k is represented by a factor node. Each factor node f_k is connected only to the nodes corresponding to s_k. This results in a bipartite graph, where edges exist only between a variable node and a factor node.

Given this graph, there are several methods for finding an optimal assignment of the random variables $X_1, ..., X_n$ such that the objective function g is maximized. Exact inference using the sum-product algorithm (Kschischang et al., 2001) is possible if there are no cycles in the graph; otherwise a belief propagation algorithm, such as loopy belief propagation (Murphy et al., 1999), can be applied. Although there are no cycles in our factor graphs, we choose to use loopy belief propagation as our inferencer, since it performs approximate inference and is therefore computationally more efficient than an exact inferencer.

4.2 Application to Subtype Inference

To apply joint inference to semantic subtype induction, we create one factor graph for each test document, where each variable node is random variable X_i over the set of semantic subtype labels L and represents an NP, NP$_i$, in the document. To retain the prior probabilities over the semantic subtype labels $l_q \in L$ obtained from the baseline classification model, each variable node is given a factor $f(X_i) = P(X_i = l_q)$. If no additional factors that model the relation between two nodes/instances are introduced, maximizing the objective function for this graph (by maximizing the product of factors) will find an assignment identical to the one obtained by taking the most probable semantic subtype label assigned to each instance by the baseline classifier.

Next, we exploit the relationship between two random variables. Specifically, we want to encourage the inference algorithm to assign the same label to two variables if there exists a relation between the corresponding NPs that can provide strong evidence that they should receive the same label. To do so, we create a *pairwise* factor node that connects two variable nodes if the aforementioned relation between the underlying NPs is satisfied. However, to implement this idea, we need to address two questions.

First, *which relation between two NPs can provide strong evidence that they have the same semantic subtype?* We exploit the coreference relation. Intuitively, the coreference relation is a reasonable choice, as coreferent entities are likely to have the same semantic subtype. Here, we naively posit two NPs as coreferent if at least one of the following conditions is satisfied: (1) they are the same string after determiners are removed; (2) they are aliases (i.e., one is an acronym or abbreviation of the other); and (3) they are both proper names and have at least one word in common (e.g., "Delta" and "Delta Airlines").[9]

Second, *how can we define a pairwise factor, f_{pair}, so that it encourages the inference algorithm to assign the same label to two nodes?* One possibility is to employ the following definition:

$$\begin{aligned} & f_{pair}(X_i, X_j) \\ = \; & P(X_i = l_p, X_j = l_q), \text{where } \; l_p, l_q \in L \\ = \; & \begin{cases} 1 & \text{if } l_p = l_q \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

In essence, f_{pair} prohibits the assignment of different labels to the two nodes it connects. In our

[9] The third condition can potentially introduce many false positives, positing "Bill Clinton" and "Hillary Clinton" as coreferent, for instance. However, this kind of false positives does not pose any problem for us, since the two NPs involved belong to the same semantic subtype (i.e., PERSON).

experiments, however, we "improve" f_{pair} by incorporating semantic supertype information into its definition, as shown below:

$$\begin{aligned} & f_{pair}(X_i, X_j) \\ =\ & P(X_i = l_p, X_j = l_q), \text{where } \ l_p, l_q \in L \\ =\ & \begin{cases} P_{sup}(sup(l_p)|\text{NP}_i)P_{sup}(sup(l_q)|\text{NP}_j) & \text{if } l_p = l_q \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

In this definition, $sup(l_q)$ is the supertype of l_q according to the semantic type hierarchy shown in Section 2, and $P_{sup}(sup(l_q)|\text{NP}_j)$ is the probability that NP$_j$ belongs to $sup(l_q)$ according to the semantic supertype classification model P_{sup} (see Section 5 for details on how this model can be trained). In essence, we estimate the joint probability by (1) assuming that the two events are independent, and then (2) computing each event using supertype information. Intuitively, this definition allows f_{pair} to favor those label assignments that are more compatible with the predictions of P_{sup}.

After graph construction, we apply an inferencer to compute a marginal probability distribution over the labels for each node/instance in the graph by maximizing the objective function g, and output the most probable label for each instance according to its marginal distribution.

5 Hierarchical Classification

The pairwise factor f_{pair} defined above exploits supertype information in a *soft* manner, meaning that the most probable label assigned to an NP by an inferencer is not necessarily consistent with its predicted supertype (e.g., an NP may receive Hotel as its subtype even if its supertype is PERSON). In this section, we discuss how to use supertype information for semantic subtype classification in a *hard* manner so that the predicted subtype is consistent with its supertype.

To exploit supertype information, we first train a model, P_{sup}, for determining the semantic supertype of an NP using MaxEnt. This model is trained in essentially the same way as the baseline model described in Section 3. In particular, it is trained on the same set of instances using the same feature set as the baseline model. The only difference is that the class value of each training instance is the semantic supertype of the associated NP rather than its semantic subtype.

Next, we train 29 supertype-specific classification models for determining the semantic subtype of an NP. For instance, the ORGANIZATION-specific classification model will be used to classify an NP as belonging to one of its subtypes (e.g., Government, Corporation, Political agencies). A supertype-specific classification model is trained much like the baseline model. Each instance is represented using the same set of features as in the baseline, and its class label is its semantic subtype. The only difference is that the model is only trained only on the subset of the instances for which it is intended. For instance, the ORGANIZATION-specific classification model is trained only on instances whose class is a subtype of ORGANIZATION.

After training, we can apply the supertype classification model and the supertype-specific subtype classification model to determine the semantic subtype of an NP in a hierarchical fashion. Specifically, we first employ the supertype model to determine its semantic supertype. Then, depending on this predicted semantic supertype, we use the corresponding subtype classification model to determine its subtype.

6 Evaluation

For evaluation, we partition the 200 Wall Street Journal Articles in the BBN Entity Type corpus into a training set and a test set following a 80/20 ratio. As mentioned before, each text in the Entity Type corpus has its NPs annotated with their semantic subtypes. Test instances are created from these texts in the same way as the training instances described in Section 3. To investigate whether we can benefit from hierarchical and collective classifications, we apply these two techniques to the Baseline classification model in isolation and in combination, resulting in the four sets of results in Tables 2 and 3.

The Baseline results are shown in the second column of Table 2. Due to space limitations, it is not possible to show the result for each semantic subtype. Rather, we present semantic supertype results, which are obtained by micro-averaging the corresponding semantic subtype results and are expressed in terms of recall (R), precision (P), and F-measure (F). Note that only those semantic

		Baseline only			Baseline+Hierarchical		
	Semantic Supertype	R	P	F	R	P	F
1	PERSON	91.9	89.7	90.8	88.8	91.1	89.9
2	PERSON DESC	91.3	87.8	89.5	92.1	89.8	91.0
3	SUBSTANCE	60.0	66.7	63.2	70.0	58.3	63.6
4	NORP	87.8	90.3	89.0	91.9	90.7	91.3
5	FACILITY DESC	72.7	88.9	80.0	68.2	93.8	79.0
6	ORGANIZATION	76.6	73.8	75.2	78.5	73.2	75.8
7	ORG DESC	75.0	70.7	72.8	75.8	75.2	75.5
8	GPE	75.6	73.9	74.7	77.0	75.4	76.2
9	GPE DESC	60.0	75.0	66.7	70.0	70.0	70.0
10	PRODUCT DESC	53.3	88.9	66.7	53.3	88.9	66.7
11	DATE	85.0	85.0	85.0	84.5	85.4	85.0
12	PERCENT	100.0	100.0	100.0	100.0	100.0	100.0
13	MONEY	83.9	86.7	85.3	88.7	96.5	92.4
14	QUANTITY	22.2	100.0	36.4	66.7	66.7	66.7
15	ORDINAL	100.0	100.0	100.0	100.0	100.0	100.0
16	CARDINAL	96.0	77.4	85.7	94.0	81.0	87.0
	Accuracy	81.56			82.60		

Table 2: Results for Baseline only and Baseline with hierarchical classification.

		Baseline+Collective			Baseline+Both		
	Semantic Supertype	R	P	F	R	P	F
1	PERSON	93.8	98.1	95.9	91.9	100.0	95.8
2	PERSON DESC	93.9	88.5	91.1	92.6	89.5	91.0
3	SUBSTANCE	60.0	85.7	70.6	70.0	63.6	66.7
4	NORP	89.2	93.0	91.0	90.5	94.4	92.4
5	FACILITY DESC	63.6	87.5	73.7	68.2	93.8	79.0
6	ORGANIZATION	85.8	76.2	80.7	87.4	76.3	81.3
7	ORG DESC	75.8	74.1	74.9	75.8	74.6	75.2
8	GPE	74.1	75.8	74.9	81.5	81.5	81.5
9	GPE DESC	60.0	60.0	60.0	70.0	77.8	73.7
10	PRODUCT DESC	53.3	88.9	66.7	53.3	88.9	66.7
11	DATE	85.0	85.4	85.2	85.0	86.3	85.6
12	PERCENT	100.0	100.0	100.0	100.0	100.0	100.0
13	MONEY	83.9	86.7	85.3	90.3	96.6	93.3
14	QUANTITY	22.2	100.0	36.4	66.7	66.7	66.7
15	ORDINAL	100.0	100.0	100.0	100.0	100.0	100.0
16	CARDINAL	96.0	78.7	86.5	94.0	83.9	88.7
	Accuracy	83.70			85.08		

Table 3: Results for Baseline with collective classification and Baseline with both techniques.

supertypes with non-zero scores are shown. As we can see, only 16 of the 29 supertypes have non-zero scores.[10] Among the "traditional" semantic types, the Baseline yields good performance for PERSON, but only mediocre performance for ORGANIZATION and GPE. While additional experiments are needed to determine the reason, we speculate that this can be attributed to the fact that PERSON and PERSON DESC have only one semantic subtype (which is the supertype itself), whereas ORGANIZATION and GPE have nine and four subtypes, respectively. The classification accuracy is shown in the last row of the table. As we can see, the Baseline achieves an accuracy of 81.56.

Results obtained when hierarchical classification is applied to the Baseline are shown in the third column of Table 2. In comparison to the Baseline, accuracy rises from 81.56 to 82.60. This represents an error reduction of 5.6%, and the difference between these two accuracies is statistically significant at the $p = 0.04$ level.[11]

[10] The 13 supertypes that have zero scores are all under-represented classes, each of which accounts for less than one percent of the instances in the dataset.

[11] All significance test results in this paper are obtained using Approximate Randomization (Noreen, 1989).

1	3	7	4	2	5	6
81.4	75.8	83.3	83.7	84.1	85.2	85.6
80.4	74.9	84.3	85.3	85.3	86.1	
80.4	78.3	83.9	86.5	86.7		
81.8	76.2	85.2	87.6			
75.4	83.4	84.6				
66.2	80.9					

Table 4: Results of feature analysis.

Results obtained when collective classification alone is applied to the Baseline are shown in the second column of Table 3. In this case, the prior probability distribution over the semantic subtypes that is needed to create the factor associated with each node in the factor graph is simply the probabilistic classification of the test instance that the node corresponds to. In comparison to the Baseline, accuracy rises from 81.56 to 83.70. This represents an error reduction of 11.6%, and the difference is significant at the $p = 0.01$ level. Also, applying collective classification to the Baseline yields slightly better results than applying hierarchical classification to the Baseline, and the difference in their results is significant at the $p = 0.002$ level.

Finally, results obtained when both hierarchical and collective classification are applied to the Baseline are shown in the third column of Table 3. In this case, the prior distribution needed to create the factor associated with each node in the factor graph is provided by the supertype-specific classification model that is used to classify the test instance in hierarchical classification. In comparison to the Baseline, accuracy rises from 81.56 to 85.08. This represents an error reduction of 19.1%, and the difference is highly significant ($p < 0.001$). Also, applying both techniques to the Baseline yields slightly better results than applying only collective classification to the Baseline, and the difference in their results is significant at the $p = 0.003$ level.

6.1 Feature Analysis

Next, we analyze the effects of the seven feature types described in Section 3 on classification accuracy. To measure feature performance, we take the best-performing system (i.e., Baseline combined with both techniques), begin with all seven feature types, and iteratively remove them one by one so that we get the best accuracy. The results are shown in Table 4. Across the top line, we list the numbers representing the seven feature classes. The feature class that corresponds to each number can be found in Section 3, where they are introduced. For instance, “2” refers to the features computed based on the governing verb. The first row of results shows the system performance after removing just one feature class. In this case, removing the sixth feature class (Gazetteers) improves accuracy to 85.6, while removing the mention string features reduces accuracy to 81.4. The second row repeats this, after removing the gazetteer features.

Somewhat surprisingly, using only mention string, semantic, and grammatical features yields the best accuracy (87.6). This indicates that gazetteers, morphological features, capitalization, and features computed based on the governing verb are not useful. Removing the grammatical features yields a 3% drop in accuracy. After that, accuracy drops by 4% when semantic features are removed, whereas a 18% drop in accuracy is observed when the mention string features are removed. Hence, our analysis suggests that the mention string features are the most useful features for semantic subtype prediction.

7 Conclusions

We examined the under-studied problem of semantic subtype induction, which involves classifying an NP as one of 92 semantic classes, and showed that two techniques — hierarchical classification and collective classification — can significantly improve a baseline classification model trained using an off-the-shelf learning algorithm on the BBN Entity Type Corpus. In particular, collective classification addresses a major weakness of the standard feature-based learning paradigm, where a classification model classifies each instance independently, failing to capture the relationships among subsets of instances that might improve classification accuracy. However, collective classification has not been extensively applied in the NLP community, and we hope that our work can increase the awareness of this powerful technique among NLP researchers.

Acknowledgments

We thank the three anonymous reviewers for their invaluable comments on an earlier draft of the paper. This work was supported in part by NSF Grant IIS-0812261.

References

Bikel, Daniel M., Richard Schwartz, and Ralph M. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning: Special Issue on Natural Language Learning*, 34(1–3):211–231.

Bunescu, Razvan and Raymond J. Mooney. 2004. Collective information extraction with relational markov networks. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 483–445.

Finkel, Jenny Rose, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 363–370.

Florian, Radu, Hany Hassan, Abraham Ittycheriah, Hongyan Jing, Nanda Kambhatla, Xiaoqiang Luo, Nicolas Nicolov, and Salim Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *HLT-NAACL 2004: Main Proceedings*, pages 1–8.

Florian, Radu, Hongyan Jing, Nanda Kambhatla, and Imed Zitouni. 2006. Factorizing complex models: A case study in mention detection. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 473–480.

Gale, William, Ken Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the 4th DARPA Speech and Natural Language Workshop*, pages 233–237.

Hovy, Eduard, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60.

Huang, Zhiheng, Guangping Zeng, Weiqun Xu, and Asli Celikyilmaz. 2009. Accurate semantic class classifier for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1232–1240.

Kschischang, Frank, Brendan J. Frey, and Hans-Andrea Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519.

Loeliger, Hans-Andrea. 2004. An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21(1):28–41.

Murphy, Kevin P., Yair Weiss, and Michael I. Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 467–475.

Ng, Vincent. 2007. Semantic class induction and coreference resolution. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 536–543.

Nicolae, Cristina and Gabriel Nicolae. 2006. BestCut: A graph algorithm for coreference resolution. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 275–283.

Noreen, Eric W. 1989. *Computer Intensive Methods for Testing Hypothesis: An Introduction*. John Wiley & Sons.

Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL 2003: Proceedings of the Main Conference*, pages 173–180.

Weischedel, Ralph and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. In Linguistic Data Consortium, Philadelphia.

Fast, Greedy Model Minimization for Unsupervised Tagging

Sujith Ravi and **Ashish Vaswani** and **Kevin Knight** and **David Chiang**
University of Southern California
Information Sciences Institute
{sravi,avaswani,knight,chiang}@isi.edu

Abstract

Model minimization has been shown to work well for the task of unsupervised part-of-speech tagging with a dictionary. In (Ravi and Knight, 2009), the authors invoke an integer programming (IP) solver to do model minimization. However, solving this problem exactly using an integer programming formulation is intractable for practical purposes. We propose a novel two-stage greedy approximation scheme to replace the IP. Our method runs fast, while yielding highly accurate tagging results. We also compare our method against standard EM training, and show that we consistently obtain better tagging accuracies on test data of varying sizes for English and Italian.

1 Introduction

The task of unsupervised part-of-speech (POS) tagging with a dictionary as formulated by Merialdo (1994) is: given a raw word sequence and a dictionary of legal POS tags for each word type, tag each word token in the text. A common approach to modeling such sequence labeling problems is to build a bigram Hidden Markov Model (HMM) parameterized by *tag-bigram* transition probabilities $P(t_i|t_{i-1})$ and *word-tag* emission probabilities $P(w_i|t_i)$. Given a word sequence w and a tag sequence t, of length N, the joint probability $P(w,t)$ is given by:

$$P(w,t) = \prod_{i=1}^{N} P(w_i|t_i) \cdot P(t_i|t_{i-1}) \qquad (1)$$

We can train this model using the Expectation Maximization (EM) algorithm (Dempster and Rubin, 1977) which learns $P(w_i|t_i)$ and $P(t_i|t_{i-1})$ that maximize the likelihood of the observed data. Once the parameters are learnt, we can find the best tagging using the Viterbi algorithm.

$$\hat{t} = \arg\max_{t} P(w,t) \qquad (2)$$

Ravi and Knight (2009) attack the Merialdo task in two stages. In the first stage, they search for a minimized transition model (i.e., the smallest set of tag bigrams) that can explain the data using an integer programming (IP) formulation. In the second stage, they build a smaller HMM by restricting the transition parameters to only those tag bigrams selected in the minimization step. They employ the EM algorithm to train this model, which prunes away some of the emission parameters. Next, they use the pruned emission model along with the original transition model (which uses the full set of tag bigrams) and retrain using EM. This alternating EM training procedure is repeated until the number of tag bigrams in the Viterbi tagging output does not change between subsequent iterations. The final Viterbi tagging output from their method achieves state-of-the-art accuracy for this task. However, their minimization step involves solving an integer program, which can be very slow, especially when scaling to large-scale data and more complex tagging problems which use bigger tagsets. In this paper, we present a novel method that optimizes the same objective function using a fast greedy model selection strategy. Our contributions are summarized below:

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 940–948,
Beijing, August 2010

- We present an efficient two-phase greedy-selection method for solving the minimization objective from Ravi and Knight (2009), which runs much faster than their IP.

- Our method easily scales to large data sizes (and big tagsets), unlike the previous minimization-based approaches and we show runtime comparisons for different data sizes.

- We achieve very high tagging accuracies comparable to state-of-the-art results for unsupervised POS tagging for English.

- Unlike previous approaches, we also show results obtained when testing on the entire Penn Treebank data (973k word tokens) in addition to the standard 24k test data used for this task. We also show the effectiveness of this approach for Italian POS tagging.

2 Previous work

There has been much work on the unsupervised part-of-speech tagging problem. Goldwater and Griffiths (2007) also learn small models employing a fully Bayesian approach with sparse priors. They report 86.8% tagging accuracy with manual hyperparameter selection. Smith and Eisner (2005) design a *contrastive estimation* technique which yields a higher accuracy of 88.6%. Goldberg et al. (2008) use linguistic knowledge to initialize the the parameters of the HMM model prior to EM training. They achieve 91.4% accuracy. Ravi and Knight (2009) use a Minimum Description Length (MDL) method and achieve the best results on this task thus far (91.6% word token accuracy, 91.8% with random restarts for EM). Our work follows a similar approach using a model minimization component and alternate EM training.

Recently, the integer programming framework has been widely adopted by researchers to solve other NLP tasks besides POS tagging such as semantic role labeling (Punyakanok et al., 2004), sentence compression (Clarke and Lapata, 2008), decipherment (Ravi and Knight, 2008) and dependency parsing (Martins et al., 2009).

3 Model minimization formulated as a Path Problem

The complexity of the model minimization step in (Ravi and Knight, 2009) and its proposed approximate solution can be best understood if we formulate it as a path problem in a graph.

Let $w = w_0, w_1, \ldots, w_N, w_{N+1}$ be a word sequence where $w_1, \ldots, w_N$ are the input word tokens and $\{w_0, w_{N+1}\}$ are the $start/end$ tokens. Let $T = \{T_1, \ldots, T_K\} \bigcup \{T_0, T_{K+1}\}$ be the fixed set of all possible tags. T_0 and T_{K+1} are special tags that we add for convenience. These would be the $start$ and end tags that one typically adds to the HMM lattice. The tag dictionary D contains entries of the form (w_i, T_j) for all the possible tags T_j that word token w_i can have. We add entries (w_0, T_0) and (w_{K+1}, T_{K+1}) to D. Given this input, we now create a directed graph $G(V, E)$. Let $C_0, C_1 \ldots, C_{K+1}$ be columns of nodes in G, where column C_i corresponds to word token w_i. For all $i = 0, \ldots, N+1$ and $j = 0, \ldots, K+1$, we add node $C_{i,j}$ in column C_i if $(w_i, T_j) \in D$. Now, $\forall i = 0, \ldots, N$, we create directed edges from every node in C_i to every node in C_{i+1}. Each of these edges $e = (C_{i,j}, C_{i+1,k})$ is given the label (T_j, T_k) which corresponds to a tag bigram. This creates our directed graph. Let $l(e)$ be the tag bigram label of edges $e \in E$. For every path P from $C_{0,0}$ to $C_{N+1,K+1}$, we say that P uses an edge label or tag bigram (T_j, T_k) if there exists an edge e in P such that $l(e) = (T_j, T_k)$. We can now formulate the the optimization problem as: *Find the smallest set S of tag bigrams such that there exists at least one path from $C_{0,0}$ to $C_{N+1,K+1}$ using only the tag bigrams in S.* Let us call this the Minimal Tag Bigram Path (MinTagPath) problem.

Figure 1 shows an example graph where the input word sequence is $w_1, \ldots, w_4$ and $T = \{T_1, \ldots, T_3\}$ is the input tagset. We add the start/end word tokens $\{w_0, w_5\}$ and corresponding tags $\{T_0, T_4\}$. The edges in the graph are instantiated according to the word/tag dictionary D provided as input. The node and edge labels are also illustrated in the graph. Our goal is to find a path from $C_{0,0}$ to $C_{5,4}$ using the smallest set of tag bigrams.

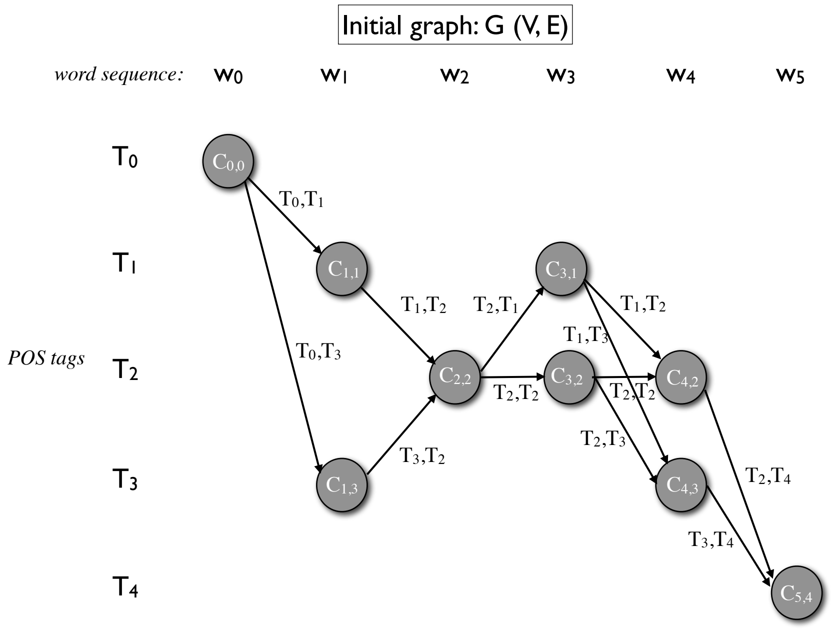


Figure 1: Graph instantiation for the MinTagPath problem.

4 Problem complexity

Having defined the problem, we now show that it can be solved in polynomial time even though the number of paths from $C_{0,0}$ to $C_{N+1,K+1}$ is exponential in N, the input size. This relies on the assumption that the tagset T is fixed in advance, which is the case for most tagging tasks.[1] Let B be the set of all the tag bigram labels in the graph, $B = \{l(e), \forall e \in E\}$. Now, the size of B would be at most $K^2 + 2K$ where every word could be tagged with every possible tag. For $m = 1 \ldots |B|$, let B^m be the set of subsets of B each of which have size m. Algorithm 1 optimally solves the MinTagPath problem.

Algorithm 1 Brute Force solution to MinTagPath

```
for $m = 1$ to $|B|$ do
  for $s_i \in \mathcal{B}^m$ do
    Use Breadth First Search (BFS) to check if $\exists$ path $P$ from $C_{0,0}$ to $C_{N+1,K+1}$ using only the tag bigrams in $s_i$.
    if $P$ exists then
      return $s_i, m$
    end if
  end for
end for
```

Algorithm 1 basically enumerates all the possible subsets of B, from the smallest to the largest, and checks if there is a path. It exits the first time a path is found and therefore finds the smallest possible set s_i of size m such that a path exists that uses only the tag bigrams in s_i. This implies the correctness of the algorithm. To check for path existence, we could either throw away all the edges from E not having a label in s_i, and then execute a Breadth-First-Search (BFS) or we could traverse only the edges with labels in s_i during BFS. The running time of Algorithm 1 is easy to calculate. Since, in the worst case we go over all the subsets of size $m = 1, \ldots, |B|$ of B, the number of iterations we can perform is at most $2^{|B|}$, the size of the powerset $\mathcal{P}$ of B. In each iteration, we do a BFS through the lattice, which has $O(N)$ time complexity[2] since the lattice size is linear in N and BFS is linear in the lattice size. Hence the running time is $\leq 2^{|B|} \cdot O(N) = O(N)$. Even though this shows that MinTagPath can be solved in polynomial time, the time complexity is prohibitively large. For the Penn Treebank, $K = 45$ and the

[1] If K, the size of the tagset, is a variable as well, then we suspect the problem is NP-hard.

[2] Including throwing away edges or not.

worst case running time would be $\approx 10^{13.55} \cdot N$. Clearly, for all practical purposes, this approach is intractable.

5 Greedy Model Minimization

We do not know of an efficient, exact algorithm to solve the MinTagPath problem. Therefore, we present a simple and fast two-stage greedy approximation scheme. Notice that an optimal path P (or any path) *covers* all the input words i.e., every word token w_i has one of its possible taggings in P. Exploiting this property, in the first phase, we set our goal to cover all the word tokens using the least possible number of tag bigrams. This can be cast as a set cover problem (Garey and Johnson, 1979) and we use the set cover greedy approximation algorithm in this stage. The output tag bigrams from this phase might still not allow any path from $C_{0,0}$ to $C_{N+1,K+1}$. So we carry out a second phase, where we greedily add a few tag bigrams until a path is created.

5.1 Phase 1: Greedy Set Cover

In this phase, our goal is to cover all the word tokens using the least number of tag bigrams. The covering problem is exactly that of set cover. Let $U = \{w_0, \ldots, w_N + 1\}$ be the set of elements that needs to be covered (in this case, the word tokens). For each tag bigram $(T_i, T_j) \in B$, we define its corresponding covering set S_{T_i,T_j} as follows:

$$\begin{aligned} S_{T_i,T_j} \;=\; \{w_n \;:\; & ((w_n, T_i) \in D \\ & \wedge (C_{n,i}, C_{n+1,j}) \in E \\ & \wedge l(C_{n,i}, C_{n+1,j}) = (T_i, T_j)) \\ \bigvee \; & ((w_n, T_j) \in D \\ & \wedge (C_{n-1,i}, C_{n,j}) \in E \\ & \wedge l(C_{n-1,i}, C_{n,j}) = (T_i, T_j))\} \end{aligned}$$

Let the set of covering sets be X. We assign a cost of 1 to each covering set in X. The goal is to select a set $CHOSEN \subseteq X$ such that $\bigcup_{S_{T_i,T_j} \in CHOSEN} = U$, minimizing the total cost of $CHOSEN$. This corresponds to covering all the words with the least possible number of tag bigrams. We now use the greedy approximation algorithm for set cover to solve this problem. The pseudo code is shown in Algorithm 2.

Algorithm 2 Set Cover : Phase 1

```
Definitions
Define $CAND$ : Set of candidate covering sets in the current iteration
Define $U_{rem}$ : Number of elements in $U$ remaining to be covered
Define $E_{S_{T_i,T_j}}$ : Current effective cost of a set
Define $Itr$ : Iteration number

Initializations
LET $CAND = X$
LET $CHOSEN = \emptyset$
LET $U_{rem} = U$
LET $Itr = 0$
LET $E_{S_{T_i,T_j}} = \frac{1}{|S_{T_i,T_j}|}, \forall\ S_{T_i,T_j} \in CAND$

while $U_{rem} \neq \emptyset$ do
  $Itr \leftarrow Itr + 1$
  Define $\hat{S}_{Itr} = \underset{S_{T_i,T_j} \in CAND}{\text{argmin}} E_{S_{T_i,T_j}}$
  $CHOSEN = CHOSEN \bigcup \hat{S}_{Itr}$
  Remove $\hat{S}_{Itr}$ from $CAND$
  Remove all the current elements in $\hat{S}_{Itr}$ from $U_{rem}$
  Remove all the current elements in $\hat{S}_{Itr}$ from every $S_{T_i,T_j} \in CAND$
  Update effective costs, $\forall\ S_{T_i,T_j} \in CAND$, $E_{S_{T_i,T_j}} = \frac{1}{|S_{T_i,T_j}|}$
end while
return $CHOSEN$
```

For the graph shown in Figure 1, here are a few possible covering sets S_{T_i,T_j} and their initial effective costs $E_{S_{T_i,T_j}}$.

- $S_{T_0,T_1} = \{w_0, w_1\}$, $E_{S_{T_0,T_1}} = 1/2$
- $S_{T_1,T_2} = \{w_1, w_2, w_3, w_4\}$, $E_{S_{T_1,T_2}} = 1/4$
- $S_{T_2,T_2} = \{w_2, w_3, w_4\}$, $E_{S_{T_2,T_2}} = 1/3$

In every iteration Itr of Algorithm 2, we pick a set $\hat{S}_{Itr}$ that is most cost effective. The elements that $\hat{S}_{Itr}$ covers are then removed from all the remaining candidate sets and U_{rem} and the effectiveness of the candidate sets is recalculated for the next iteration. The algorithm stops when all elements of U i.e., all the word tokens are covered. Let, $B_{CHOSEN} = \{(T_i, T_j) : S_{T_i,T_j} \in$

$CHOSEN\}$, be the set of tag bigrams that have been chosen by set cover. Now, we check, using BFS, if there exists a path from $C_{0,0}$ to $C_{N+1,K+1}$ using only the tag bigrams in B_{CHOSEN}. If not, then we have to add tag bigrams to B_{CHOSEN} to enable a path. To accomplish this, we carry out the second phase of this scheme with another greedy strategy (described in the next section).

For the example graph in Figure 1, one possible solution $B_{CHOSEN} = \{(T_0, T_1), (T_1, T_2), (T_2, T_4)\}$.

5.2 Phase 2: Greedy Path Completion

We define a graph $G_{CHOSEN}(V', E') \subseteq G(V, E)$ that contains the edges $e \in E$ such $l(e) \in B_{CHOSEN}$.

Let $B_{CAND} = B \setminus B_{CHOSEN}$, be the current set of candidate tag bigrams that can be added to the final solution which would create a path. We would like to know how many *holes* a particular tag bigram (T_i, T_j) can fill. We define a hole as an edge e such that $e \in G \setminus G_{CHOSEN}$ and there exists $e', e'' \in G_{CHOSEN}$ such that $tail(e') = head(e) \wedge tail(e) = head(e'')$.

Figure 2 illustrates the graph G_{CHOSEN} using tag bigrams from the example solution to Phase 1 (Section 5.1). The dotted edge $(C_{2,2}, C_{3,1})$ represents a hole, which has to be filled in the current phase in order to complete a path from $C_{0,0}$ to $C_{5,4}$.

In Algorithm 3, we define the effectiveness of a candidate tag bigram $H(T_i, T_j)$ to be the number of holes it covers. In every iteration, we pick the most effective tag bigram, fill the holes and recalculate the effectiveness of the remaining candidate tag bigrams.

Algorithm 3 returns B_{FINAL}, the final set of chosen tag bigrams. It terminates when a path has been found.

Algorithm 3 Greedy Path Complete : Phase 2

```
Define $B_{FINAL}$ : Final set of tag bigrams selected by the two-phase greedy approach

LET $B_{FINAL} = B_{CHOSEN}$
LET $H(T_i, T_j) = |\{e\}|$ such that $l(e) = (T_i, T_j)$ and $e$ is a hole, $\forall\ (T_i, T_j) \in B_{CAND}$

while $\nexists$ path $P$ from $C_{0,0}$ to $C_{N+1,K+1}$ using only $(T_i, T_j) \in B_{CHOSEN}$ do
  Define $(\hat{T_i}, \hat{T_j}) = \underset{(T_i,T_j) \in B_{CAND}}{\text{argmax}} H(T_i, T_j)$
  $B_{FINAL} = B_{FINAL} \bigcup (\hat{T_i}, \hat{T_j})$
  Remove $(\hat{T_i}, \hat{T_j})$ from $B_{CAND}$
  $G_{CHOSEN} = G_{CHOSEN} \bigcup \{e\}$ such that $l(e) = (T_i, T_j)$
  $\forall\ (T_i, T_j) \in B_{CAND}$, Recalculate $H(T_i, T_j)$
end while
return $B_{FINAL}$
```

5.3 Fitting the Model

Once the greedy algorithm terminates and returns a minimized grammar of tag bigrams, we follow the approach of Ravi and Knight (2009) and fit the minimized model to the data using the alternating EM strategy. The alternating EM iterations are terminated when the change in the size of the observed grammar (i.e., the number of unique tag bigrams in the tagging output) is $\leq 5\%$. We refer to our entire approach using greedy minimization followed by EM training as MIN-GREEDY.

6 Experiments and Results

6.1 English POS Tagging

Data: We use a standard test set (consisting of 24,115 word tokens from the Penn Treebank) for the POS tagging task (described in Section 1). The tagset consists of 45 distinct tag labels and the dictionary contains 57,388 word/tag pairs derived from the entire Penn Treebank. Per-token ambiguity for the test data is about 1.5 tags/token. In addition to the standard 24k dataset, we also train and test on larger data sets of 48k, 96k, 193k, and the entire Penn Treebank (973k).

Methods: We perform comparative evaluations for POS tagging using three different methods:

1. **EM**: Training a bigram HMM model using EM algorithm.

2. **IP**: Minimizing grammar size using integer programming, followed by EM training (Ravi and Knight, 2009).

3. **MIN-GREEDY**: Minimizing grammar size using the Greedy method described in Sec-

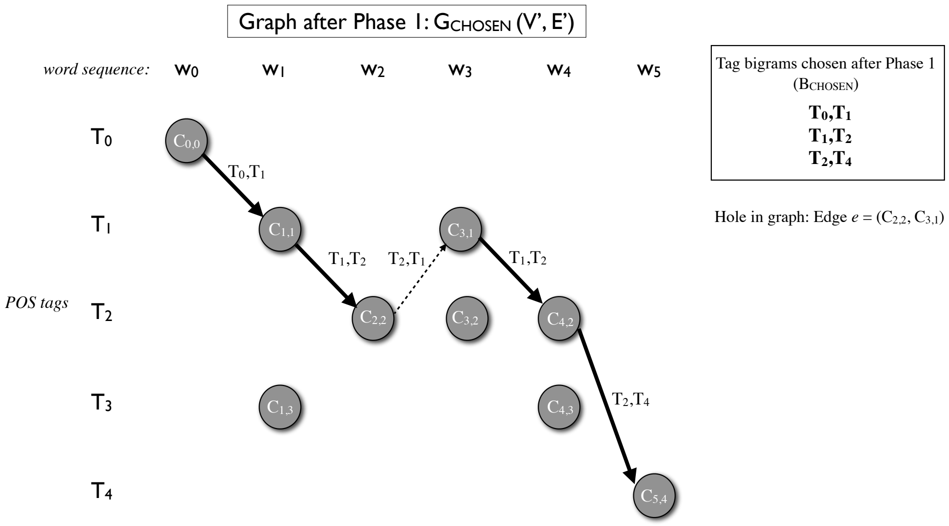


Figure 2: Graph constructed with tag bigrams chosen in Phase 1 of the MIN-GREEDY method.

tion 5, followed by EM training.

Results: Figure 3 shows the tagging performance (word token accuracy %) achieved by the three methods on the standard test (24k tokens) as well as Penn Treebank test (PTB = 973k tokens). On the 24k test data, the MIN-GREEDY method achieves a high tagging accuracy comparable to the previous best from the IP method. However, the IP method does not scale well which makes it infeasible to run this method in a much larger data setting (the entire Penn Treebank). MIN-GREEDY on the other hand, faces no such problem and in fact it achieves high tagging accuracies on all four datasets, consistently beating EM by significant margins. When tagging all the 973k word tokens in the Penn Treebank data, it produces an accuracy of 87.1% which is much better than EM (82.3%) run on the same data.

Ravi and Knight (2009) mention that it is possible to interrupt the IP solver and obtain a suboptimal solution faster. However, the IP solver did not return any solution when provided the same amount of time as taken by MIN-GREEDY for any of the data settings. Also, our algorithms were implemented in Python while the IP method employs the best available commercial software package (CPLEX) for solving integer programs.

Figure 4 compares the running time efficiency for the IP method versus MIN-GREEDY method as we scale to larger datasets. Since the IP solver shows variations in running times for different datasets of the same size, we show the average running times for both methods (for each row in the figure, we run a particular method on three different datasets with similar sizes and average the running times). The figure shows that the greedy approach can scale comfortably to large data sizes, and a complete run on the entire Penn Treebank data finishes in just 1485 seconds. In contrast, the IP method does not scale well—on average, it takes 93 seconds to finish on the 24k test (versus 34 seconds for MIN-GREEDY) and on the larger PTB test data, the IP solver runs for

Test set	**Efficiency** (average running time in secs.)	
	IP	MIN-GREEDY
24k test	93.0	34.3
48k test	111.7	64.3
96k test	397.8	93.3
193k test	2347.0	331.0
PTB (973k) test	*	1485.0

Figure 4: Comparison of MIN-GREEDY versus MIN-GREEDY approach in terms of *efficiency* (average running time in seconds) for different data sizes. All the experiments were run on a single machine with a 64-bit, 2.4 GHz AMD Opteron 850 processor.

Method	**Tagging accuracy (%)** when training & testing on:				
	24k	*48k*	*96k*	*193k*	*PTB (973k)*
EM	81.7	81.4	82.8	82.0	82.3
IP	91.6	89.3	89.5	91.6	*
MIN-GREEDY	91.6	88.9	89.4	89.1	87.1

Figure 3: Comparison of tagging accuracies on test data of varying sizes for the task of unsupervised English POS tagging with a dictionary using a 45-tagset. *(* IP method does not scale to large data).*

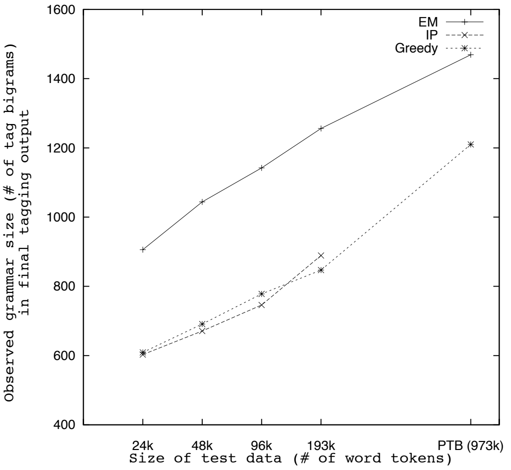


Figure 5: Comparison of observed grammar size (# of tag bigram types) in the final tagging output from EM, IP and MIN-GREEDY.

more than 3 hours without returning a solution.

It is interesting to see that for the 24k dataset, the greedy strategy finds a grammar set (containing only 478 tag bigrams). We observe that MIN-GREEDY produces 452 tag bigrams in the first minimization step (phase 1), and phase 2 adds another 26 entries, yielding a total of 478 tag bigrams in the final minimized grammar set. That is almost as good as the optimal solution (459 tag bigrams from IP) for the same problem. But MIN-GREEDY clearly has an advantage since it runs much faster than IP (as shown in Figure 4). Figure 5 shows a plot with the size of the observed grammar (i.e., number of tag bigram types in the final tagging output) versus the size of the test data for EM, IP and MIN-GREEDY methods. The figure shows that unlike EM, the other two approaches reduce the grammar size considerably and we observe the same trend even when scaling to larger data. Minimizing the grammar size helps remove many spurious tag combinations from the grammar set, thereby yielding huge improvements in tagging accuracy over the EM method (Figure 3). We observe that for the 193k dataset, the final observed grammar size is greater for IP than MIN-GREEDY. This is because the alternating EM steps following the model minimization step add more tag bigrams to the grammar.

Test set	**Average Speedup**	**Optimality Ratio**
24k test	2.7	0.96
48k test	1.7	0.98
96k test	4.3	0.98
193k test	7.1	0.93

Figure 6: *Average speedup* versus *Optimality ratio* computed for the model minimization step (when using MIN-GREEDY over IP) on different datasets.

We compute the optimality ratio of the MIN-GREEDY approach with respect to the grammar size as follows:

$$\text{Optimality ratio} \quad = \quad \frac{\text{Size of IP grammar}}{\text{Size of MIN-GREEDY grammar}}$$

A value of 1 for this ratio implies that the solution found by MIN-GREEDY algorithm is exact. Figure 6 compares the optimality ratio versus average speedup (in terms of running time) achieved in the minimization step for the two approaches. The figure illustrates that our solution is nearly optimal for all data settings with significant speedup.

The MIN-GREEDY algorithm presented here can also be applied to scenarios where the dictionary is incomplete (i.e., entries for all word types are not present in the dictionary) and rare words

Method	Tagging accuracy (%)	Number of unique tag bigrams in final tagging output
EM	83.4	1195
IP	88.0	875
MIN-GREEDY	88.0	880

Figure 7: Results for unsupervised Italian POS tagging with a dictionary using a set of 90 tags.

take on all tag labels. In such cases, we can follow a similar approach as Ravi and Knight (2009) to assign tag possibilities to every unknown word using information from the known word/tag pairs present in the dictionary. Once the completed dictionary is available, we can use the procedure described in Section 5 to minimize the size of the grammar, followed by EM training.

6.2 Italian POS Tagging

We also compare the three approaches for Italian POS tagging and show results.

Data: We use the Italian CCG-TUT corpus (Bos et al., 2009), which contains 1837 sentences. It has three sections: newspaper texts, civil code texts and European law texts from the JRC-Acquis Multilingual Parallel Corpus. For our experiments, we use the POS-tagged data from the CCG-TUT corpus, which uses a set of 90 tags. We created a tag dictionary consisting of 8,733 word/tag pairs derived from the entire corpus (42,100 word tokens). We then created a test set consisting of 926 sentences (21,878 word tokens) from the original corpus. The per-token ambiguity for the test data is about 1.6 tags/token.

Results: Figure 7 shows the results on Italian POS tagging. We observe that MIN-GREEDY achieves significant improvements in tagging accuracy over the EM method and comparable to IP method. This also shows that the idea of model minimization is a general-purpose technique for such applications and provides good tagging accuracies on other languages as well.

7 Conclusion

We present a fast and efficient two-stage greedy minimization approach that can replace the integer programming step in (Ravi and Knight, 2009). The greedy approach finds close-to-optimal solutions for the minimization problem. Our algorithm runs much faster and achieves accuracies close to state-of-the-art. We also evaluate our method on test sets of varying sizes and show that our approach outperforms standard EM by a significant margin. For future work, we would like to incorporate some linguistic constraints within the greedy method. For example, we can assign higher costs to unlikely tag combinations (such as "SYM SYM", etc.).

Our greedy method can also be used for solving other unsupervised tasks where model minimization using integer programming has proven successful, such as word alignment (Bodrumlu et al., 2009).

Acknowledgments

The authors would like to thank Shang-Hua Teng and Anup Rao for their helpful comments and also the anonymous reviewers. This work was jointly supported by NSF grant IIS-0904684, DARPA contract HR0011-06-C-0022 under subcontract to BBN Technologies and DARPA contract HR0011-09-1-0028.

References

Bodrumlu, T., K. Knight, and S. Ravi. 2009. A new objective function for word alignment. In *Proceedings of the NAACL/HLT Workshop on Integer Programming for Natural Language Processing*.

Bos, J., C. Bosco, and A. Mazzei. 2009. Converting a dependency treebank to a categorial grammar treebank for Italian. In *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories (TLT8)*.

Clarke, J. and M. Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research (JAIR)*, 31(4):399–429.

Dempster, A.P., N.M. Laird and D.B. Rubin. 1977. Maximum likelihood from incomplete data via the

EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38.

Garey, M. R. and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. John Wiley & Sons.

Goldberg, Y., M. Adler, and M. Elhadad. 2008. EM can find pretty good HMM POS-taggers (when given a good start). In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL/HLT)*.

Goldwater, Sharon and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Martins, A., N. A. Smith, and E. P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Merialdo, B. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

Punyakanok, V., D. Roth, W. Yih, and D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Ravi, S. and K. Knight. 2008. Attacking decipherment problems optimally with low-order n-gram models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ravi, S. and K. Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing of the AFNLP*.

Smith, N. and J. Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Bringing Active Learning to Life

Ines Rehbein
Computational Linguistics
Saarland University

Josef Ruppenhofer
Computational Linguistics
Saarland University

Alexis Palmer
Computational Linguistics
Saarland University

{rehbein|josefr|apalmer}@coli.uni-sb.de

Abstract

Active learning has been applied to different NLP tasks, with the aim of limiting the amount of time and cost for human annotation. Most studies on active learning have only simulated the annotation scenario, using prelabelled gold standard data. We present the first active learning experiment for Word Sense Disambiguation with human annotators in a realistic environment, using fine-grained sense distinctions, and investigate whether AL can reduce annotation cost and boost classifier performance when applied to a real-world task.

1 Introduction

Active learning has recently attracted attention as having the potential to overcome the knowledge acquisition bottleneck by limiting the amount of human annotation needed to create training data for statistical classifiers. Active learning has been shown, for a number of different NLP tasks, to reduce the number of manually annotated instances needed for obtaining a consistent classifier performance (Hwa, 2004; Chen et al., 2006; Tomanek et al., 2007; Reichart et al., 2008).

The majority of such results have been achieved by simulating the annotation scenario using prelabelled gold standard annotations as a stand-in for real-time human annotation. Simulating annotation allows one to test different parameter settings without incurring the cost of human annotation. There is, however, a major drawback: we do not know whether the results of experiments performed using hand-corrected data carry over to real-world scenarios in which individual human annotators produce noisy annotations. In addition, we do not know to what extent error-prone annotations mislead the learning process. A systematic study of the impact of erroneous annotation on classifier performance in an active learning (AL) setting is overdue. We need to know a) whether the AL approach can really improve classifier performance and save annotation time when applied in a real-world scenario with noisy data, and b) whether AL works for classification tasks with fine-grained or complex annotation schemes and a low inter-annotator agreement.

In this paper we bring active learning to life in the context of frame semantic annotation of German texts within the SALSA project (Burchardt et al., 2006). Specifically, we apply AL methods for learning to assign semantic frames to predicates, following Erk (2005) in treating frame assignment as a Word Sense Disambiguation task. Under our fine-grained annotation scheme, annotators have to deal with a high level of ambiguity, resulting in low inter-annotator agreement for some word senses. This fact, along with the potential for wrong annotation decisions or possible biases from individual annotators, results in an annotation environment in which we get noisy data which might mislead the classifier. A second characteristic of our scenario is that there is no gold standard for the newly annotated data, which means that evaluation is not straightforward. Finally, we have multiple annotators whose decisions on particular instances may diverge, raising the question of which annotations should be used to guide the AL process. This paper thus investigates whether active learning can be successfully applied in a real-world scenario with the particular challenges described above.

Section 2 of the paper gives a short overview of the AL paradigm and some related work, and Section 3 discusses the multi-annotator scenario. In Section 4 we present our experimental design and describe the data we use. Section 5 presents results, and Section 6 concludes.

2 Active Learning

The active learning approach aims to reduce the amount of manual annotation needed to create training data sufficient for developing a classifier with a given performance. At each iteration of the AL cycle, the actual knowledge state of the learner guides the learning process by determining which instances are chosen next for annotation. The main goal is to advance the learning process by selecting instances which provide important information for the machine learner.

In a typical active learning scenario, a small set of manually labelled seed data serves as the initial training set for the classifier (learner). Based on the predictions of the classifier, a large pool of unannotated instances is queried for the next instance (or batch of instances) to be presented to the human annotator (sometimes called the oracle). The underlying active learning algorithm controlling the learning process tries to select the most informative instances in order to get a strong boost in classifier performance. Different methods can be used for determining informativity of instances. We use uncertainty sampling (Cohn et al., 1995) in which "most informative" instances are those for which the classifier has the lowest confidence in its label predictions. The rough intuition behind this selection method is that it identifies instance types which have yet to be encountered by the classifier. The learning process proceeds by presenting the selected instances to the human annotator, who assigns the correct label. The newly-annotated instances are added to the seed data and the classifier is re-trained on the new data set. The newly trained classifier now picks the next instances, based on its updated knowledge, and the process repeats. If the learning process can provide precisely that information which the classifier still needs to learn, a smaller number of instances should suffice to achieve the same accuracy as on a larger training set of randomly selected training examples.

Active learning has been applied to a number of natural language processing tasks like POS tagging (Ringger et al., 2007), NER (Laws and Schütze, 2008; Tomanek and Hahn, 2009), syntactic parsing (Osborne and Baldridge, 2004; Hwa, 2004), Word Sense Disambiguation (Chen et al., 2006; Chan and Ng, 2007; Zhu and Hovy, 2007; Zhu et al., 2008) and morpheme glossing for language documentation (Baldridge and Palmer, 2009). While most of these studies successfully show that the same classification accuracy can be achieved with a substantially smaller data set, these findings are mostly based on simulations using gold standard data.

For our task of Word Sense Disambiguation (WSD), mixed results have been achieved. AL seems to improve results in a WSD task with coarse-grained sense distinctions (Chan and Ng, 2007), but the results of (Dang, 2004) raise doubts as to whether AL can successfully be applied to a fine-grained annotation scheme, where Inter-Annotator Agreement (IAA) is low and thus the consistency of the human annotations decreases. In general, AL has been shown to reduce the cost of annotation when applied to classification tasks where a single human annotator predicts labels for new data points with a reasonable consistency and accuracy. It is not clear whether the same settings can be applied to a multi-annotator environment where IAA is low.

3 Active Learning in a realistic task including multiple annotators

Another possible difference between active learning simulations and real-world scenarios is the multi-annotator environment. In such a setting, two or more annotators assign labels to the same instances, which are then merged to check for conflicting decisions from different annotators. This is standard practise in many annotation projects doing fine-grained semantic annotation with a high level of ambiguity, and it necessitates that all annotators work on the same data set.

Replicating an active learning simulation on hand-corrected data, starting with a fixed set of seed data and fixed parameter settings, using the same algorithm, will always result in the same training set selected from the pool. Human annotators, however, will assign different labels to the same instances, thus influencing the selection of the next instance from the pool. This means that individual annotators might end up with very different sets of annotated data, depending on factors like their interpretation of the annotation guidelines, an implicit bias towards a particular label, or simply errors made during annotation.

There is not much work addressing this problem. (Donmez and Carbonell, 2008) consider modifications of active learning to accommodate variability of annotators. (Baldridge and Palmer, 2009) present a real-world study with human annotators in the context of language documentation. The task consists of producing interlinear glossed text, including morphological and grammatical analysis, and can be described as a sequence labelling task. Annotation cost is measured as the actual time needed for annotation. Among other settings, the authors compare the performance of two annotators with different grades of expertise. The classifier trained on the data set created by the expert annotator in an active learning setting does obtain a higher accuracy on the gold standard. For the non-expert annotator, however, the active learning setting resulted in a lower accuracy than for a classifier trained on a randomly selected data set. This finding suggests that the quality of annotation needs to be high enough for active learning to actually work, and that annotation noise is a problem for AL.

There are two problems arising from this:

1. It is not clear whether active learning will work when applied to noisy data
2. It is not straightforward to apply active learning to a real-world scenario, where low IAA asks for multiple annotators

In our experiment we address these questions by systematically investigating the impact of annotation noise on classifier performance and on the composition of the training set. The next section presents the experimental design and the data used in our experiment.

4 Experimental Design

In the experiment we annotated 8 German causation nouns, namely *Ausgang, Anlass, Ergebnis, Resultat, Grund, Konsequenz, Motiv, Quelle* (outcome, occasion, effect, result, reason, consequence, motive, source of experience). These nouns were chosen because they exhibit a range of difficulty in terms of the number of senses they have in our annotation scheme. They all encode subtle distinctions between different word senses, but some of them are clearly easier to disambiguate than others. For instance, although *Ausgang* has 9 senses, they are easier to distinguish for humans than the 4 senses of *Konsequenz*.

Six annotators participated in the experiment. While all annotators were trained, having at least one year experience in frame-semantic annotation, one of the annotators is an expert with several years of training and working experience in the Berkeley FrameNet Project. This annotator also defined the frames (word senses) used in our experiment.

Prior to the experiment, all annotators were given 100 randomly chosen sentences. After annotating the training data, problematic cases were discussed to make sure that the annotators were familiar with the fine-grained distinctions between word senses in the annotation scheme. The data sets used for training were adjudicated by two of the annotators (one of them being the expert) and then used as a gold standard to test classifier performance in the active learning process.

4.1 Data and Setup

For each lemma we extracted sentences from the Wahrig corpus[1] containing this particular lemma. The annotators had to assign word senses to 300 instances for each target word, split into 6 packages of 50 sentences each. This resulted in 2,400 annotated instances per annotator (14,400 annotated instances in total). The annotation was done

[1] The Wahrig corpus includes more than 113 mio. sentences from German newspapers and magazines covering topics such as politics, science, fashion, and others.

Anlass	**Motiv**	**Konsequenz**	**Quelle**	**Ergebnis / Resultat**	**Ausgang**	**Grund**
Occasion (37)	Motif (47)	Causation (32)	Relational_nat_feat.(3)	Causation (4/10)	Outcome (67)	Causation (24)
Reason (63)	Reason(53)	Level_of_det.(6)	Source_of_getting (14)	Competitive_score(12/36)	Have_leave (4)	Reason (58)
		Response (61)	Source_of_exp. (14)	Decision (11/6)	Portal (21)	Death (1)
		MWE1 (1)	Source_of_info. (56)	Efficacy (2/3)	Outgoing_goods (4)	Part_orientation. (0)
			Well (6)	Finding_out (24/23)	Ostomy (0)	Locale_by_owner(3)
			Emissions_source (7)	Mathematics (1/0)	Origin (5)	Surface_earth (0)
				Operating_result (36/5)	Tech_output (7)	Bottom_layer (0)
				Outcome (10/17)	Process_end (2)	Soil (1)
					Departing (1)	CXN1 (0)
						CXN2 (0)
						MWE1 (0)
						MWE2 (10)
						MWE3 (0)
						MWE4 (3)
						MWE5 (0)
						MWE6 (0)
Fleiss' kappa for the 6 annotators for the 150 instances annotated in the random setting						
0.67	0.79	0.55	0.77	0.63 / 0.59	0.82	0.43

Table 1: 8 causation nouns and their word senses (numbers in brackets give the distribution of word senses in the gold standard (100 sentences); CXN: constructions, MWE: multi-word expressions; note that Ergebnis and Resultat are synonyms and therefore share the same set of frames.)

using a Graphical User Interface where the sentence was presented to the annotator, who could choose between all possible word senses listed in the GUI. The annotators could either select the frame by mouse click or use keyboard shortcuts. For each instance we recorded the time it took the annotator to assign an appropriate label. To ease the reading process the target word was highlighted.

As we want to compare time requirements needed for annotating random samples and sentences selected by active learning, we had to control for training effects which might speed up the annotation. Therefore we changed the annotation setting after each package, meaning that the first annotator started with 50 sentences randomly selected from the pool, then annotated 50 sentences selected by AL, followed by another 50 randomly chosen sentences, and so on. We divided the annotators into two groups of three annotators each. The first group started annotating in the random setting, the second group in the AL setting. The composition of the groups was changed for each lemma, so that each annotator experienced all different settings during the annotation process. The annotators were not aware of which setting they were in.

Pool data For the random setting we randomly selected three sets of sentences from the Wahrig corpus which were presented for annotation to all six annotators. This allows us to compare annotation time and inter-annotator agreement between the annotators. For the active learning setting we randomly selected three sets of 2000 sentences each, from which the classifier could pick new instances during the annotation process. This means that for each trial the algorithm could select 50 instances out of a pool of 2000 sentences. On any given AL trial each annotator uses the same pool as all the other annotators. In an AL simulation with fixed settings and gold standard labels this would result in the same subset of sentences selected by the classifier. For our human annotators, however, due to different annotation decisions the resulting set of sentences is expected to differ.

Sampling method Uncertainty sampling is a standard sampling method for AL where new instances are selected based on the confidence of the classifier for predicting the appropriate label. During early stages of the learning process when the classifier is trained on a very small seed data set, it is not beneficial to add the instances with the lowest classifier confidence. Instead, we use a dynamic version of uncertainty sampling (Rehbein and Ruppenhofer, 2010), based on the confidence of a maximum entropy classifier[2], taking into account how much the classifier has learned so far. In each iteration one new instance is selected from the pool and presented to the oracle. After annotation the classifier is retrained on the new data set. The modified uncertainty sampling results in a more robust classifier performance during early stages of the learning process.

[2] http://maxent.sourceforge.net

	Anlass		Motiv		Konsequenz		Quelle		Ergebnis		Resultat		Ausgang		Grund	
	R	U	R	U	R	U	R	U	R	U	R	U	R	U	R	U
A_1	8.6	9.6	5.9	6.6	10.7	10.5	6.0	4.8	10.5	7.4	10.1	9.6	6.4	10.0	10.2	11.1
A_2	4.4	5.7	4.8	5.9	8.2	9.2	4.9	4.9	6.4	4.4	11.7	8.5	5.1	7.7	9.0	9.3
A_3	9.9	9.2	6.8	6.7	6.8	8.3	7.4	6.1	9.4	7.6	9.0	12.3	7.5	8.5	11.7	10.2
A_4	5.8	4.9	3.6	3.6	9.9	11.3	4.8	3.5	7.9	7.1	9.7	11.1	3.6	4.1	9.9	9.4
A_5	3.0	3.5	3.0	2.6	4.8	4.9	3.8	3.0	6.8	4.8	6.7	6.1	3.1	3.5	6.3	6.0
A_6	5.4	6.3	5.3	4.7	6.7	8.6	5.4	4.6	7.8	6.1	8.7	9.0	6.9	6.6	9.3	8.5
ø	**6.2**	**6.5**	**4.9**	**5.0**	**7.8**	**8.8**	**5.4**	**4.5**	**8.1**	**6.2**	**9.3**	**9.4**	**5.4**	**6.7**	**9.4**	**9.1**
sl	25.8	27.8	27.8	26.0	24.2	25.8	24.9	26.5	25.7	25.2	29.0	35.9	25.5	27.9	26.8	29.7

Table 2: Annotation time (sec/instance) per target/annotator/setting and average sentence length (sl)

5 Results

The basic idea behind active learning is to select the most informative instances for annotation. The intuition behind "more informative" is that these instances support the learning process, so we might need fewer annotated instances to achieve a comparable classifier performance, which could decrease the cost of annotation. On the other hand, "more informative" also means that these instances might be more difficult to annotate, so it is only fair to assume that they might need more time for annotation, which increases annotation cost. To answer the question of whether AL reduces annotation cost or not we have to check a) how long it took the annotators to assign labels to the AL samples compared to the randomly selected instances, and b) how many instances we need to achieve the best (or a sufficient) performance in each setting. Furthermore, we want to investigate the impact of active learning on the distribution of the resulting training sets and study the correlation between the performance of the classifier trained on the annotated data and these factors: the difficulty of the annotation task (assessed by IAA), expertise and individual properties of the annotators.

5.1 Does AL speed up the annotation process when working with noisy data?

Table 2 reports annotation times for each annotator and target for random sampling (R) and uncertainty sampling (U). For 5 out of 8 targets the time needed for annotating in the AL setting (averaged over all annotators) was higher than for annotating the random samples. To investigate whether this might be due to the length of the sentences in the samples, Table 2 shows the average sentence length for random samples and AL samples for each target lemma. Overall, the sentences selected by the classifier during AL are longer (26.2 vs. 28.1 token per sentence), and thus may take the annotators more time to read.[3] However, we could not find a significant correlation (Spearman rank correlation test) between sentence length and annotation time, nor between sentence length and classifier confidence.

The three target lemmas which took longer to annotate in the random setting are *Ergebnis* (result), *Grund* (reason) and *Quelle* (source of experience). This observation cannot be explained by sentence length. While sentence length for *Ergebnis* is nearly the same in both settings, for *Grund* and *Quelle* the sentences picked by the classifier in the AL setting are significantly longer and therefore should have taken more time to annotate. To understand the underlying reason for this we have to take a closer look at the distribution of word senses in the data.

5.2 Distribution of word senses in the data

In the literature it has been stated that AL implicitly alleviates the class imbalance problem by extracting more balanced data sets, while random sampling tends to preserve the sense distribution present in the data (Ertekin et al., 2007). We could not replicate this finding when using noisy data to guide the learning process. Table 3 shows the distribution of word senses for the target lemma *Ergebnis* a) in the gold standard, b) in the random samples, and c) in the AL samples.

The variance in the distribution of word senses in the random samples and the gold standard can

[3] The correlation between sentence length and annotation time is not obvious, as the annotators only have to label one target in each sentence. For ambiguous sentences, however, reading time may be longer, while for the clear cases we do not expect a strong effect.

Ergebnis **Frame**	**gold (%)**	**R (%)**	**U (%)**
Causation	4.0	4.8	3.7
Outcome	10.0	17.8	10.5
Finding_out	24.0	26.2	8.2
Efficacy	2.0	0.8	0.1
Decision	11.0	5.1	3.2
Mathematics	1.0	1.6	0.4
Operating_result	36.0	24.5	66.7
Competitive_score	12.0	19.2	7.2

Table 3: Distribution of frames (word senses) for the lemma *Ergebnis* in the gold standard (100 sentences), in the random samples (R) and AL samples (U) (150 sentences each)

	Random			**Uncertainty**		
	50	100	150	50	100	150
Anlass	0.85	**0.86**	0.85	0.84	0.85	0.84
Motiv	0.57	0.62	0.63	0.64	0.67	**0.70**
Konseq.	0.55	0.59	0.60	0.61	**0.62**	**0.62**
Quelle	0.56	0.53	0.54	0.52	0.52	**0.57**
Ergebnis	0.39	**0.42**	0.41	0.39	0.37	0.38
Resultat	0.31	0.35	**0.37**	0.32	0.34	0.34
Ausgang	0.67	0.69	0.69	0.68	0.72	**0.74**
Grund	**0.48**	0.47	0.47	0.47	0.44	**0.48**

Table 4: Avg. classifier performance (acc.) over all annotators for the 8 target lemmas when training on 50, 100 and 150 annotated instances for random samples and uncertainty samples

be explained by low inter-annotator agreement caused by the high level of ambiguity for the target lemmas. The frame distribution in the data selected by uncertainty sampling, however, crucially deviates from those of the gold standard and the random samples. A disproportionately high 66% of the instances selected by the classifier have been assigned the label Operating_result by the human annotators. This is the more surprising as this frame is fairly easy for humans to distinguish.

The classifier, however, proved to have serious problems learning this particular word sense and thus repeatedly selected more instances of this frame for annotation. As a result, the distribution of word senses in the training set for the uncertainty samples is highly skewed, having a negative effect on the overall classifier performance. The high percentage of instances of the "easy-to-decide" frame Operating_result explains why the instances for *Ergebnis* took less time to annotate in the AL setting. Thus we can conclude that annotating the same number of instances on average takes more time in the AL setting, and that this effect is not due to sentence length.

5.3 What works, what doesn't, and why

For half of the target lemmas *(Motiv, Konsequenz, Quelle, Ausgang),* we did obtain best results in the AL setting (Table 4). For *Ausgang* and *Motiv* AL gives a substantial boost in classifier performance of 5% and 7% accuracy, while the gains for *Konsequenz* and *Quelle* are somewhat smaller with 2% and 1%, and for *Grund* the highest accuracy was reached on both the AL and the random sample.

Figure 1 (top row) shows the learning curves for *Resultat*, our worst-performing lemma, for the classifier trained on the manually annotated samples for each individual annotator. The solid black line represents the majority baseline, obtained by assigning the most frequent word sense in the gold standard to all instances. For both random and AL settings, results are only slightly above the baseline. The curves for the AL setting show how erroneous decisions can mislead the classifier, resulting in classifier accuracy below the baseline for two of the annotators, while the learning curves for these two annotators on the random samples show the same trend as for the other 4 annotators.

For *Konsequenz* (Figure 1, middle), the classifier trained on the AL samples yields results over the baseline after around 25 iterations, while in the random sampling setting it takes at least 100 iterations to beat the baseline. For *Motiv* (Figure 1, bottom row), again we observe far higher results in the AL setting. A possible explanation for why AL seems to work for *Ausgang, Motiv* and *Quelle* might be the higher IAA[4] (κ 0.825, 0.789, 0.768) as compared to the other target lemmas. This, however, does not explain the good results achieved on the AL samples for *Konsequenz,* for which IAA was quite low with κ 0.554.

Also startling is the fact that AL seems to work particularly well for one of the annotators (A_6, Figure 1) but not for others. Different possible explanations come to mind: (a) the accuracy of the annotations for this particular annotator, (b) the

[4] IAA was computed on the random samples, as the AL samples do not include the same instances.

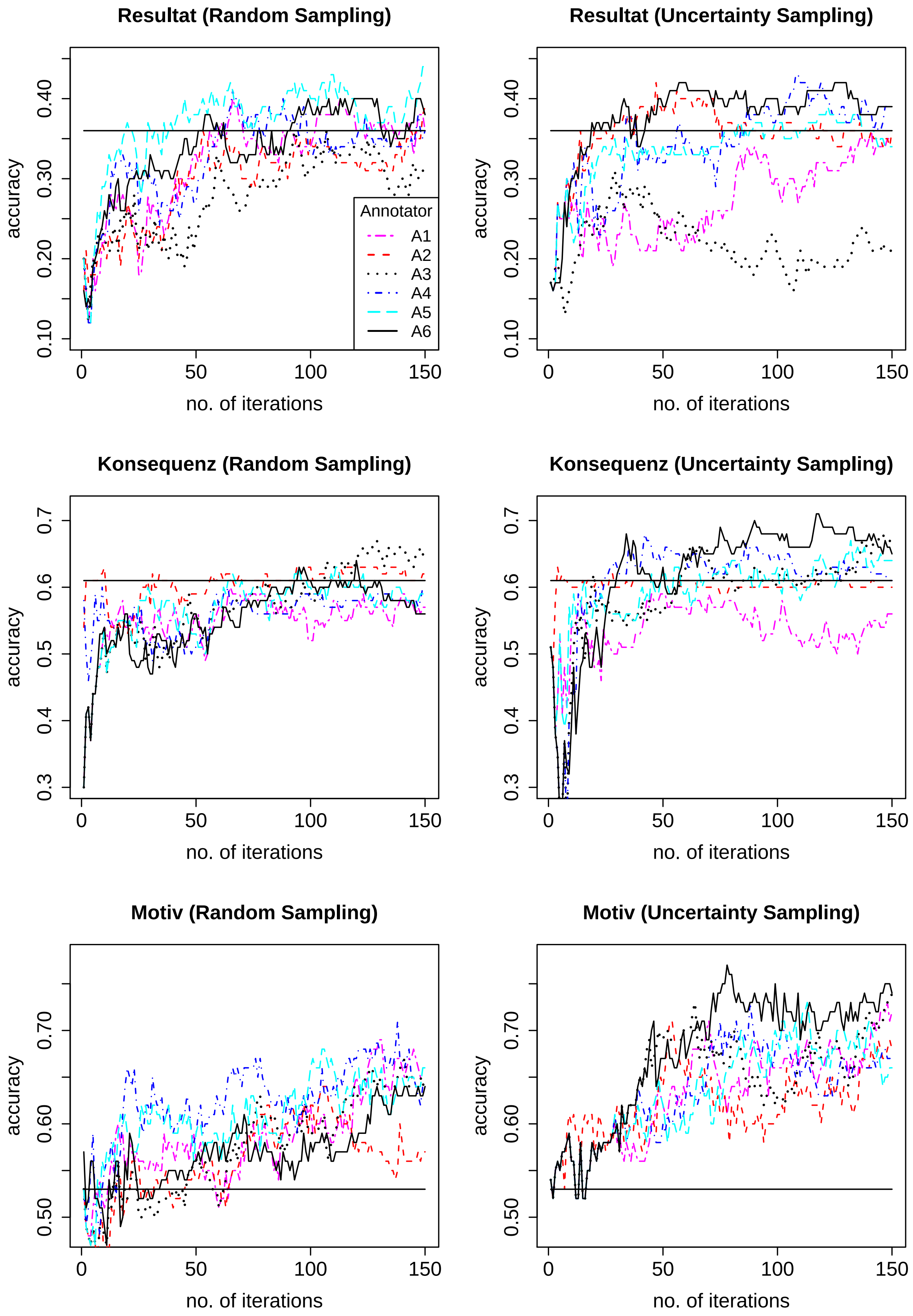


Figure 1: Active learning curves for Resultat, Konsequenz and Motiv (random sampling versus uncertainty sampling; the straight black line shows the majority baseline)

Konsequenz	A1	A2	A3	A4	A5	A6
human	0.80	0.72	0.89	0.73	0.89	0.76
maxent	0.60	0.63	0.67	0.60	0.63	0.64

Table 5: Acc. for human annotators against the adjudicated random samples and for the classifier

instances selected by the classifier based on the annotation decisions of the individual annotators, and (c) the distribution of frames in the annotated training sets for the different annotators.

To test (a) we evaluated the annotated random samples for *Konsequenz* for each annotator against the adjudicated gold standard. Results showed that there is no strong correlation between the accuracy of the human annotations and the performance of the classifier trained on these annotations. The annotator for whom AL worked best had a medium score of 0.76 only, while the annotator whose annotations were least helpful for the classifier showed a good accuracy of 0.80 against the gold standard.

Next we tested (b) the impact of the particular instances in the AL samples for the individual annotators on classifier performance. We took all instances in the AL data set from A_6, whose annotations gave the greatest boost to the classifier, removed the frame labels and gave them to the remaining annotators for re-annotation. Then we trained the classifier on each of the re-annotated samples and compared classifier performance. Results for 3 of the remaining annotators were in the same range or even higher than the ones for A_6 (Figure 2). For 2 annotators, however, results remained far below the baseline.

This again shows that the AL effect is not directly dependent on the accuracy of the individual annotators, but that particular instances are more informative for the classifier than others. Another crucial point is (c) the distribution of frames in the samples. In the annotated samples for A_1 and A_2 the majority frame for *Konsequenz* is Causation, while in the samples for the other annotators Response was more frequent. In our test set Response also is the most frequent frame, therefore it is not surprising that the classifiers trained on the samples of A_3 to A_6 show a higher performance. This means that high-quality annotations (identified by IAA) do not necessarily provide the information from which the classifier benefits most, and that in a realistic annotation task addressing the class imbalance problem (Zhu and Hovy, 2007) is crucial.

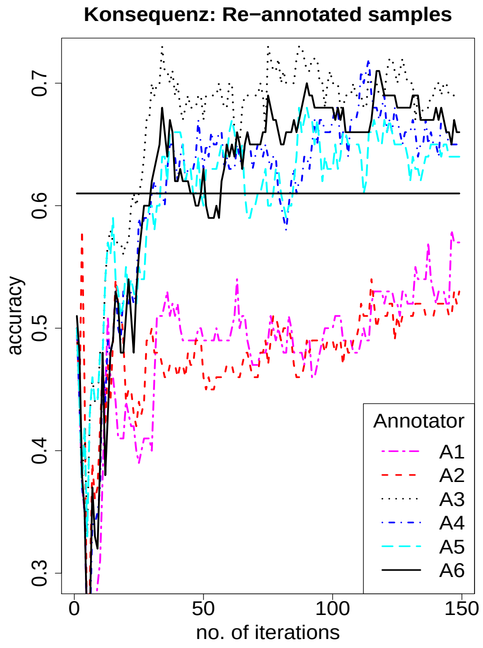


Figure 2: Re-annotated instances for Konsequenz (AL samples from annotator A_6)

6 Conclusions

We presented the first experiment applying AL in a real-world scenario by integrating the approach in an ongoing annotation project. The task and annotation environment pose specific challenges to the AL paradigm. We showed that annotation noise caused by biased annotators as well as erroneous annotations mislead the classifier and result in skewed data sets, and that for this particular task no time savings are to be expected when applied to a realistic scenario. Under certain conditions, however, classifier performance can improve over the random sampling baseline even on noisy data and thus yield higher accuracy in the active learning setting. Critical features which seem to influcence the outcome of AL are the amount of noise in the data as well as the distribution of frames in training- and test sets. Therefore, addressing the class imbalance problem is crucial for applying AL to a real annotation task.

Acknowledgments

This work was funded by the German Research Foundation DFG (grant PI 154/9-3 and the MMCI Cluster of Excellence).

References

Baldridge, Jason and Alexis Palmer. 2009. How well does active learning actually work?: Time-based evaluation of cost-reduction strategies for language documentation. In *Proceedings of EMNLP 2009*.

Burchardt, Aljoscha, Katrin Erk, Anette Frank, Andrea Kowalski, Sebastian Padó, and Manfred Pinkal. 2006. The salsa corpus: a german corpus resource for lexical semantics. In *Proceedings of LREC-2006*.

Chan, Yee Seng and Hwee Tou Ng. 2007. Domain adaptation with active learning for word sense disambiguation. In *Proceedings of ACL-2007*.

Chen, Jinying, Andrew Schein, Lyle Ungar, and Martha Palmer. 2006. An empirical study of the behavior of active learning for word sense disambiguation. In *Proceedings of NAACL-2006*, New York, NY.

Cohn, David A., Zoubin Ghahramani, and Michael I. Jordan. 1995. Active learning with statistical models. In Tesauro, G., D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press.

Dang, Hoa Trang. 2004. *Investigations into the role of lexical semantics in word sense disambiguation*. PhD dissertation, University of Pennsylvania, Pennsylvania, PA.

Donmez, Pinar and Jaime G. Carbonell. 2008. Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of CIKM08*.

Erk, Katrin. 2005. Frame assignment as word sense disambiguation. In *Proceedings of the IWCS-6*.

Ertekin, Şeyda, Jian Huang, L'eon Bottou, and Lee Giles. 2007. Learning on the border: active learning in imbalanced data classification. In *Proceedings of CIKM '07*.

Hwa, Rebecca. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276.

Laws, Florian and Heinrich Schütze. 2008. Stopping criteria for active learning of named entity recognition. In *Proceedings of Coling 2008*.

Osborne, Miles and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. In *Proceedings of HLT-NAACL 2004*.

Rehbein, Ines and Josef Ruppenhofer. 2010. Theres no data like more data? revisiting the impact of data size on a classification task. In *Proceedings of LREC-07, 2010*.

Reichart, Roi, Katrin Tomanek, Udo Hahn, and Ari Rappoport. 2008. Multi-task active learning for linguistic annotations. In *Proceedings of ACL-08: HLT*.

Ringger, Eric, Peter Mcclanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, and Deryle Lonsdale. 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proceedings of ACL Linguistic Annotation Workshop*.

Tomanek, Katrin and Udo Hahn. 2009. Reducing class imbalance during active learning for named entity annotation. In *Proceedings of the 5th International Conference on Knowledge Capture*, Redondo Beach, CA.

Tomanek, Katrin, Joachim Wermter, and Udo Hahn. 2007. An approach to text corpus construction which cuts annotation costs and maintains corpus reusability of annotated data. In *Proceedings of EMNLP-CoNLL 2007*.

Zhu, Jingbo and Ed Hovy. 2007. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *Proceedings of EMNLP-CoNLL 2007*.

Zhu, Jingbo, Huizhen Wang, Tianshun Yao, and Benjamin K. Tsou. 2008. Active learning with sampling by uncertainty and density for word sense disambiguation and text classification. In *Proceedings of Coling 2008*.

Computing EM-based Alignments of Routes and Route Directions as a Basis for Natural Language Generation

Michael Roth and **Anette Frank**
Department of Computational Linguistics
Heidelberg University
{mroth,frank}@cl.uni-heidelberg.de

Abstract

Route directions are natural language (NL) statements that specify, for a given navigational task and an automatically computed route representation, a sequence of actions to be followed by the user to reach his or her goal. A corpus-based approach to generate route directions involves (i) the selection of elements along the route that need to be mentioned, and (ii) the induction of a mapping from route elements to linguistic structures that can be used as a basis for NL generation.

This paper presents an Expectation-Maximization (EM) based algorithm that aligns geographical route representations with semantically annotated NL directions, as a basis for the above tasks. We formulate one basic and two extended models, the latter capturing special properties of the route direction task. Although our current data set is small, both extended models achieve better results than the simple model and a random baseline. The best results are achieved by a combination of both extensions, which outperform the random baseline and the simple model by more than an order of magnitude.

1 Introduction

The purpose of route directions is to inform a person, who is typically not familiar with his current environment, of how to get to a designated goal. Generating such directions poses difficulties on various conceptual levels such as planning the route, selecting landmarks (i.e., recognizable buildings or structures) and splitting the task into appropriate single instructions of how to navigate along the route using the selected landmarks as reference points.

Previously developed natural language generation (NLG) systems make use of simple heuristics for the task of content selection for route directions (Dale et al., 2005; Roth and Frank, 2009). In our work, we aim for a corpus-based approach that can be flexibly modeled after natural, human-produced directions for varying subtasks (e.g., indoor vs. outdoor navigation), and that facilitates multilingual extensions. By employing salient landmarks and allowing for variation in NL realization, such a system is expected to generate natural sounding directions that are easier to memorize and easier to follow than directions given by a classical route planner or navigation system.

NLG for route directions crucially differs from other generation tasks such as document summarization (Mani, 2001) in that the selection and ordering of input structures for language generation is heavily situation-dependent, i.e., dependent on the specific properties of a given route to be followed.

In line with a corpus-based NLG approach, we propose to automatically align geographical route representations as produced by a route planner with an annotated corpus of NL directions given by humans for the respective routes. The induced alignments will (i) serve to identify which elements of a route to select for verbalization, and (ii) deliver correspondences between route segments and linguistic input structures that can be used as a basis for statistical NL generation. We investi-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 958–966,
Beijing, August 2010

gate a minimally supervised method for inducing such alignments to ensure maximal flexibility for adaptations to different scenarios.

The remainder of this paper is structured as follows: In Section 2 we discuss related work. Section 3 introduces the task, and the representation formats and resources we use. Section 4 introduces a basic Expectation-Maximization model and two extensions for the alignment task. Section 5 outlines the experiments and presents the evaluation results. In Section 6 we conclude and discuss future work.

2 Related Work

Various aspects of route directions have been subject of research in computational linguistics, ranging from instructional dialogues in MapTask (Anderson et al., 1991) to recent work on learning to follow route directions (Vogel and Jurafsky, 2010). However, little work has been done on generating NL directions based on data from Geographical Information Systems (Dale et al., 2005; Roth and Frank, 2009).

NLG systems are typically realized as pipeline architectures (Reiter and Dale, 2000). As a first step, they compute a set of messages that represent the information to be conveyed to a user, given a specific communicative task (*Content Selection*). Selecting appropriate content for a task can be defined heuristically, by manually crafted rules or by learning content selection rules automatically from corpus data. Previous work by Dale et al. (2005) and Roth and Frank (2009) on generating NL directions used hand-crafted heuristics. Duboue and McKeown (2003) were the first to model content selection as a machine learning task, in which selection rules are induced from pairs of human-written text and associated sets of database entries. They induce baseline selection rules from exact matches of NL expressions with database entries; in addition, class-based rules are computed by matching database entry types against NL expressions, using statistical co-occurrence clusters. Barzilay and Lapata (2005) incorporate the interplay between multiple events and entities when learning content selection rules using a special link function.

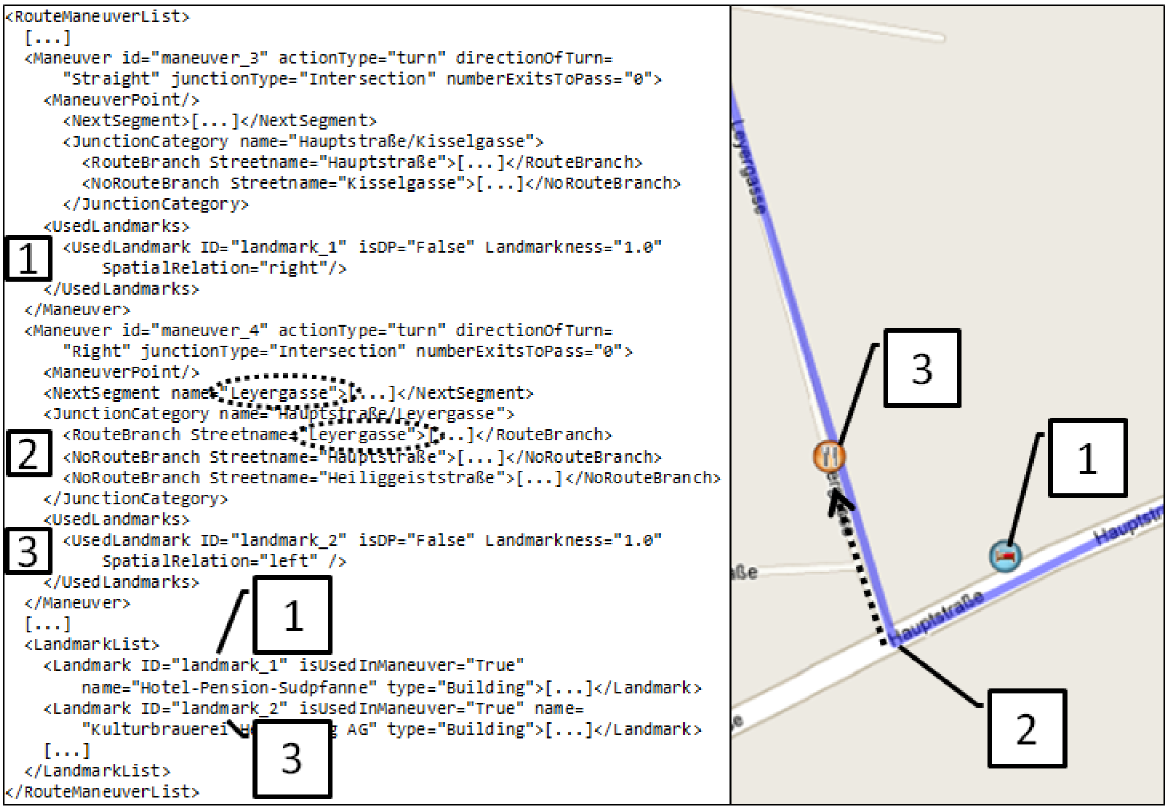
Recent work by Liang et al. (2009) focuses on modeling grounded language, by aligning real-world representations with NL text that references corresponding world states. They show how a generative model can be used to segment text into utterances and to identify relevant facts with minimal supervision. Both tasks are handled jointly in a unified framework by training a hierarchical semi-Markov model on pairs of text and world states, thereby modeling sequencing effects in the presentation of facts. While their work is not primarily concerned with NLG, the learned correspondences and their probabilities could be applied to induce content selection rules and linguistic mappings in a NLG task. The approach is shown to be effective in scenarios typical for NLG settings (weather forecasts, RoboCup sportscasting, NFL recaps) that differ in the amount of available data, length of textual descriptions, and density of alignments.

In the following, we will adapt ideas from their EM-based approach to align (segments of) route representations and NL route directions in a minimally supervised manner. We will investigate increasingly refined models that are tailored to the nature of our task and underlying representations. In particular, we extend their approach by exploiting semantic markup in the NL direction corpus.

3 Aligning Routes and Directions

In this work we explore the possibility of using an implementation of the EM algorithm (Dempster et al., 1977) to learn correspondences between (segments of) the geographical representation of a route and linguistic instructions of how to follow this route in order to arrive at a designated goal. We are specifically interested in identifying which parts of a route are realized in natural language and which kinds of semantic constructions are used to express them.

As a data source for inducing such correspondences we use a parallel corpus of route representations and corresponding route directions that were collected in a controlled experiment for navigation in an urban street network (cf. Schuldes et al. (2009)). For the alignment task, the routes were compiled to a specification format that has been realized in an internal version of an online route planner. Figure 1 displays the route rep-



```
<RouteManeuverList>
  [...]
  <Maneuver id="maneuver_3" actionType="turn" directionOfTurn=
      "Straight" junctionType="Intersection" numberExitsToPass="0">
    <ManeuverPoint/>
      <NextSegment>[...]</NextSegment>
      <JunctionCategory name="Hauptstraße/Kisselgasse">
        <RouteBranch Streetname="Hauptstraße">[...]</RouteBranch>
        <NoRouteBranch Streetname="Kisselgasse">[...]</NoRouteBranch>
      </JunctionCategory>
    <UsedLandmarks>
      <UsedLandmark ID="landmark_1" isDP="False" Landmarkness="1.0"
          SpatialRelation="right"/>
    </UsedLandmarks>
  </Maneuver>
  <Maneuver id="maneuver_4" actionType="turn" directionOfTurn=
      "Right" junctionType="Intersection" numberExitsToPass="0">
    <ManeuverPoint/>
    <NextSegment name="Leyergasse">[...]</NextSegment>
    <JunctionCategory name="Hauptstraße/Leyergasse">
      <RouteBranch Streetname="Leyergasse">[...]</RouteBranch>
      <NoRouteBranch Streetname="Hauptstraße">[...]</NoRouteBranch>
      <NoRouteBranch Streetname="Heiliggeiststraße">[...]</NoRouteBranch>
    </JunctionCategory>
    <UsedLandmarks>
      <UsedLandmark ID="landmark_2" isDP="False" Landmarkness="1.0"
          SpatialRelation="left" />
    </UsedLandmarks>
  </Maneuver>
  [...]
  <LandmarkList>
    <Landmark ID="landmark_1" isUsedInManeuver="True"
        name="Hotel-Pension-Sudpfanne" type="Building">[...]</Landmark>
    <Landmark ID="landmark_2" isUsedInManeuver="True" name=
        "Kulturbrauerei H[illegible]g AG" type="Building">[...]</Landmark>
    [...]
  </LandmarkList>
</RouteManeuverList>
```

Figure 1: A (partial) route representation of the route segment displayed on the right.

resentation for a small route segment (a junction connecting 'Hauptstraße' and 'Leyergasse'). The corresponding part of a NL route direction is displayed in Figure 2. The route representation and the NL direction share some common concepts: For example, both contain references to a landmark called "Sudpfanne" (marked as [1]) and a street named "Leyergasse" (marked as [2]). Using pairs of route representations and directions, we aim to automatically induce alignments between such correspondences. In the following we describe our data in more detail.

3.1 Route Representation Format

The route representation format we use (illustrated in Figure 1) is an extended version of the *OpenGIS Location Service (OpenLS) Implementation Standards*, a set of XML-based representations specified by the Open Geospatial Consortium[1]. Previous approaches on extending the latter with landmarks in an interoperable way have been presented by Neis and Zipf (2008). The representation format of our data has been developed in close collaboration with researchers from Geoinformatics at Heidelberg University[2] and adopts ideas previously proposed in the *Cognitive OpenLS* specification by Hansen et al. (2006). The resulting specification will be implemented in an extended (internal) version of the online route planner OpenRouteService.org.

Our work revolves around two kinds of elements in this format: so-called *maneuvers*, i.e., elements that describe a decision point including the required action and the following route segment, and *landmarks* that occur along the route. For the alignment task we focus on the following types of attributes that are part of the XML specification, specified here as **Attribute (Element)**:

directionOfTurn (Maneuver) – the direction of movement for the current maneuver, i.e., "left", "right" or "straight"

[1] http://www.opengeospatial.org/standards/is

[2] Chair of GIScience, Alexander Zipf, http://www.geog.uni-heidelberg.de/lehrstuehle/gis/

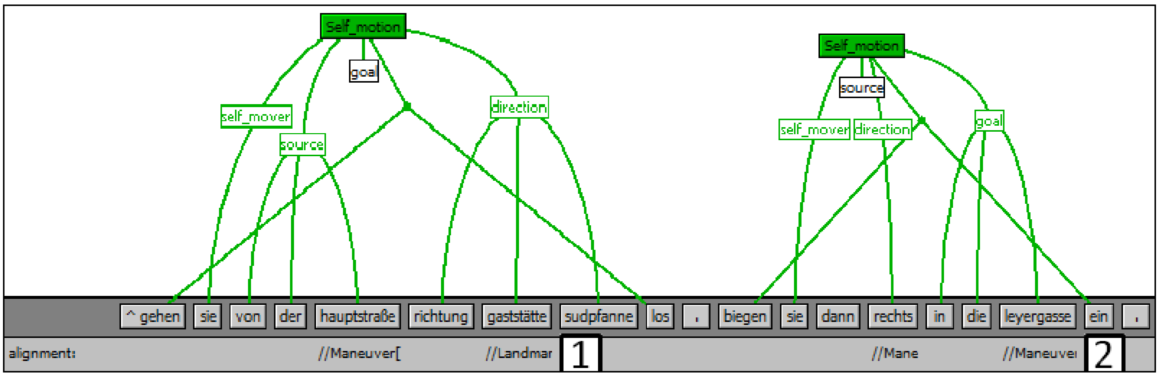


Figure 2: Directions for the route segment displayed in Figure 1 annotated with frame-semantic markup and alignment information. The directions translate to "You start walking from Hauptstraße towards Gaststätte Sudpfanne, then you turn right onto Leyergasse"

junctionType (Maneuver) – the type of junction at the current maneuver, e.g., "intersection", "crossing"

name (JunctionCategory) – the name of the junction at the current maneuver, e.g., "Hauptstraße/Leyergasse"

name (NextSegment) – the name of the street of the next route segment, e.g., "Hauptstraße"

streetName (RouteBranch) – the street name of a branch along which the route continues, e.g., "Leyergasse"

streetName (NoRouteBranch) – the street name of a branch that is not part of the route, e.g., "Kisselgasse"

name (Landmark) – the name of a landmark, e.g., "Hotel Sudpfanne"

spatialRelation (UsedLandmark) – the spatial relation between a landmark and the current maneuver, e.g., "left", "right", "before"

3.2 A Parallel Corpus of Route Directions

The corpus of route directions used in this work is a subset of the data collected by Schuldes et al. (2009) in a desk-based experiment. To elicit NL route directions, subjects were shown a web application that guided them along a route by means of a 2D animation. Subsequently they had to write NL route directions in German for the shown routes. The subjects were allowed to use all information displayed by the web application: named places, buildings, bridges and street names, etc. The resulting directions were POS-tagged with TreeTagger (Schmid, 1997), dependency-parsed with XLE (Maxwell and Kaplan, 1993), and manually revised. Additionally, we annotated frame-semantic markup (Fillmore et al., 2003) and gold standard alignments to the route representation using the SALTO annotation tool (Burchardt et al., 2006).

Frame semantic markup. The texts are annotated with an inventory of 4 frames relevant for directions (SELF_MOTION, PERCEPTION, BEING_LOCATED, LOCATIVE_RELATION), with semantic roles (*frame elements*) such as DIRECTION, GOAL, PATH, LOCATION. Figure 2 illustrates a typical example for the use of the SELF_MOTION frame, once with the elements SOURCE and DIRECTION, and once with the elements DIRECTION and GOAL. Our alignment model uses the frame semantic annotation as structuring information.

Gold standard alignments. For evaluation we constructed gold alignments. We asked two annotators to align text parts with corresponding attributes in the respective route representation[3]. The information about corresponding attributes was added to a single word by manually insert-

[3] The alignments have not been double annotated, hence no measure for inter-annotator agreement can be provided.

	#S	#W	#FE	#aligned FE
avg. per direction	8	98	28	14 (50%)
overall	412	5298	1519	750

Table 1: Corpus statistics: number of sentences (S), words (W), frame elements (FE) and alignments.

	#attributes	#aligned attr.
avg. per route	115	14 (12%)
overall	921	

Table 2: Corpus statistics: total number and percentage of relevant attribute alignments.

ing XPATH expressions that unambiguously refer to the aligned attribute in the route representation format. For learning the alignment model, the annotations were spread to all words in the span of the respective frame element.

Corpus statistics. We made use of a corpus of 54 NL directions collected for 8 routes in an urban street network. Tables 1 and 2 give some statistics about the number of words (W) and frame elements (FE) in the parallel corpus. Comparing the total number of relevant attributes (as listed in Section 3.1) and attributes annotated in the gold alignments (aligned attr.) we note that only 12% are actually mentioned in NL directions. Thus it is necessary to select the most salient attributes to avoid the generation of overly redundant text.

4 Alignment Model

For the induction of alignments between (parts of) route structures and semantic representations, we adopt ideas from the models presented in Liang et al. (2009) (cf. Section 2).

We start from a basic *frame alignment model.* It specifies a conditional probability distribution $\mathrm{p}(f|a)$ for the alignment to a frame element f of type f_t (e.g., source, goal, direction) in the frame-semantic annotation layer given an attribute a of type a_t (e.g., streetName, directionOfTurn) in the route representation format. Note that this model does not take into account the actual value a_v of the attribute a nor the words that are annotated as part of f. We assume that the frame annotation represents a reliable segmentation for this alignment. This allows us to omit modeling segmentation explicitly.

As extensions to the basic frame alignment model, we specify two further models that capture properties that are specific to the task of direction alignment. As route directions are typically presented in a linear order with respect to the route, we incorporate an additional *distance model* λ in our alignment. We further account for *word choice* within a frame element as an additional factor. The word choice model $\mathrm{p}(w|a)$ will exploit attribute type and value information in the route representations that are reflected in word choice in the linguistic instructions. Both extensions are inspired by and share similarities with models that have been successfully applied in work on text alignment for the task of machine translation (Vogel et al., 1996; Tiedemann, 2003).

Our full model is a distribution over frame elements f and words w that factorizes the three above mentioned parts under the assumption of independence between each component and each attribute:

$$\mathrm{p}(f,w|a) = \mathrm{p}(f|a)\lambda(\mathrm{dist}(f,a))\,\mathrm{p}(w|a) \quad (1)$$

The individual models are described in more detail in the following subsections.

4.1 Frame Alignment Model

This basic frame alignment model specifies the probabilities $\mathrm{p}(f|a)$ for aligning an attribute a of type a_t (i.e., one of the types listed in Section 3.1) to a frame element f labeled as type f_t. This alignment model is initialized as a uniform distribution over f and trained using a straight-forward implementation of the EM algorithm, following the well-known IBM Model 1 for alignment in machine translation (Brown et al., 1993). The expectation step (E-step) computes expected counts given occurrences of f_t and a_t under the assumption that all alignments are independent 1:1 correspondences:

$$\mathrm{count}(f_t,a_t) = \frac{\sum_{\{\langle f',a'\rangle|f'_t=f_t\wedge a'_t=a_t\}}\mathrm{p}(f'|a')}{\sum_{\{\langle f',y\rangle|f'_t=f_t\}}\mathrm{p}(f'|y)} \quad (2)$$

The probabilities are re-estimated to maximize the overall alignment probability by normalizing the estimated counts (M-step):

$$\mathrm{p}(f|a) = \frac{\mathrm{count}(f_t, a_t)}{\sum_x \mathrm{count}(x_t, a_t)} \tag{3}$$

4.2 Distance Model

We hypothesize that the order of route directions tends to be consistent with the order of maneuvers encoded by the route representation. We include this information in our alignment model by defining a distance measure $\mathrm{dist}(f, a)$ between the relative position of a frame element f in the text and the relative position of an attribute a in the route representation. The probabilities are specified in form of a distance distribution $\lambda(i)$ over normalized distances $i \in [0 : 1]$ and learned during EM training. The weights are initialized as a uniform distribution and re-estimated in each M-step by normalizing the estimated counts:

$$\lambda(i) = \frac{\sum_{\{\langle x,y\rangle |\, \mathrm{dist}(x,y)=i\}} \mathrm{count}(x, y)}{\sum_{\{\langle x,y\rangle\}} \mathrm{count}(x, y)} \tag{4}$$

4.3 Word Choice Model

We define a word choice model for word usage within a frame element. This additional factor is necessary to distinguish between various occurrences of the same type of frame element with different surface realizations. For example, assuming that the frame alignment model correctly aligns directionOfTurn attributes to a frame element of type DIRECTION, the word choice model will provide an additional weight for the alignment between the value of an attribute (e.g., “left”) and the corresponding words within the frame element (e.g., “links”). Similarly to the word choice model within fields in (Liang et al., 2009), our model specifies a distribution over words given the attribute a. Depending on whether the attribute is typed for strings or categorial values, two different distributions are used.

String Attributes. For string attributes, we determine a weighting factor based on the longest common subsequence ratio (LCSR). The reason for using this measure is that we want to allow for spelling variants and the use of synonymous common nouns in the description of landmarks and street names (e.g., “Main St.” vs. “Main Street”, “Texas Steakhouse” vs. “Texas Restaurant”). The weighting factor $\mathrm{p}_{\mathrm{str}}(w|a)$ for an alignment pair $\langle f, a\rangle$ is a constant in the E-step and is calculated as the LCSR of the considered attribute value a_v and the content words $w = \mathrm{cw}(f)$ in an annotated frame element f divided by the sum over the LCSR values of all alignment candidates for a:

$$\mathrm{p}_{\mathrm{str}}(w|a) = \frac{\mathrm{LCSR}(a_v, w)}{\sum_x \mathrm{LCSR}(a_v, \mathrm{cw}(x))} \tag{5}$$

Categorial Attributes. We define categorial attributes as attributes that can only take a finite and prescribed set of values. For these we do not expect to find matching strings in NL directions as the attribute values are defined independently of the language in use (e.g., values for directionOfTurn are “left”, “right” and “straight”. However, the directions in our data set are in German, thus containing the lexemes “links”, “rechts” und “geradeaus” instead). As the set of values $\{a_v \in \mathbb{D}_{a_t}\}$ for a categorial attribute type a_t is finite, we can define and train probability distributions over words for each of them during EM training. The models are initialized as uniform distributions and are used as a weighting factor in the E-Step. We re-calculate the parameters of a distribution $\mathrm{p}_{cat}(w|a)$ in each EM iteration by normalizing the estimated counts during M-step:

$$\mathrm{p}_{\mathrm{cat}}(w|a) = \frac{\mathrm{count}(a_v, w)}{\sum_x \mathrm{count}(a_v, x)} \tag{6}$$

5 Experiments and Results

5.1 Setting

We test the performance of different combinations of these EM-based models on our data, starting from a simple baseline model (**EM**), combined with the distance (**EM+dst**) and word choice models (**EM+wrd**) and finally the full model (**Full**). We perform additional experiments to examine the impact of different corpus sizes and an alignment threshold (**+thld**).

EM is a baseline model that consists of a simple EM implementation for aligning attributes and frame elements (equation (3)).

EM+dst consists of the simple EM model and the additional distance factor (equation (4)).

Model	P (+thld)	R (+thld)	F_1 (+thld)
Random	2.7 (2.7)	3.9 (3.9)	3.2 (3.2)
EM	2.0 (3.6)	2.9 (3.7)	2.34 (3.6)
EM+dst	7.3 (11.6)	10.8 (11.7)	8.7 (11.6)
EM+wrd	26.8 (36.3)	39.5 (35.5)	32.0 (35.9)
Full	28.9 (38.9)	42.5 (37.9)	34.4 (38.4)

Table 3: Precision (P), Recall (R) and F_1 measure results with and without threshold (+thld) on the alignment task (all numbers in percentages).

EM+wrd consists of the simple EM model with the word choice model (equations (5) and (6), respectively).

Full is the full alignment model including distance and word choice as described in Section 4 (cf. equation (1)).

We use the data set described in Section 3. The predictions made by the different models are evaluated against the gold standard alignments (cf. Tables 1 and 2). We run a total number of 30 iterations[4] of EM training on the complete data set to learn the parameters of the probability distributions. From the set of all possible 1-to-1 alignments, we select the most probable alignments according to the model in a way that no attribute and no frame element is aligned twice.

5.2 Results

We measure precision as the number of predicted alignments also annotated in the gold standard divided by the total number of alignments generated by our model. Recall is measured as the number of correctly predicted alignments divided by the total number of alignment annotations. As baselines we consider a random baseline (obtained from the average results measured over 1,000 random alignment runs) and the simple EM model.

The results in Table 3 show that the simple EM model performs below the random baseline. The individual extended models achieve significant improvement over the simple model and the random baseline. While the distance model has a smaller impact, the influence of the word choice model is considerable. Applying the full model yields further performance gains. We note that for all models recall is higher compared to precision.

# directions	Precison	Recall	F_1
1	28.94%	42.31%	34.38%
2	29.04%	41.90%	34.31%
3	29.01%	42.18%	34.38%
4	28.75%	41.81%	34.07%
5	29.36%	42.69%	34.79%
6	30.18%	43.91%	35.77%

Table 4: Average results when using only a specific number of directions for each route with the model Full (-thld).

One of the reasons for this phenomenon may be that the EM-based models align as many attributes as possible to frame elements in the route directions. In our gold standard, however, only around 12% of all relevant attributes correspond to frame elements in the route directions (cf. Section 3.2). We estimate this quota from a part of the corpus and use it as an alignment threshold, i.e., for evaluation we select the best alignments proposed by the models, until we reach the threshold. With this we achieve a F_1 measure of 38.40% in a 6-fold cross validation test. This represents an improvement of 3.97 points and considerably boosts precision, yielding overall balanced precision (38.90%) and recall (37.92%).

A general problem of the current setup is the small amount of available data. With a total of 54 route directions, the data consists of 6 to 8 directions for each route. We compute a learning curve by using only exactly 1 to 6 directions per route to examine whether performance improves with increasing data size. The results are computed as an average over multiple runs with different data partitions (see Table 4). The results indicate small but consistent improvements with increasing data sizes, however, the differences are minimal. Thus we are not able to conclude at this point whether performance increases are possible with the addition of more data.

5.3 Error Analysis

In an error analysis on the results of the full model, we found that 363 out of 784 (46%) misalign-

[4] This number was determined by experiments as a general heuristics.

ments are related to attributes not aligned in our gold standard. This is due to the fact that not all relevant attributes are realized in natural language directions. By addressing this problem in the model Full+threshold, we are able to reduce these errors, as evidenced by a gain of almost 10 points in precision and 4 points in F_1 measure.

We further observe that the word choice model does not correctly reflect the distribution of categorial attributes in the parallel corpus. In the data, we observe that humans often aggregate multiple occurrences of the same attribute value into one single utterance. An example of such a phenomenon can be seen with the attribute type 'directionOfTurn': Even though "straight" is the most common value for this attribute, it is only realized in directions in 33 (5%) cases (compared to 65% and 47% for "left" and "right" respectively). While our EM implementation maximizes the likelihood for all alignment probabilities based on expected counts, many pairs are not – or not frequently – found in the corpus. This results in the model often choosing incorrect alignments for categorial attributes and makes up for 23% of the misaligned attributes in total.

We found that further 5% of the attributes are misaligned with frame elements containing pronouns that actually refer to a different attribute. As our word choice model does not account for the use of anaphora, none of the affected frame elements are aligned correctly. Given the genre of our corpus, integrating simple heuristics to resolve anaphora (e.g., binding to the closest preceding mention) could solve this problem for the majority of the cases.

6 Conclusion

We presented a weakly supervised method for aligning route representations and natural language directions on the basis of parallel corpora using EM-based learning. Our models adopt ideas from Liang et al. (2009) with special adaptations to the current application scenario. As a major difference to their work, we make use of frame-semantic annotations on the NL side as a basis for segmentation.

While we can show that the extended models significantly outperform a simple EM-based model, the overall results are still moderate. We cannot draw a direct comparison to the results presented in Liang et al. (2009) due to the different scenarios and data sets. However, the corpus they used for the NFL recaps scenario is the closest to ours in terms of available data size and percentage of aligned records (in our case attributes). For this kind of corpus, they achieve an F_1 score of 39.9% with the model that is closest to ours (Model 2'). Their model achieves higher performance for scenarios with more available data and a higher percentage of alignments. Thus we expect that our model benefits from additional data sets, which we plan to gather in web-based settings.

Still, we do not expect to achieve near to perfect alignments due to speaker variation, a factor we also observe in the current data. As our ultimate goal is to generate NL instructions from given route representations, we can nevertheless make use of imperfectly aligned data for the compilation of high-confidence rules to compute semantic input structures for NLG. Following previous work by Barzilay and Lee (2002), we can also exploit the fact that our data consists of multiple directions for each route to identify alternative realization patterns for the same route segments. In addition, (semi-)supervised models could be used to assess the gain we may achieve in comparison to the minimally supervised setting.

However, we still see potential for improving our current models by integrating refinements based on the observations outlined above: Missing alignment targets on the linguistic side – especially due to anaphora, elliptical or aggregating constructions – constitute the main error source. We aim to capture these phenomena within the linguistic markup in order to provide hidden alignment targets. Also, our current model only considers frame elements as alignment targets. This can be extended to include their verbal predicates.

Acknowledgements: This work is supported by the DFG-financed innovation fund FRONTIER as part of the Excellence Initiative at Heidelberg University (ZUK 49/1). We thank Michael Bauer and Pascal Neis for the specification of the route representation format and Carina Silberer and Jonathan Geiger for annotation.

References

Anderson, Anne H., Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, Catherine Sotillo, Henry Thompson, and Regina Weinert. 1991. The HCRC Map Task corpus. *Language and Speech*, 34(4):351–366.

Barzilay, Regina and Mirella Lapata. 2005. Collective content selection for concept-to-text-generation. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing,* Vancouver, B.C., Canada, 6–8 October 2005, pages 331–338.

Barzilay, Regina and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing,* Philadelphia, Penn., 6–7 July 2002, pages 164–171.

Brown, Peter F., Vincent J. Della Pietra, Stephan A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

Burchardt, Aljoscha, Katrin Erk, Anette Frank, Andrea Kowalski, and Sebastian Pado. 2006. SALTO: A versatile multi-level annotation tool. In *Proceedings of the 5th International Conference on Language Resources and Evaluation,* Genoa, Italy, 22–28 May 2006, pages 517–520.

Dale, Robert, Sabine Geldof, and Jean-Philippe Prost. 2005. Using natural language generation in automatic route description. *Journal of Research and Practice in Information Technology*, 37(1):89–106.

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistics Society, Series B (Methodological)*, 39(1):1–38.

Duboue, Pablo A. and Kathleen R. McKeown. 2003. Statistical acquisition of content selection rules. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing,* Sapporo, Japan, 11–12 July 2003, pages 121–128.

Fillmore, Charles J., Christopher R. Johnson, and Miriam R.L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250.

Hansen, Stefan, Kai-Florian Richter, and Alexander Klippel. 2006. Landmarks in OpenLS: A data structure for cognitive ergonomic route directions. In *Proceedings of the 4th International Conference on Geographic Information Science,* Münster, Germany, 20-23 September 2006.

Liang, Percy, Michael Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *Proceedings of ACL-IJCNLP 2009*, pages 91–99, August.

Mani, Inderjeet. 2001. *Automatic Summarization*. John Benjamins, Amsterdam, Philadelphia.

Maxwell, John T. and Ronald M. Kaplan. 1993. The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4):571–590.

Neis, Pascal and Alexander Zipf. 2008. Extending the OGC OpenLS route service to 3D for an interoperable realisation of 3D focus maps with landmarks. *Journal of Location Based Services*, 2(2):153–174.

Reiter, Ehud and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge, U.K.: Cambridge University Press.

Roth, Michael and Anette Frank. 2009. A NLG-based Application for Walking Directions. In *Companion Volume to the Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing,* Singapore, 2–7 August 2009, pages 37–40.

Schmid, Helmut. 1997. Probabilistic Part-of-Speech tagging using decision trees. In Jones, Daniel and Harold Somers, editors, *New Methods in Language Processing*, pages 154–164. London, U.K.: UCL Press.

Schuldes, Stephanie, Michael Roth, Anette Frank, and Michael Strube. 2009. Creating an annotated corpus for generating walking directions. In *Proceedings of the ACL-IJCNLP 2009 Workshop on Language Generation and Summarisation,* Singapore, 6 August 2009, pages 72–76.

Tiedemann, Jörg. 2003. Combining Clues for Word Alignment. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 339–346, Budapest, Hungary.

Vogel, Adam and Dan Jurafsky. 2010. Learning to Follow Navigational Directions. In *Proceedings of ACL-2010*, Uppsala, Sweden.

Vogel, Stephan, Hermann Ney, and Christoph Tillmann. 1996. HMM-based Word Alignment in Statistical Translation. In *Proceedings of the 16h International Conference on Computational Linguistics (COLING)*, pages 836–841, Copenhagen, Denmark.

A Multiple-Domain Ontology Builder

Sara Salem **Samir AbdelRahman**
Computer Science Department
Faculty of Computers and Information - Cairo University
{s.salem,s.abdelrahman@fci-cu.edu.eg}

Abstract

The interpretation of a multiple-domain text corpus as a single ontology leads to misconceptions. This is because some concepts may be syntactically equal; though, they are semantically lopsided in different domains. Also, the occurrences of a domain concept in a large multiple-domain corpus may not gauge correctly the concept significance. This paper tackles the mentioned problems and proposes a novel ontology builder to extract separate domain specific ontologies from such a corpus**.** The builder contribution is to sustain each domain specific concepts and relations to get precise answers for user questions. We extend a single ontology builder named Text2Onto to apply our thought. We fruitfully enhance it to answer, more precisely, questions on a subset of AQUAINT corpus.

1 Introduction

Domain ontology is a knowledge representation of the domain as a set of concepts and relations. Ontology notion always presents handy semantic solutions for various hot research areas such as Semantic Web, Informational Retrieval, and Question Answering.

Currently, automatic ontology builders presume that the given corpus has a single domain. When used with a multiple-domain corpus, these builders generate 1 large ontology for the whole corpus. Dramatically, this causes 2 domain misconception problems. First, the ontology conceptual model becomes imprecise for the common concepts in various domains having different semantics. Second, the relevance weights assigned to the concepts do not measure precisely their significance in specific domains.

This paper presents a promising solution for the mentioned problems. The proposed solution is an integrated 2-layer ontology builder. The ontology layers are: 1) the conceptual layer, which has the key concepts and relations of each separate domain, and 2) the general layer, which maintains the general domain information regarding related persons, organizations, locations, and dates. Our proposed 2-layer ontology improves the extracted answers for single-domain and cross-domain questions. We successfully prove our thought against Text2Onto builder.

Ontology extraction from a domain corpus has been targeted by many researchers. The core extraction approaches can be classified into 3 approaches. The first approach is to build the ontology from scratch (Buitelaar et al., 2004; Cimiano and Völker, 2005). The second approach is to extend a predefined general ontology, such as WordNet, with possible application domain concepts and relations (Navigli and Velardi, 2004). The last approach is to build ontology as a composition of other predefined ontologies (Cimiano et al., 2006). Moreover, as an ontology building design decision, the resultant ontology is either a single layer ontology or a multi-layered ontology (Benslimane et al., 2000; Dumontier and Villanueva-Rosales, 2007).

The paper is organized as follows: Section 2 introduces some related systems; Section 3 explains the misconceptions due to extracting a single ontology from a multiple-domain corpus; Section 4 describes our proposed builder; Section 5 illustrates our Question Answering system, which is used for the evaluation; Section 6 states our evaluation results; and Section 7 draws our conclusion and directions for the future work.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 967–975,
Beijing, August 2010

2 Related Work

There are 3 main approaches for ontology building, namely building from scratch, extending a general ontology, or building an ontology as a composition of other predefined ontologies.

Text2Onto (Cimiano and Völker, 2005) applies the first approach. It is a framework for learning ontologies automatically from textual data. It implements diverse linguistic and statistical techniques to extract domain concepts and relations. It combines results from different techniques, and it represents the extracted ontology elements in a so called Probabilistic Ontology Model (POM), which assigns a confidence value for each learnt element.

OntoLT (Buitelaar et al., 2004) is another example of building from scratch. It is a Protégé[1] plug-in that extracts ontology from text by defining a set of mapping rules. The rules map certain linguistic structures in an annotated text into ontological elements. The extracted elements are validated by the user before being inserted into the ontology.

OntoLearn (Navigli and Velardi, 2004) employs the second approach. It is a framework for trimming and extending general purpose ontologies, like WordNet, with specific domain terminologies and taxonomies. It extracts domain terminologies, and it uses a relevance measure to keep out non-relevant terms. OntoLearn uses a novel technique, called SSI, to assign a domain specific term to the correct sense in a general ontology.

The third approach is proposed in (Cimiano et al., 2006). It presents a system that integrates several heterogeneous semantic sources into 1 ontology, which is used to extract answers for user queries from various knowledge sources.

As a design decision, the ontology may consist of a single layer or of multiple layers. Benslimane et al. (2000) apply the multiple-layer approach for manually generating a set of interrelated ontology layers; each layer models a spatial domain specific function. Also, Dumontier and Villanueva-Rosales (2007) suggest a 3-layer ontology design. The first layer (primitive layer) defines the basic domain concepts and relations. The second layer (complex layer) imposes more complex domain restrictions on the primitive layer. The top layer (application layer) maintains application specific restrictions.

Our builder constructs a layered ontology from scratch. Its main distinguished features are: 1) generating separate domain specific ontologies from a multiple-domain corpus, 2) extracting general domain information, in addition to core domain conceptual information, and 3) it is an automatic multi-layered ontology builder, unlike other automatic builders, which generate single layer ontologies.

Our system can extend current builders, which extract ontologies from textual data, allowing them to handle a multiple-domain corpus. We selected Text2Onto because it is an automatic ontology builder, and it implements a variety of algorithms to extract many types of ontology elements. We use a news corpus as a multiple-domain corpus since it contains documents from different domains like Politics, Sports, Arts, and Finance.

3 Ontology Misconceptions

Building a single ontology for a given corpus is a familiar method. However, when dealing with a multiple-domain corpus, the builder usually suffers from the following 2 problems:

First, the ontology conceptual model becomes imprecise in the definition of common concepts that are semantically lopsided in different domains. For example, the concept "wall street" in the Finance domain is defined as a financial institution, and it is in the Arts domain defined as a movie. It is inaccurate to define the concept with 2 totally different meanings in 1 ontology. It is also incorrect to ignore a definition of them. When using Text2Onto for that example, it generates only 1 definition for "wall street" as a subclass-of "institution".

Second, when weighing concepts in a multiple-domain corpus, the relevance weights assigned to the concepts do not indicate the significance of each concept in a certain domain. As a result, some core domain specific concepts may have low weights with respect to the whole corpus. For example the concept "trading" has a low weight in a multiple-domain corpus; although, it is a main concept in the Finance domain (Section 6.2). This gives wrong indication of the concept importance to the user.

[1] http://protege.stanford.edu/

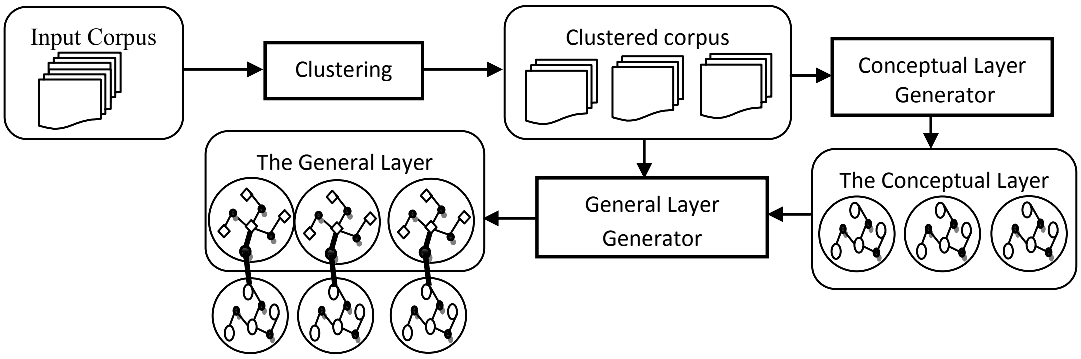


Figure 1. The Multiple-Domain Ontology Builder.

4 The Proposed Ontology Builder

Our builder aims to extract precise ontologies, which model possible knowledge in a multiple-domain corpus. A domain corpus, mostly, not only contains information about the core domain concepts and their relations, but it also contains general domain information such as dates of events and names of persons, locations, or organizations participating in the domain. Existing ontology builders either ignore this general information or they provide a limited implementation to extract it.

4.1 System Overview

The input to our builder (Figure 1) is a multiple-domain corpus. The first step is the ***clustering operation***, which divides the given corpus documents into clusters that are different among each other with high internal similarity. The next step is the ***conceptual layer generation***. In this step, we use Text2Onto to extract a separate ontology for each domain. Finally, the ***general layer generator*** uses each domain corpus and the conceptual layer ontology to extract relations among the concepts and the Named Entities in that domain.

4.2 The Conceptual Layer

The first step in constructing the conceptual layer is the clustering operation. We separate a multiple-domain corpus into various domain specific corpora such that the domain concepts are weighted based on their significance in that domain; also, the common concepts in different domains are separated. We favored a hierarchical clustering technique over a flat clustering one. That was because the number of resulting clusters should be known as a parameter in the latter. However, the number of corpus domains might be unknown in our case.

We employ the agglomerative hierarchical clustering technique (Manning et al., 2008). The technique starts with each document as a singleton cluster, and then it successively merges pairs of similar clusters until all clusters are merged into 1 cluster. We use the vector space model (Manning et al., 2008) to represent each document as a vector of terms' weights. The weight of a term *w* in a document *d* is calculated using the TF-IDF measure (Equation 1).

$$TFIDF(w,d) = TF(w,d) * log\frac{N}{DF(w)} \qquad (1)$$

Where *N* is the corpus size, *TF (w,d)* is the number of occurrences of the term *w* in the document *d*, and *DF (w)* is the number of documents containing the term *w*.

The similarity between 2 documents is calculated using the Cosine Similarity measure (Equation 2).

$$cosine(d1,d2) = \frac{V(d1) \, . \, V(d2)}{||V(d1)|| * ||V(d2)||} \qquad (2)$$

Where *V(d)* is the terms' weights vector for the document *d*, ||*V(d)*|| is the Euclidean length of the vector *V(d)*, and the numerator is the dot product of the 2 vectors.

The similarity between 2 clusters is calculated using the UPGMA measure (Steinbach et al., 2000) (Equation 3).

$$sim(C1, C2) = \frac{\sum_{\substack{d1 \in C1 \\ d2 \in C2}} cosine(d1, d2)}{size(C1) * size(C2)} \qquad (3)$$

We use the UPGMA measure to cluster a subset of DMOZ[2] data (1322 documents, in 4 domains), and it performs F-Measure of 0.86. Steinbach et al. (2000) describe how to calculate F-Measure for a hierarchy of clusters.

The combination similarity is the similarity of 2 merged clusters. We use this measure to cut the clusters hierarchy into M clusters by grouping ones having a minimum combination similarity of the threshold value €[3]. After clustering, we use Text2Onto to generate an ontology for each cluster (domain).

4.3 The General Layer

Text2Onto performs well in extracting ontology elements such as concepts, sub-class-of relations, instance-of relations, and part-of relations. Unfortunately, it performs inaccurately in extracting general domain information such as Named Entities and numeric information. There are 3 reasons for such misconception. First, proper nouns are not extracted as concepts. Second, numeric data is ignored. Third, restricted patterns are applied for the relations of Named Entities, that include only verb relations like [(NP |PNP) *verb* (NP|PNP)] and instance-of relations like [NP *such as* PNP], [*such* NP *as* PNP], and [PNP (NP)].

Because of the above reasons, we propose a highly flexible pattern based relation extractor. In our system, a pattern is a sequence of tags in the form of a regular expression. The possible tags are the normal POS tags like NN, VB, JJ, IN besides the following 5 tags CONCEPT, PERSON, LOCATION, ORGANIZATION, and DATE. This criterion is called Mixed Tagging. Currently, dates are the only data containing numbers extracted by our builder, but we can easily extend it to handle more numeric data.

The Mixed Tagging operation inputs are a document and the related conceptual ontology (Figure 2). The operation output is a mixed tagged document. The tagged text is then provided to the Relations Extractor to take out all relations matching our current predefined patterns. Example patterns are listed in Table 1; the first 2 patterns are verb relations, and the last 2 are noun relations.

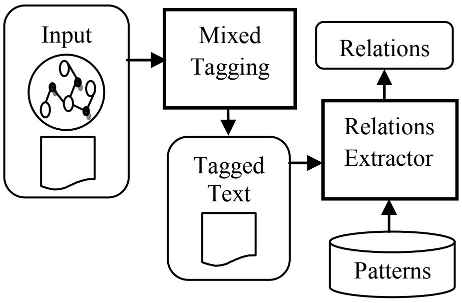


Figure 2. The General Relations Extraction.

The regular expression ([.{1,12}]){0,5} is used to limit the maximum number of tokens between the subject, the object, and the relation to 5 tokens. The expression [NN.?.?] matches any noun tag, and [VB.?] matches any verb tag.

After extracting the relations in all domain documents, the domain general ontology is created. It imports the corresponding conceptual ontology to model the relations among Named Entities and concepts.

([PERSON]) ([.{1,12}]){0,5}**([VB.?])+** ([.{1,12}]){0,5}**([CONCEPT])**
([ORGANIZATION])([.{1,12}]){0,5} **([DATE])**([.{1,12}]){0,5}**([VB.?])+**
([PERSON])([.{1,12}]){0,5} **([NN.?.?])+**([.{1,12}]){0,5}**([DATE])**
([NN.?.?])+([.{1,12}]){0,5}**([PERSON])** ([.{1,12}]){0,5}**([ORGANIZATION])**

Table 1. Sample Relation Patterns.

5 Question Answering System

Based on (Brank et al., 2005), a generated ontology can be evaluated using 4 different ways: 1) by a human who assesses it based on specific criteria, 2) by a comparison with the source data, 3) by a comparison with a golden standard, or 4) by using the ontology in an application and measuring the application performance. We chose the last option because the manual human assessment and the comparison with the source data are time consuming. Also, there is no golden standard ontology for a multiple-domain corpus.

Recently, researchers have studied the use of ontologies to extract answers to the user questions. AquaLog (Lopez et al., 2007) and

[2] http://www.dmoz.org/

[3] For clustering 600 AQUAINT documents, we use €=0.55 resulting in 7 Clusters (Secion 6.4).

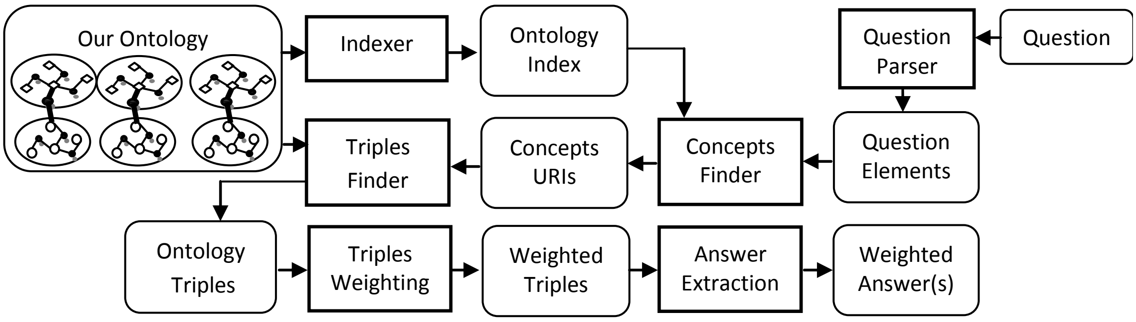


Figure 3. The Question Answering System.

PowerAqua (Lopez et al., 2009) are both ontology based Question Answering systems. PowerAqua extracts answers from various ontologies available on the web, unlike AquaLog, which extracts answers from 1 configurable ontology.

5.1 System Description

We implemented our simple Question Answering system handling who, when, where, and what questions. In the following, we describe the components of the system (Figure 3).

The Indexer: to make it easier for the system to locate the question concepts, an index is generated for our layered ontology. All concepts in different ontologies containing a certain stem are grouped in an index entry in the index file. The form of an index entry is as follows:

Stem,(Concept URI)+

The Question Parser: this component parses the user question, and it extracts 4 elements from it. First, the answer type; it can be PERSON, LOCATION, ORGANIZATION, DATE, or ANY based on the question type such as who, where, when, or what. Second, the answer restriction; it is used to limit the answers of *what* questions. For example, the answers for "what sport …?" question are restricted only to the sport types. Third, the question target; it defines the thing in which the question is interested. The fourth element is the relation; it contains the main verb(s) in the question. As an example, the elements of the question "*What sport does Jennifer Capriati play?*" are: the answer type (ANY), the restriction (sport), the question target (Jennifer Capriati), and the relation (play).

For a compound (2-clause) question such as "*What countries have Rhodes Scholars come from* ***and*** *has the Hale Bopp comet visible?*", each question clause is parsed as a separate question; finally, the answer extraction step intersects the answers of both clauses.

The Concepts Finder: using the ontology index, it locates concepts containing the stems of the question target and the restriction (if exists).

The Triples Finder: it extracts the triples which contain the question target concepts either as subjects or as objects. If the question is a definition question like "What is something?", the triple finder extracts only the sub-class-of triples.

The Triples Weighting: the triples are weighted based on their similarity to the question using our similarity criterion (Equation 4):

$$sim(Q,T) = \frac{\sum_{\substack{a\in Q \\ b\in T}} Lin(a,b)}{L(Q) * L(T)} \qquad (4)$$

Where Q and T are sets of the bag-of-words for the question relation and the triple relation respectively, *Lin(a,b)* is a measure for the semantic similarity between *a* and *b* based on WordNet (Lin, 1998), and *L(x)* is the number of elements in the set *x*.

The Answer Extraction: this component first filters out the triples mismatching the expected answer type. Then, if there is no restriction element, it extracts the answer from the weighted triples by considering the triple object if the question target is the subject, and vice versa. The extracted answer from a triple is assigned the same triple weight. If the question has a restriction element, the answer(s) will be limited to the sub concepts of the restriction element. A weight (Equation 5) is assigned to each sub concept *s* based on its similarity to the extracted triples as follows:

$$W(s) = \frac{\sum_{T \in R} sim(S,T)}{L(R)} \qquad (5)$$

Where *R* is the set of extracted triples, *S* and *T* are the sets of bag-of-words for the sub concept and the triple relation respectively, *sim(S,T)* is calculated using Equation 4, and *L(R) is* the number of elements in *R*.

For a compound question, the list of resulting answers contains only the common answers extracted for the 2 clauses.

6 Evaluation and Discussion

In our evaluation, we assess: 1) the enhancement of the concepts' weights in a specific domain corpus, 2) the enhancement of modeling common concepts in different domains with different semantics, and 3) the performance of our Question Answering system. The assessment is done through a comparison between our approach and Text2Onto.

In the development of our builder, we used Text2Onto[4], Stanford Part-Of-Speech Tagger (POS Tagger)[5], Stanford Named Entity Recognizer (NER)[6], and Jena[7]. In the Question Answering system, we also used the Java WordNet Similarity Library (JWSL)[8]; it implements the Lin measure.

6.1 Data Set

Our evaluation is based on the AQUAINT[9] corpus (Graff, 2002). It is an English news corpus containing documents from the New York Times News Service, the Xinhua News Service, and the Associated Press Worldstream News Service. The Question Answering track in TREC[10] (The Text REtrieval Conference) provides a set of questions on AQUAINT corpus along with their answers.

6.2 Concepts Weights Enhancement

For this experiment, we generated a corpus for the 3 domains, namely Finance, Sports, and Movies, from AQUAINT documents, such that each domain has equal number of documents. We measured the concepts' significance weights when using Text2Onto to generate a single ontology for the whole corpus and when using our builder to generate 3 different domains ontologies. We consider 3 measures implemented in Text2Onto, namely the Relative Term Frequency (RTF), the Entropy, and the TF-IDF.

The RTF for a concept *w* is the probability of the concept occurrence in the corpus (Equation 6).

$$p(w) = \frac{No.of\ occurences\ of\ w}{No.of\ all\ corpus\ concepts} \qquad (6)$$

The entropy and the normalized entropy for a concept *w* are calculated as follows (Equations 7 and 8 respectively):

$$Ent(w) = p(w) * \log p(w) \qquad (7)$$

$$N\big(Ent(w)\big) = \frac{Ent(w)}{Min\ Ent - Max\ Ent} \qquad (8)$$

In Section 4.2, we mention how to calculate the TF-IDF value for a term *w* in a document *d* (Equation 1). The TF-IDF weight and the normalized TF-IDF weight for a concept *w* in the whole corpus are calculated as follows (Equations 9 and 10 respectively):

$$TFIDF(w) = \frac{\sum_{d \in D} TFIDF(w,d)}{N} \qquad (9)$$

$$N\big(TFIDF(w)\big) = \frac{TFIDF(w)}{\sqrt{\sum_{ci \in C} TFIDF(ci)^2}} \qquad (10)$$

Where *D* is the set of documents containing *w*, *N* is the corpus size, and *C* is the set of all concepts in the corpus.

Since the concept weight is proportional to its occurrences in the corpus with respect to the other concepts, the fair distribution of the occurrences leads to precise weight calculation. In the specific domain corpus, the distribution is more reasonable than in multiple-domain corpus.

[4] http://code.google.com/p/text2onto/

[5] http://nlp.stanford.edu/software/tagger.shtml

[6] http://nlp.stanford.edu/software/CRF-NER.shtml

[7] http://jena.sourceforge.net/

[8] http://grid.deis.unical.it/similarity/

[9] http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002T31

[10] http://trec.nist.gov/data/qa.html

Domain	Concept	Entropy		TF-IDF		RTF	
		Text2 Onto	Our Builder	Text2 Onto	Our Builder	Text2 Onto	Our Builder
Finance	Stock	0.181	0.999	0.053	0.103	0.001	0.020
	Trading	0.155	0.670	0.044	0.139	0.001	0.010
	Shares	0.100	0.670	0.036	0.139	0.000	0.010
	Economy	0.100	0.670	0.026	0.051	0.000	0.010
Sports	Sport	0.822	0.974	0.344	0.379	0.012	0.019
	Baseball	0.321	0.389	0.147	0.190	0.003	0.006
	League	0.299	0.363	0.134	0.174	0.003	0.005
	Football	0.205	0.251	0.085	0.111	0.002	0.003
Movies	Actor	0.525	0.613	0.150	0.194	0.007	0.022
	Movie Industry	0.230	0.362	0.098	0.263	0.002	0.011
	Music	0.205	0.326	0.085	0.230	0.002	0.009
	Home Video	0.038	0.066	0.012	0.032	0.000	0.001

Table 2. Concepts Weights Comparison between Our Builder and Text2Onto.

This fact can be verified easily from Table 2. The 3 measures give higher weights in the domain specific ontologies than in a single ontology for the whole corpus.

6.3 Modeling Common Concepts

To study the enhancement in modeling common concepts having different meaning in different domains, we chose 5 concepts as samples (Table 3). For each concept, we selected documents from AQUAINT and from the Wikipedia concerning the concepts in 2 different domains.

In this experiment, the single ontology generated by Text2Onto contains only 1 definition for each concept namely wall_street *is_a* institution, marijuana *is_a* drug, bear *is_a* mammal, jaguar *is_a* cat, and world_war *is_a* war. On the other hand, our builder maintains both concept definitions in different ontologies.

Concept	Definition 1	Definition 2
Wall Street	A financial Institution	A movie
Marijuana	A drug	A song
The bear	A Mammal	A movie
Jaguar	A big cat	A car
World War	A war	A museum

Table 3. Sample of Lopsided Concepts.

6.4 Question Answering Enhancement

The experiment includes common concepts definition questions, single-domain questions, and cross-domain questions.

To illustrate the effect of the common concepts misconception problem solved by our builder against Text2Onto, we generated 5 definition questions for the 5 concepts in Table 3, like "what is wall street?", "what is marijuana?"…etc.

For the single-domain questions, we used a subset of AQUAINT corpus composed of 600 documents clustered into 7 domains using combination similarity threshold value of 0.55. We selected 60 factoid questions from TREC 2004 questions having their answers in these documents. Examples of single-domain questions are:

- *Who discovered prions?*
- *When was the IFC established?*

In addition to factoid questions, TREC 2004 also includes list questions. The answers of each question are aggregated from multiple documents. We used these questions in generating 10 cross-domain questions. Each question combines 2 of TREC list questions such that the 2 list questions are in different domains. Examples of these questions are:

- *What cities have an Amtrak terminal and have Crip gangs?*
- *What countries are Burger King located in and have IFC financed projects?*

Evaluation Criteria: the accuracy (A) (Equation 11) is used for evaluating single-domain questions because each factoid question has only 1 correct answer.

$$A = \frac{No.of\ correct\ answers}{No.of\ questions} \qquad (11)$$

The definition and cross-domain questions have multiple correct answers. The average Precision (P), Recall (R), and F-Measure (F) (Equations 12, 13, and 14 respectively) of all questions are used for our evaluation.

$$P = \frac{No.of\ correct\ answers}{No.of\ retrieved\ answers} \qquad (12)$$

$$R = \frac{No.of\ correct\ answers}{No.of\ actual\ answers} \qquad (13)$$

$$F = \frac{2 * P * R}{P + R} \qquad (14)$$

Table 4 shows that, in the definition questions, we achieve F-Measure of 1, while Text2Onto achieves 0.5. This is because our builder maintains the 2 different definitions of each concept, unlike Text2Onto, which contains only one.

Questions Type	Our Ontology	Text2Onto Ontology

Definition Questions	P=1.0
	R=1.0
	F=1.0
		P=0.5
		R=0.5
		F=0.5

Single-Domain	A=68%	A=0.05%

Cross-Domain	P=0.49
	R=0.59
	F=0.44
		P=0
		R=0
		F=0

Table 4. Question Answering Evaluation.

In the single-domain questions, using our ontology, we could answer 41 questions while using Text2Onto ontology we could answer only 3 questions ("what particle is a quark?", "what are prions made of?", and "What is the treatment of cataract?"). The low coverage of Named Entities in Text2Onto hinders it from answering correctly any question of types Who, When, and Where. This indicates the enhancement introduced by the proposed general layer for modeling accurately more domain information. In the cross-domain questions, we achieve F-Measure of 0.44. None of the cross-domain questions are answered using Text2Onto ontology due to the mentioned Named Entity coverage problem.

Although our results are better than Text2Onto, there is a room for more improvements. There are 4 main sources for retrieving wrong or incomplete answers (Table 5). Some relations are not extracted because their elements (subject, relation, and object) are not near enough from each other in the text, so none of our patterns or Text2Onto patterns could match them. This is the source of 65% of the errors. Missed Named Entities or wrongly tagged ones cause 16% of the errors. Some relations are not extracted because co-reference has not been handled yet. That leads to 12% of the total errors. Finally, in the factoid questions, we consider the answer with the highest weight to be the correct answer; 7% of the answers are extracted but with lower weights.

Error Type	Error percentage
No matching pattern	65%
NER Error	16%
Co-Reference	12%
Low answer weight	7%

Table 5. Answer Error Sources.

Based on the mentioned experiments, our builder outperforms Text2Onto in Question Answering. In addition, it can be used skillfully to enhance other Natural Language Processing applications such as Information Retrieval from multiple-domain data. Our initial results using 220 queries on 600 AQUAINT documents records 0.35 F-Measure against Lucene[11], which achieves 0.18.

7 Conclusion and Future Work

This paper presents the misconception problems when interpreting a multiple-domain corpus in a single ontology. A novel ontology builder is presented handling these problems by generating separate domain ontologies describing core and general domain information.

Currently, we hand on improving our builder relation extractor to answer more TREC questions by automatically learning patterns from text and by handling co-reference. Moreover, we are working to enhance the performance of our Information Retrieval system.

[11] http://lucene.apache.org/

References

Benslimane, D., E. Leclercq, M. Savonnet, M.-N. Terrasse, and K. Yétongnon. 2000. *On the Definition of Generic Multi-layered Ontologies for Urban Applications.* In the International Journal of Computers, Environment, and Urban Systems, volume 24: 191-214.

Brank, Janez, Marko Grobelnik, and Dunja Mladenić. 2005. *A Survey of Ontology Evaluation Techniques.* In the Proceedings of the 8th International Multiconference on Information Society: 166-169.

Buitelaar, Paul, Daniel Olejnik, and Michael Sintek. 2004. *A Protégé Plug-In for Ontology Extraction from Text Based on Linguistic Analysis.* In the Proceedings of the 1st European Semantic Web Symposium: 31-44.

Cimiano, Philipp, and Johanna Völker. 2005. *Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery*. In the Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems: 227-238.

Cimiano, Philipp, Peter Haase, York Sure, Johanna Völker, and Yimin Wang. 2006. *Question Answering on Top of the BT Digital Library*. In the Proceedings of the 15th International Conference on World Wide Web: 861-862.

Dumontier, Michel, and Natalia Villanueva-Rosales. 2007. *Three-Layer OWL Ontology Design.* In the Proceedings of the *2nd* International Workshop on Modular Ontologies. CEUR Workshop Proceedings, volume 315.

Graff, David. 2002. *The AQUAINT Corpus of English News Text*. Linguistic Data Consortium, Philadelphia.

Lin, Dekang. 1998. *An Information-Theoretic Definition of Similarity*. In the Proceedings of the 15th International Conference on Machine Learning: 296-304.

Lopez, Vanessa, Victoria Uren, Enrico Motta, and Michele Pasin. 2007. *AquaLog: An Ontology-driven Question Answering System for Organizational Semantic Intranets*. In the Journal of Web Semantics, volume 5: 72-105.

Lopez, Vanessa, Victoria Uren, Marta Sabou, and Enrico Motta. 2009. *Cross Ontology Query Answering on the Semantic Web: An Initial Evaluation*. In the Proceedings of the 5th International Conference on Knowledge Capture: 17-24.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Online edition. http://nlp.stanford.edu/IR-book/information-retrieval-book.html. Cambridge University Press.

Navigli, Roberto, and Paola Velardi. 2004. *Learning Domain Ontologies from Document Warehouses and Dedicated Web Sites.* In the Journal of Computational Linguistics, volume 30: 151-179.

Steinbach, Michael, George Karypis, and Vipin Kumar. 2000. *A Comparison of Document Clustering Techniques.* Technical Report #00-034, University of Minnesota.

Weakly Supervised Morphology Learning for Agglutinating Languages Using Small Training Sets

Ksenia Shalonova
Computer Science,
University of Bristol
ksenia@cs.bris.ac.uk

Bruno Golénia
Computer Science,
University of Bristol
csbsgg@bristol.ac.uk

Abstract

The paper describes a weakly supervised approach for decomposing words into all morphemes: stems, prefixes and suffixes, using wordforms with marked stems as training data. As we concentrate on under-resourced languages, the amount of training data is limited and we need some amount of supervision in the form of a small number of wordforms with marked stems. In the first stage we introduce a new Supervised Stem Extraction algorithm (SSE). Once stems have been extracted, an improved unsupervised segmentation algorithm GBUMS (Graph-Based Unsupervised Morpheme Segmentation) is used to segment suffix or prefix sequences into individual suffixes and prefixes. The approach, experimentally validated on Turkish and isiZulu languages, gives high performance on test data and is comparable to a fully supervised method.

1 Introduction

The major function of morphological analysis is decomposition of words into their constituents - stems and prefixes/suffixes. In recent years Machine Learning approaches were used for word decomposition. There is a number of both unsupervised morphology learning systems that use ”raw” wordforms as training data (Creutz and Lagus, 2002; Goldsmith, 2001; Kazakov and Manandhar, 2001) and supervised morphology learning systems using segmented wordforms into stems and affixes as training data (Oflazer et al., 2001). The supervised morphology learning systems are usually based on two-level morphology (Koskenniemmi, 1983). There is also a weakly supervised approach that uses, for example, wordpairs as input, and this was applied mainly to fusional languages for stem extraction (Erjavec and Dzeroski, 2004). Our project concerns developing speech technology for under-resourced languages. For this type of applications we need a relatively fast, cheap (i.e. does not require large training sets), almost knowledge-free approach that gives high performance. We have chosen to use wordforms with marked stems as training data in order to fulfill the criteria mentioned above.

Morphological analysis is used in many practical Natural Language Processing applications such as Machine Translation, Text Mining, spellcheckers etc. Our near-term goal is the integration of the morphology learning algorithms into the language-independent Text-to-Speech (TTS) system for improvement of grapheme-to-phoneme rules, stress prediction and tone assignment. In particular, the morphology learning algorithms described in this paper will be incorporated into the available isiZulu TTS system for automatic prediction of lexical tones. In the isiZulu language lexical tone assignment depends on the morpheme boundary. The current isiZulu TTS system is tone-deaf due to the lack of morphological decomposition. A number of perception tests will be carried out in order to evaluate which performance of morphology decomposition is acceptable for TTS and will improve the quality of the synthesised speech. It seems that the unsupervised morphology learning systems can be relatively easy to implement from scratch, but their performance probably cannot be regarded as high enough to improve the performance of the synthesized speech. In order to overcome this problem we present a novel synthesis of supervised and unsupervised induction techniques for morphology learning.

Our approach consists of two parts: the new supervised stem extraction algorithm for agglutinat-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 976–983,
Beijing, August 2010

ing languages and the improved version of the unsupervised algorithm for segmenting affix sequences. In (Shalonova et al., 2009) the authors presented the function learning approach called TASR (Tree of Aligned Suffix Rules) for extracting stems in fusional languages given wordpairs (word in grammatical form - word in basic form). While this algorithm gives good performance for Russian and English, it gives quite poor performance for agglutinating languages as shown in Section 4. A new approach for stem extraction in agglutinating languages is required for two main reasons. Firstly, suffix (or prefix) sequences in agglutinating languages can be much longer than in fusional languages and TASR does not seem to be efficient on long affix sequences as it does not generalise data in the efficient way and generates too many specific rules. This leads to poor performance on unseen data. Secondly, in some agglutinating languages it could be easier for native speakers to provide a stem (i.e. to provide a list of wordforms with annotated stems), whereas in highly inflective fusional languages the stem is often strongly bound with suffix sequences, and providing a proper stem requires high linguistic expertise. TASR approach is more appropriate for word-and-paradigm or realizational morphology that focuses on the whole word form rather than on word segmentation. For example, in Russian the infinitive verb *govorit'* ('to speak') generates a set of grammatical forms or a paradigm - *govorivshij*, *govor'aschij*, *govorim* etc.

The second part of our approach is the improved version of GBUAS algorithm (Shalonova et al., 2009) that provides affix segmentation given unannotated affix sequences. Given stem boundaries in the training set, our method splits the input word into all morphemes: stems and prefixes/suffixes. Our two-stage approach is tested on the under-resourced language isiZulu containing both prefixes and suffixes, as well as on Turkish containing only suffixes. Turkish is the most commonly spoken of the Turkic languages (over 77 million people). isiZulu is the Bantu language with about 10 million speakers and it is the most widely spoken home language in South Africa. Both Turkish and isiZulu use agglutination to form new words from noun and verb stems. In comparison to Turkish, isiZulu is a tonal language. In contrast to East Asian languages, in isiZulu there are three steps for tone assignment: lexical, morphemic and terraced. For TTS the lexical and morphemic tones will need to be recovered from the lexicon and the grammar as the orthography has no tone marking. The terraced tone relation can in general be recovered and marked automatically from the tone sequence with a finite state model.

2 Stem Extraction Algorithm

The Stem Extraction Algorithm (SSE) is the supervised algorithm for stem extraction. The training data for the SSE represent wordforms with the marked stem boundaries. During the training stage we collect a set of all possible stem extraction rules from training data and assign precision measures to each rule. Each rule is of the form *L_R* where "_" is the stem boundary, L and R are the left and right graphemic contexts of a stem boundary of different lengths. We differentiate prefix *Lpref_Rstem* and suffix *Lstem_Rsuff* stem extraction rules that correspond to the rules containing the left-hand stem boundary and the right-hand stem boundary respectively. For example, the Turkish word *yer* ('earth') with the marked word boundary #ye_r# generates the following *Lstem_Rsuff* rules: #ye_r#, #ye_r, ye_r#, #ye_, ye_r, e_r#, _r#, ye_, e_r, _r, and e_, where the symbol '#' signifies the word initial and final positions. We are implementing similar feature vectors used for automatic pronunciation prediction based on the focal grapheme (in our case it is a stem boundary) and left/right graphemic contexts of different length (Davel and Barnard, 2008). The idea of implementing expanding context in NLP tasks is usually applied for two-level data like grapheme-to-phoneme mapping rules (Torkkola, 1993), whereas in our case we use it for one-level data.

The precision measure for each rule is calculated by the formula $p/(p+n+\varepsilon)$ where p and n are the number of positive and negative examples, and ε is used to cover the cases where there are no negative examples. A high precision is desirable and this occurs when there are high values of p and low values of n (i.e. many positive examples and few negative examples). Using negative examples in contrast to using only rule frequencies (or positive examples) improves the performance of the algorithm.

Definition 1. *The number of positive examples for the rule Lstem_Rsuff (or rule Lpref_Rstem) is the number of training instances of $Stem\_Suffixes$ (or $Prefixes\_Stem$) containing the substring L_R.*

Definition 2. *The number of negative examples for rule Lstem_Rsuff (or Lpref_Rstem) is the number of training instances $Stem\_Suffixes$ (or $Prefixes\_Stem$) such that $Stem + Suffixes$ (or $Prefixes + Stem$) contains substring L+R and $Stem\_Suffixes$ (or $Prefixes\_Stem$) does not contain substring L_R where '+' denotes string concatenation.*

In the above definitions '_' is a stem boundary.

Example 1. *Suppose we have only three isiZulu verbs: zi_bek_e, zi_nak_eke and a_hlul_eke. For the $Lstem\_Rsuff$ rule* ***'ek_e'****, the word zi_bek_e generates one positive example and the two other words zi_nak_eke and a_hlul_eke generate one negative example each.*

The approach given in Algorithm 1 aims to find the unique longest rule-pair '$Lpref\_Rstem$ and $Lstem\_Rsuff$' with the highest precision that is applied to the input wordform for stem extraction. In case the language does not have prefixes like Turkish, the longest rule $Lstem\_Rsuff$ with the highest precision is applied. The decision of using either a rule-pair or just a single suffix rule is influenced by prior knowledge that a particular language has got either both prefixes and suffixes like isiZulu or only suffixes like Turkish. From now on we will use the term 'rulepair' in application both to the rulepair '$Lpref\_Rstem$ and $Lstem\_Rsuff$' in case of isiZulu and to the rulepair ' and $Lstem\_Rsuff$' with an empty first element in case of Turkish.

Algorithm 1 Choosing rule pair for stem extraction.

```
input W = raw wordform; $P$ and $S$ are sets of unique $Lpref\_Rstem$ and $Lstem\_Rsuff$ rules
output $result\_rule\_pair$

$result\_rule\_pair \leftarrow \emptyset$
iMaxlength $\leftarrow \infty$
repeat
  $(p1,s1) \leftarrow$ getrulepair (P $\times$ S, W, iMaxlength)
  $(p2,s2) \leftarrow$ getrulepair (P $\times$ S $\setminus$ $(p1,s1)$, W, iMaxlength)
  iMaxlength $\leftarrow$ length$(p1,s1)$
until $(p1,s1) = \emptyset$ or precision $(p1,s1) \neq$ precision$(p2,s2)$ or length $(p1,s1) \neq$ length$(p2,s2)$
$result\_rule\_pair \leftarrow (p1,s1)$

function getrulepair(PS, W, iMaxlength)
ilength $\leftarrow 0$
r $\leftarrow \emptyset$
for all $(p,s) \in PS$ do
  if $(p,s)$ matches W then
    if length$(p,s) < iMaxlength$ and length$(p,s) >$ ilength then
      ilength $\leftarrow$ length$(p,s)$
      $r \leftarrow (p,s)$
    else
      if length$(p,s) =$ ilength and precision$(p,s) >$ precision$(r)$ then
        $r \leftarrow (p,s)$
      end if
    end if
  end if
end for
return $r$
end function
```

The search is carried out on the set of rule pairs matching an input raw wordform. The set is sorted by length first, and then by precision measure within each length category.

For example, if rulepairs have the following length-precision values:
'4-0.5,'4-0.5','4-0.2'

'3-0.4','3-0.3'
'2-0.3'
rulepair with the value 3-0.4 is selected.

The rulepair matches the input word if *Lpref_Rstem* and *Lstem_Rsuff* rules can be applied without contradicting each other. For example, the rule pair '#a_hl' and 'l_eke' matches the word *a_hlul_eke*, whereas the rule pair '#a_hlulek' and 'le_ke' does not match this word. For each input wordform the set of its own rulepair candidates is generated. The search in the algorithm among these rulepairs starts from the longest rulepairs, and this allows more specific rules and exceptions to be applied first, whereas the more general rules are applied if no specific rules cover the input wordform.

3 Graph-Based Unsupervised Morpheme Segmentation

In this section we extend GBUMS (Graph-Based Unsupervised Morpheme Segmentation) that segments sequences of prefixes and suffixes (Golénia et al., 2009). We propose an extension of GBUMS which uses the graph structure of GBUMS through a brute-force method. Our experiments showed the improved results on training set and allowed GBUMS to be run on the test sets for two languages: Turkish and isiZulu.

The algorithm GBUMS was originally developed in (Shalonova et al., 2009) under the name GBUSS (Graph-Based Unsupervised Suffix Segmentation) to extract suffix sequences efficiently and it was applied to Russian and Turkish languages on training sets. We refer to prefixes and suffixes generally as morphemes. GBUMS uses a morpheme graph in a bottom-up way. Similar to Harris (Harris, 1955), the algorithm is based on letter frequencies. However, when Harris uses successor and predecessor frequencies, they use position-independent *n*-gram statistics to merge single letters into morphemes until a stopping criterion is fulfilled.

In the morpheme graph, each node represents a morpheme and each directed edge the concatenation of two morphemes labelled with the frequencies in a M-corpus (see Figure 1). M-corpus is a list of morpheme sequences

Definition 3. *Let* $M = \{m_i | 1 \leq i \leq |M|\}$ *be a set of morphemes, let* f_i *be the frequency with which morpheme* m_i *occurs in a M-corpus of morpheme sequences, let* $v_i = (m_i, f_i)$ *for* $1 \leq i \leq n$*, and let* $f_{i,j}$ *denote the number of morpheme sequences in the corpus in which morpheme* m_i *is followed by morpheme* m_j*. The* morpheme graph $G = (V, E)$ *is a directed graph with vertices or nodes* $V = \{v_i | 1 \leq i \leq |V|\}$ *and edges* $E = \{(v_i, v_j) | f_{i,j} > 0\}$*. We treat* $f_{i,j}$ *as the label of the edge from* v_i *to* v_j*.*

In *G*, each node is initialised with a letter according to a M-corpus, then one by one, nodes are merged to create the real morphemes. To merge nodes, an evaluation function is required. In (Golénia et al., 2009), Golenia et al. employed the *Morph_Lift* evaluation function based on its relation to the *lift* of a rule for association rules in data mining (Brin et al., 1997).

Definition 4. *Morph_Lift is defined for a pair of morphemes* m_1 *and* m_2 *as follows:*

$$Morph\_Lift(m_1, m_2) = \frac{f_{1,2}}{f_1 + f_2} \quad (1)$$

From now on, we know how to merge nodes. Now, we need to figure out the most important part of GBUMS, which is the stopping criterion. The stopping criterion is to prevent overgeneralisation. In other words, the algorithm needs to be stopped before getting the initial M-corpus (since no merging is possible). This criterion is based on the Bayesian Information Criterion (BIC) and Jensen-Shannon divergence (Li, 2001).

BIC is used for selecting a model (set of morphemes) which fits a data set (M-Corpus) without being too complex. We want to point out that BIC is related to MDL. BIC is a trade-off between the maximum likelihood, the parameters of the model (probability and length of each morpheme) and the number of elements in the data set (frequency of each morpheme). A smaller value of BIC corresponds to a better model fit. The maximum of the Jensen-Shannon divergence is used in order to analyse the increase of log-likelihood among all possible models. The Jensen-Shannon divergence is defined as follows (Dagan et al., 1997):

Definition 5. *The* Jensen-Shannon divergence *is defined for two morphemes* m_1 *and* m_2 *as the de-*

crease in entropy between the concatenated and the individual morphemes:

$$D_{JS}(m_1,m_2) = H(m_1 \cdot m_2) - \frac{L_{m_1}H(m_1) + L_{m_2}H(m_2)}{N} \tag{2}$$

where $H(m) = -P(m)\log_2 P(m)$ $N = \sum_m Freq(m)$ *and* L_m *is the string length of* m*.*

Stopping criterion *requires that* $\Delta BIC < 0$ *which translates to:*

$$\max_{m_1,m_2} D_{JS}(m_1,m_2) \leq 2\log_2 N \tag{3}$$

Algorithm 2 The GBUMS morpheme segmentation algorithm

```
input M-Corpus = List of Strings
output M-CorpusSeg = List of Strings
  M-CorpusSeg $\leftarrow$ SegmentInLetters(M-Corpus);
  Graph $\leftarrow$ InitialiseGraph(M-CorpusSeg);
  repeat
    Max $\leftarrow$ 0;
    for all (p,q) $\in$ Graph do
      ML_Max $\leftarrow$ Morph_Lift(p, q);
      if ML_Max $>$ Max then
        Max $\leftarrow$ ML_Max;
        pMax $\leftarrow$ p;
        qMax $\leftarrow$ q;
      end if
    end for
    Graph $\leftarrow$ MergeNodes(Graph, pMax, qMax);
    M-CorpusSeg $\leftarrow$ DeleteBoundaries(M-CorpusSeg, Label(pMax), Label(qMax));
    Graph $\leftarrow$ AdjustGraph(M-corpusSeg, Graph);
  until StoppingCriterion(pMax, qMax, Max)
```

After several merging iterations, the output of the algorithm is the graph shown in Figure 1. The GBUMS is presented in Algorithm 2.

Note that the M-Corpus is completely segmented at the beginning of the algorithm. Then, the boundaries in the segmented M-Corpus are removed step by step according to a pair found in the graph with the maximum value for *Morph_Lift*. When the stopping criterion is fulfilled, the segmented M-Corpus represents the morpheme sequences.

At this point we present our extension of GBUMS based on a brute-force heuristic which scores every possible segmentation of an input morpheme sequence using graph values. We consider the morpheme graph as a model where each morpheme sequence can be extracted by the *MGraph* function (eq. 4).

Definition 6. *We define MGraph of a morpheme sequence without boundaries x as follows:*

$$MGraph(x) = \arg\max_{t \subseteq x} \frac{1}{N_t - C_t} \sum_{m \in t} L_m log(f_m + 1) \tag{4}$$

where

- *t is a morpheme sequence with boundaries of x,*
- *m is a morpheme of t,*
- *f_m is the frequency of the morpheme m,*
- *N_t is the number of morphemes existing in the graph,*
- *C_t is the number of morphemes existing and contiguous in the graph.*

Firstly, as a post-processing procedure the *MGraph* function improves the performance on training data. Secondly, it permits the identification of unseen morphemes. That is why the model generated by GBUMS can be run on test data sets.

Example 2. *Let our final morpheme graph be as shown in Figure 1 where nodes represent suffixes and their frequencies.*
Let x="ekwe" be our input suffix sequence that we want to segment into individual suffixes. We split this input sequence into all possible substrings from individual characters up to size of the input string length: e-k-w-e, e-k-we, e-kw-e, ek-w-e, ..., ekwe.
Using equation 4, we evaluate each substring and select the one with the highest score as the correct segmentation. Here, we have 7 potential segmentations with a score higher than 0 (MGraph > 0), e.g: e-k-w-e $= (\log(3)+\log(3))/2 = 1.0986$*, ek-w-e* $=(2\log(4)+\log(3))/2 = 1.9356$ *and ek-we* $=$

$2\log(4) = 2.7726$.
Consequently, ek-we is chosen as the correct segmentation for the substring "ekwe".

We would like to highlight that our new method can identify unseen cases with M-Graph, for instance, in the previous example suffix "we" was not present in the training graph, but was correctly extracted.

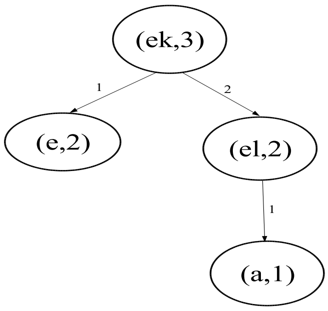


Figure 1: Example of a suffix subgraph in the training phase for isiZulu.

4 Results

Our experiments were based on Turkish data containing 1457 verbs and 2267 nouns, and isiZulu data containing 846 nouns and 931 verbs, with one single unambiguous segmentation per word.[1] Both isiZulu and Turkish data were uniquely sampled from the most frequent word lists.

Our first experiments compared TASR and the new SSE algorithm for stem extraction (10-fold cross validation assumes the following training and test set sizes: training sets containing 1311 wordforms for verbs and 2040 wordforms for nouns; test sets containing 146 wordforms for verbs and 227 wordforms for nouns). As can be seen from the Table 1, the performance of the SSE algorithm on Turkish data is much higher than that of TASR on the same dataset. As we mentioned in Section 1, TASR is not suitable for agglutinating languages with long suffix sequences. Although TASR algorithm gives an excellent performance on Russian, for most Turkish words it fails to extract proper stems.

	Test	FMea
TASR	Nouns	20.7±6.8
	Verbs	12.6±5.9
SSE	Nouns	84.3±3.2
	Verbs	82.1±3.7

Table 1: Comparison of TASR and SSE for Turkish using 10-fold cross validation.

Our next experiments evaluated the performance of GBUMS on its own given unsegmented suffix sequences from Turkish nouns and verbs as training data. The performance on these training data increased by approximately 3-4 % in comparison to the results presented in (Shalonova et al., 2009). We would like to point out that the results in (Shalonova et al., 2009) are based on training data rather than on test data, whereas in the current paper we run our algorithms on test (or unseen) data. Our final experiments examined performance on the test sets and were run both on Turkish and isiZulu data. We compared our approach with Morfessor run both in supervised and in unsupervised mode. Although Morfessor is known as one of the best unsupervised morphology learning systems, it is possible to run it in the supervised mode as well (Spiegler et al., 2008). The training data for SSE+ GBUMS contained wordforms with marked stems. During training stage the SSE algorithm was collecting information about stem boundaries and the GBUMS algorithm was run on unlabelled suffix and prefix sequences from the same training set. The test stage for the SSE+GBUMS approach was run on "raw" wordforms by applying the SSE algorithm first for stem extraction and then running GBUMS algorithm for segmenting prefix or suffix sequences after the SSE has extracted stems. Training data for supervised Morfessor used the same wordforms as for the SSE+GBUMS training set and contained wordforms segmented into stems and affixes (i.e. words segmentated into all morphemes were given as training data). The test data for supervised Morfesor were the same as those used for SSE+GBUMS. Morfessor in unsupervised mode was run on "raw" wordforms as training data. To evaluate our current work we ap-

[1] In agglutinating languages some wordforms even within one POS category can have several possible segmentations.

	Test	FMea
Supervised Morfessor	Nouns	74.6±2.3
	Verbs	84.5±2.2
SSE+ GBUMS	Nouns	78.8±2.4
	Verbs	76.9±0.7
Unsupervised Morfessor	Nouns	26.6±2.6
	Verbs	28.4±2.8

Table 2: Comparison of Morfessor and SSE+GBUMS for Turkish using 10-fold cross validation.

	Test	FMea
Supervised Morfessor	Nouns	76.7±1.6
	Verbs	88.5±2.4
SSE+ GBUMS	Nouns	87.9±1.9
	Verbs	84.5±2.5
Unsupervised Morfessor	Nouns	27.4±5.1
	Verbs	26.9±5.0

Table 3: Comparison of Morfessor and SSE+GBUMS for isiZulu using 10-fold cross validation.

plied the SSE+GBUMS approach for the under-resourced agglutinating language isiZulu containing both prefixes and suffixes and for Turkish containing only suffixes. The results show (Table 2 and Table 3) that our weakly supervised approach is comparable with the supervised Morfessor and decisively outperforms the unsupervised Morfessor. We think that it is useful to point out that unsupervised morphology learning systems in general require much larger training sets for better performance. F-measure is the harmonic mean of precision and recall, whereas precision is the proportion of true morpheme boundaries among the boundaries found, recall is the proportion of boundaries found among the true boundaries. In our experiments the GBUMS algorithm had no restrictions on affix length (Shalonova et al., 2009), but if there were restrictions, performance could be better. For isiZulu nouns our approach significantly outperformed supervised Morfessor, whereas for Turkish verbs SSE+GBUMS performed much worse. The best overall results obtained by GBUMS were based on the isiZulu nouns where about **53%** of all affixes were single letter affixes, whereas the worst results our approach gave for Turkish verbs where only about **12%** of affixes are composed of one letter. It is important to notice that the GBUMS algorithm, which is completely unsupervised, gives better results for extracting one letter affixes compared to Morfessor.

5 Conclusions

In the paper we described a weakly supervised approach for learning morphology in agglutinating languages. We were successful in our ultimate goal of synthesis of supervised and unsupervised induction techniques by achieving high performance on small amount of training data. Our weakly supervised approach is comparable with the supervised morphology learning system. As we are working with the languages for which linguistic resources are very limited (in particular words with morpheme boundaries), the developed method fulfills our goals of providing key components for speech and language products for such under-resourced languages. We speculate that the current performance might be improved by adding a small amount of completely ”raw” data to the training set.

The integration of our algorithms into working TTS systems is of key importance. As our near-term goal is the integration of morphology learning component into the currently working isiZulu TTS system, we will have to analyse the necessity of a Part of Speech Tagger (POS) and morphological disambiguation. In agglutinating languages some wordforms can be segmented in different ways (i.e. have different surface forms) and Machine Learning approaches normally select the most probable segmentation, and therefore our morphology disambiguation can be important. Morphological disambiguation for TTS can be considered a less complex problem than full morphological disambiguation as it can be linked, for example, to lexical tone disambiguation that may not require the full POS tag set. We intend to carry out user perception tests in order to evaluate the possible improvement in the isiZulu TTS quality after morphology information is added.

6 Acknowledgment

We would like to thank Kemal Oflazer from Sabanci University in Istanbul for his help and the Turkish data. We also thank Viktor Zimu and Etienne Barnard from CSIR in Pretoria for providing us isiZulu data. We also thank Roger Tucker for his support in the project. The work was sponsored by the EPSRC Grant EP/E010857/1 'Learning the morphology of complex synthetic languages'.

References

Brin, S., R. Motwani, J. Ullman, and S. Tsur. 1997. Dynamic itemset counting and implication rules for market basket data. In *ACM SIGMOD international conference on Management of data*, pages 255–264. ACM.

Creutz, M. and K. Lagus. 2002. Unsupervised discovery of morphemes. *Proceedings of the Workshop on Morphological and Phonological Learning of ACL-02*, pages 21–30.

Dagan, I., L. Lee, and F. Pereira. 1997. Similarity-Based Methods for Word Sense Disambiguation. *Thirty-Fifth Annual Meeting of the ACL and Eighth Conference of the EACL*, pages 56–63.

Davel, M. and E. Barnard. 2008. Pronunciation prediction with default refine. *Computer Speech and Language*, 22:374–393.

Erjavec, T. and S. Dzeroski. 2004. Machine learning of morphosyntactic structure: Lemmatising unknown Slovene words. *Applied Artificial Intelligence*, 18(1):17–40.

Goldsmith, J. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27:153–198.

Golénia, B., S. Spiegler, and P. Flach. 2009. UNGRADE: UNsupervised GRAph DEcomposition. In *Working Notes for the CLEF 2009 Workshop, CLEF 2009, Corfu, Greece.*

Harris, Z. 1955. From Phoneme to Morpheme. *Language*, 31(2):190–222.

Kazakov, D. and S. Manandhar. 2001. Unsupervised learning of word segmentation rules with genetic algorithms and inductive logic programming. *Machine Learning*, 43:121–162.

Koskenniemmi, K. 1983. *Two-level Morphology: A General Computational Model for Word Form Recognition and Production*. Ph.D. thesis, University of Helsinki.

Li, W. 2001. New stopping criteria for segmenting DNA sequences. *Physical Review Letters*, 86(25):5815–5818.

Oflazer, K., M. McShane, and S. Nirenburg. 2001. Bootstrapping morphological analyzers by combining human elicitation and machine learning. *Computational Linguistics*, 27(1):59–85.

Shalonova, K., B. Golenia, and P. Flach. 2009. Towards learning morphology for under-resourced languages. *IEEE Transactions on Audio, Speech and Language Procesing*, 17(5):956–965.

Spiegler, S., B. Golenia, K. Shalonova, P. Flach, and R. Tucker. 2008. Learning the morphology of Zulu with different degrees of supervision. *IEEE Spoken Language Technology Workshop*, pages 9–12.

Torkkola, K. 1993. An efficient way to learn English grapheme-to-phoneme rules automatically. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 199–202.

Multi-Document Summarization via the Minimum Dominating Set

Chao Shen and **Tao Li**
School of Computing and Information Sciences
Florida Internation University
{cshen001|taoli}@cs.fiu.edu

Abstract

Multi-document summarization has been an important problem in information retrieval. It aims to distill the most important information from a set of documents to generate a compressed summary. Given a sentence graph generated from a set of documents where vertices represent sentences and edges indicate that the corresponding vertices are similar, the extracted summary can be described using the idea of graph domination. In this paper, we propose a new principled and versatile framework for multi-document summarization using the minimum dominating set. We show that four well-known summarization tasks including generic, query-focused, update, and comparative summarization can be modeled as different variations derived from the proposed framework. Approximation algorithms for performing summarization are also proposed and empirical experiments are conducted to demonstrate the effectiveness of our proposed framework.

1 Introduction

As a fundamental and effective tool for document understanding and organization, multi-document summarization enables better information services by creating concise and informative reports for a large collection of documents. Specifically, in multi-document summarization, given a set of documents as input, the goal is to produce a condensation (i.e., a generated summary) of the content of the entire input set (Jurafsky and Martin, 2008). The generated summary can be generic where it simply gives the important information contained in the input documents without any particular information needs or query/topic-focused where it is produced in response to a user query or related to a topic or concern the development of an event (Jurafsky and Martin, 2008; Mani, 2001).

Recently, new summarization tasks such as update summarization (Dang and Owczarzak, 2008) and comparative summarization (Wang et al., 2009a) have also been proposed. Update summarization aims to generate short summaries of recent documents to capture new information different from earlier documents and comparative summarization aims to summarize the differences between comparable document groups.

In this paper, we propose a new principled and versatile framework for multi-document summarization using the *minimum dominating set*. Many known summarization tasks including generic, query-focused, update, and comparative summarization can be modeled as different variations derived from the proposed framework. The framework provides an elegant basis to establish the connections between various summarization tasks while highlighting their differences.

In our framework, a sentence graph is first generated from the input documents where vertices represent sentences and edges indicate that the corresponding vertices are similar. A natural method for describing the extracted summary is based on the idea of graph domination (Wu and Li, 2001). *A dominating set* of a graph is a subset of vertices such that every vertex in the graph is either in the subset or adjacent to a vertex in the subset; and

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 984–992,
Beijing, August 2010

a minimum dominating set is a dominating set with the minimum size. The minimum dominating set of the sentence graph can be naturally used to describe the summary: it is *representative* since each sentence is either in the minimum dominating set or connected to one sentence in the set; and it is with *minimal redundancy* since the set is of minimum size. Approximation algorithms are proposed for performing summarization and empirical experiments are conducted to demonstrate the effectiveness of our proposed framework. Though the dominating set problem has been widely used in wireless networks, this paper is the first work on using it for modeling sentence extraction in document summarization.

The rest of the paper is organized as follows. In Section 2, we review the related work about multi-document summarization and the dominating set. After introducing the minimum dominating set problem in graph theory in Section 3, we propose the minimum dominating set based framework for multi-document summarization and model the four summarization tasks including generic, query-focused, update, and comparative summarization in Section 4. Section 5 presents the experimental results and analysis, and finally Section 6 concludes the paper.

2 Related Work

Generic Summarization For generic summarization, a saliency score is usually assigned to each sentence and then the sentences are ranked according to the saliency score. The scores are usually computed based on a combination of statistical and linguistic features. MEAD (Radev et al., 2004) is an implementation of the centroid-based method where the sentence scores are computed based on sentence-level and inter-sentence features. SumBasic (Nenkova and Vanderwende, 2005) shows that the frequency of content words alone can also lead good summarization results. Graph-based methods (Erkan and Radev, 2004; Wan et al., 2007b) have also been proposed to rank sentences or passages based on the PageRank algorithm or its variants.

Query-Focused Summarization In query-focused summarization, the information of the given topic or query should be incorporated into summarizers, and sentences suiting the user's declared information need should be extracted. Many methods for generic summarization can be extended to incorporate the query information (Saggion et al., 2003; Wei et al., 2008). Wan et al. (Wan et al., 2007a) make full use of both the relationships among all the sentences in the documents and relationship between the given query and the sentences by manifold ranking. Probability models have also been proposed with different assumptions on the generation process of the documents and the queries (Daumé III and Marcu, 2006; Haghighi and Vanderwende, 2009; Tang et al., 2009).

Update Summarization and Comparative Summarization Update summarization was introduced in Document Understanding Conference (DUC) 2007 (Dang, 2007) and was a main task of the summarization track in Text Analysis Conference (TAC) 2008 (Dang and Owczarzak, 2008). It is required to summarize a set of documents under the assumption that the reader has already read and summarized the first set of documents as the main summary. To produce the update summary, some strategies are required to avoid redundant information which has already been covered by the main summary. One of the most frequently used methods for removing redundancy is Maximal Marginal Relevance(MMR) (Goldstein et al., 2000). Comparative document summarization is proposed by Wang et. al. (Wang et al., 2009a) to summarize the differences between comparable document groups. A sentence selection approach is proposed in (Wang et al., 2009a) to accurately discriminate the documents in different groups modeled by the conditional entropy.

The Dominating Set Many approximation algorithms have been developed for finding minimum dominating set for a given graph (Guha and Khuller, 1998; Thai et al., 2007). Kann (Kann, 1992) shows that the minimum dominating set problem is equivalent to set cover problem, which is a well-known NP-hard problem. Dominating set has been widely used for clustering in wireless networks (Chen and Liestman, 2002; Han and Jia, 2007). It has been used to find topic words for hierarchical summarization (Lawrie et al., 2001), where a set of topic words is extracted as a dominating set of word graph. In our work, we use the minimum dominating set to formalize the sentence extraction for document summarization.

3 The Minimum Dominating Set Problem

Given a graph $G =< V, E >$, *a dominating set* of G is a subset S of vertices with the following property: each vertex of G is either in the dominating set S, or is adjacent to some vertices in S.

Problem 3.1. *Given a graph G, the minimum dominating set problem (MDS) is to find a minimum size subset S of vertices, such that S forms a dominating set.*

MDS is closely related to the set cover problem (SC), a well-known NP-hard problem.

Problem 3.2. *Given F, a finite collection* $\{S_1, S_2, \dots, S_n\}$ *of finite sets, the set cover problem (SC) is to find the optimal solution*

$$F^* = \arg\min_{F' \subseteq F} |F'| \; s.t. \bigcup_{S' \in F'} S' = \bigcup_{S \in F} S.$$

Theorem 3.3. *There exists a pair of polynomial time reduction between MDS and SC.*

So, MDS is also NP-hard and it has been shown that there are no approximate solutions within $c \log |V|$, for some $c > 0$ (Feige, 1998; Raz and Safra, 1997).

3.1 An Approximation Algorithm

A greedy approximation algorithm for the SC problem is described in (Johnson, 1973). Basically, at each stage, the greedy algorithm chooses the set which contains the largest number of uncovered elements.

Based on Theorem 3.3, we can obtain a greedy approximation algorithm for MDS. Starting from an empty set, if the current subset of vertices is not the dominating set, a new vertex which has the most number of the adjacent vertices that are not adjacent to any vertex in the current set will be added.

Proposition 3.4. *The greedy algorithm approximates SC within* $1 + \ln s$ *where s is the size of the largest set.*

It was shown in (Johnson, 1973) that the approximation factor for the greedy algorithm is no more than $H(s)$, the s-th harmonic number:

$$H(s) = \sum_{k=1}^{s} \frac{1}{k} \leq \ln s + 1$$

Corollary 3.5. *MDS has a approximation algorithm within* $1 + \ln \Delta$ *where* Δ *is the maximum degree of the graph.*

Corollary 3.5 follows directly from Theorem 3.3 and Proposition 3.4.

4 The Summarization Framework

4.1 Sentence Graph Generation

To perform multi-document summarization via minimum dominating set, we need to first construct a sentence graph in which each node is a sentence in the document collection. In our work, we represent the sentences as vectors based on tf-isf, and then obtain the cosine similarity for each pair of sentences. If the similarity between a pair of sentences s_i and s_j is above a given threshold λ, then there is an edge between s_i and s_j.

For generic summarization, we use all sentences for building the sentence graph. For query-focused summarization, we only use the sentences containing at least one term in the query. In addition, when a query q is involved, we assign each node s_i a weight, $w(s_i) = d(s_i, q) = 1 - \cos(s_i, q)$, to indicate the distance between the sentence and the query q.

After building the sentence graph, we can formulate the summarization problem using

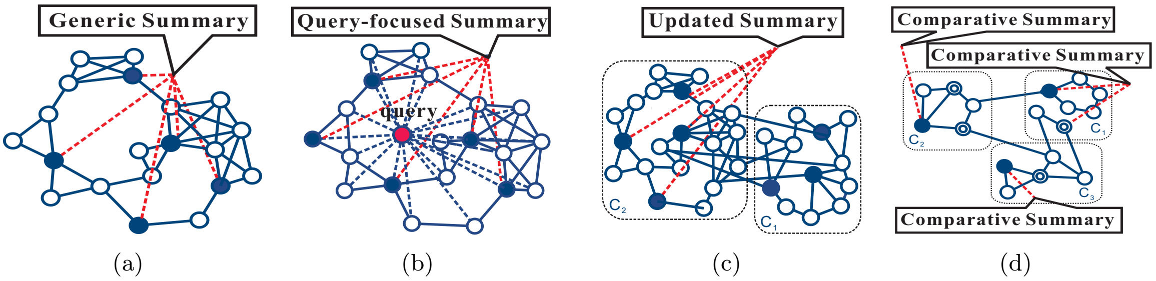


Figure 1: Graphical illustrations of multi-document summarization via the minimum dominating set. (a): The minimum dominating set is extracted as the generic summary. (b):The minimum weighted dominating set is extracted as the query-based summary. (c):Vertices in the right rectangle represent the first document set C_1, and ones in the left represent the second document set where update summary is generated. (d):Each rectangle represents a group of documents. The vertices with rings are the dominating set for each group, while the solid vertices are the complementary dominating set, which is extracted as comparative summaries.

the minimum dominating set. A graphical illustration of the proposed framework is shown in Figure 1.

4.2 Generic Summarization

Generic summarization is to extract the most representative sentences to capture the important content of the input documents. Without taking into account the length limitation of the summary, we can assume that the summary should represent all the sentences in the document set (i.e., every sentence in the document set should either be extracted or be similar with one extracted sentence). Meanwhile, a summary should also be as short as possible. Such summary of the input documents under the assumption is exactly the minimum dominating set of the sentence graph we constructed from the input documents in Section 4.1. Therefore the summarization problem can be formulated as the minimum dominating set problem.

However, usually there is a length restriction for generating the summary. Moreover, the MDS is NP-hard as shown in Section 3. Therefore, it is straightforward to use a greedy approximation algorithm to construct a subset of the dominating set as the final summary. In the greedy approach, at each stage, a sentence which is optimal according to the local criteria will be extracted. Algorithm 1 describes an approximation algorithm for generic summarization. In Algorithm 1, G is the sentence graph, $L(S)$ is the length of the summary, W is the maximal length of the summary, and $ADJ(v) = \{v'|(v', v) \in E(G)\}$ is the set of vertices which are adjacent to the vertex v. A graphical illustration of generic summarization using the minimum dominating set is shown in Figure 1(a).

Algorithm 1 Algorithm for Generic Summarization

```
INPUT: G, W
OUTPUT: S
$S = \emptyset$
$T = \emptyset$
while $L(S) < W$ and $V(G)! = S$ do
    for $v \in V(G) - S$ do
        $s(v) = |\{ADJ(v) - T\}|$
    $v^* = \arg\max_v s(v)$
    $S = S \cup \{v^*\}$
    $T = T \cup ADJ(v^*)$
```

4.3 Query-Focused Summarization

Letting G be the sentence graph constructed in Section 4.1 and q be the query, the query-focused summarization can be modeled as

$$D^* = \arg\min_{D \subseteq G} \sum_{s \in D} d(s, q) \qquad (1)$$
$$\text{s.t. } D \text{ is a dominating set of } G.$$

Note that $d(s, q)$ can be viewed as the weight of vertex in G. Here the summary length is minimized implicitly, since if $D' \subseteq D$, then

$\sum_{s \in D'} d(s,q) \leq \sum_{s \in D} d(s,q)$. The problem in Eq.(1) is exactly a variant of the minimum dominating set problem, i.e., the minimum weighted dominating set problem (MWDS).

Similar to MDS, MWDS can be reduced from the weighted version of the SC problem. In the weighted version of SC, each set has a weight and the sum of weights of selected sets needs to be minimized. To generate an approximate solution for the weighted SC problem, instead of choosing a set i maximizing $|SET(i)|$, a set i minimizing $\frac{w(i)}{|SET(i)|}$ is chosen, where $SET(i)$ is composed of uncovered elements in set i, and $w(i)$ is the weight of set i. The approximate solution has the same approximation ratio as that for MDS, as stated by the following theorem (Chvatal, 1979).

Theorem 4.1. *An approximate weighted dominating set can be generated with a size at most* $1+\log\Delta\cdot|OPT|$, *where* Δ *is the maximal degree of the graph and* OPT *is the optimal weighted dominating set.*

Accordingly, from generic summarization to query-focused summarization, we just need to modify line 6 in Algorithm 1 to

$$v^* = \arg\min_{v} \frac{w(v)}{s(v)}, \tag{2}$$

where $w(v)$ is the weight of vertex v. A graphical illustration of query-focused summarization using the minimum dominating set is shown in Figure 1(b).

4.4 Update Summarization

Give a query q and two sets of documents C_1 and C_2, update summarization is to generate a summary of C_2 based on q, given C_1. Firstly, summary of C_1, referred as D_1 can be generated. Then, to generate the update summary of C_2, referred as D_2, we assume D_1 and D_2 should represent all query related sentences in C_2, and length of D_2 should be minimized.

Let G_1 be the sentence graph for C_1. First we use the method described in Section 4.3 to extract sentences from G_1 to form D_1. Then we expand G_1 to the whole graph G using the second set of documents C_2. G is then the graph presentation of the document set including C_1 and C_2. We can model the update summary of C_2 as

$$D^* = \arg\min_{D_2} \textstyle\sum_{s \in D_2} w(s) \tag{3}$$
$$\text{s.t. } D_2 \cup D_1 \text{ is a dominating set of } G.$$

Intuitively, we extract the smallest set of sentences that are closely related to the query from C_2 to complete the partial dominating set of G generated from D_1. A graphical illustration of update summarization using the minimum dominating set is shown in Figure 1(c).

4.5 Comparative Summarization

Comparative document summarization aims to summarize the differences among comparable document groups. The summary produced for each group should emphasize its difference from other groups (Wang et al., 2009a).

We extend our method for update summarization to generate the discriminant summary for each group of documents. Given N groups of documents $C_1, C_2, \ldots, C_N$, we first generate the sentence graphs $G_1, G_2, \ldots, G_N$, respectively. To generate the summary for $C_i, 1 \leq i \leq N$, we view C_i as the update of all other groups. To extract a new sentence, only the one connected with the largest number of sentences which have no representatives in any groups will be extracted. We denote the extracted set as the complementary dominating set, since for each group we obtain a subset of vertices dominating those are not dominated by the dominating sets of other groups. To perform comparative summarization, we first extract the standard dominating sets for $G_1, \ldots, G_N$, respectively, denoted as $D_1, \ldots, D_N$. Then we extract the so-called complementary dominating set CD_i for G_i by continuing adding vertices in G_i to find the dominating set of $\cup_{1 \leq j \leq N} G_j$ given $D_1, \ldots, D_{i-1}, D_{i+1}, \ldots, D_N$. A graphical illustration of comparative summarization is shown in Figure 1(d).

	DUC04	DUC05	DUC06	TAC08 A	TAC08 B
Type of Summarization	Generic	Topic-focused	Topic-focused	Topic-focused	Update
#topics	NA	50	50	48	48
#documents per topic	10	25-50	25	10	10
Summary length	665 bytes	250 words	250 words	100 words	100 words

Table 1: Brief description of the data set

5 Experiments

We have conducted experiments on all four summarization tasks and our proposed methods based on the minimum dominating set have outperformed many existing methods. For the generic, topic-focused and update summarization tasks, the experiments are perform the DUC data sets using ROUGE-2 and ROUGE-SU (Lin and Hovy, 2003) as evaluation measures. For comparative summarization, a case study as in (Wang et al., 2009a) is performed. Table 1 shows the characteristics of the data sets. We use DUC04 data set to evaluate our method for generic summarization task and DUC05 and DUC06 data sets for query-focused summarization task. The data set for update summarization, (i.e. the main task of TAC 2008 summarization track) consists of 48 topics and 20 newswire articles for each topic. The 20 articles are grouped into two clusters. The task requires to produce 2 summaries, including the initial summary (TAC08 A) which is standard query-focused summarization and the update summary (TAC08 B) under the assumption that the reader has already read the first 10 documents.

We apply a 5-fold cross-validation procedure to choose the threshold λ used for generating the sentence graph in our method.

5.1 Generic Summarization

We implement the following widely used or recent published methods for generic summarization as the baseline systems to compare with our proposed method (denoted as MDS). (1) Centroid: The method applies MEAD algorithm (Radev et al., 2004) to extract sentences according to the following three parameters: centroid value, positional value, and first-sentence overlap. (2) LexPageRank: The method first constructs a sentence connectivity graph based on cosine similarity and then selects important sentences based on the concept of eigenvector centrality (Erkan and Radev, 2004). (3) BSTM: A Bayesian sentence-based topic model making use of both the term-document and term-sentence associations (Wang et al., 2009b).

Our method outperforms the simple Centroid method and another graph-based LexPageRank, and its performance is close to the results of the Bayesian sentence-based topic model and those of the best team in the DUC competition. Note however that, like clustering or topic based methods, BSTM needs the topic number as the input, which usually varies by different summarization tasks and is hard to estimate.

5.2 Query-Focused Summarization

We compare our method (denoted as MWDS) described in Section 4.3 with some recently published systems. (1) TMR (Tang et al., 2009): incorporates the query information into the topic model, and uses topic based score and term frequency to estimate the importance of the sentences. (2) SNMF (Wang et al., 2008): calculates sentence-sentence similarities by sentence-level semantic analysis, clusters the sentences via symmetric non-negative matrix factorization, and extracts the sentences based on the clustering result. (3) Wiki (Nastase, 2008): uses Wikipedia as external knowledge to expand query and builds the connection between the query and the sentences in documents.

Table 3 presents the experimental comparison of query-focused summarization on the two datasets. From Table 3, we observe that our method is comparable with these systems. This is due to the good interpretation of the summary extracted by our method, an ap-

	DUC04	
	ROUGE-2	ROUGE-SU
DUC Best	0.09216	0.13233
Centroid	0.07379	0.12511
LexPageRank	0.08572	0.13097
BSTM	0.09010	0.13218
MDS	0.08934	0.13137

Table 2: Results on generic summarization.

	DUC05		DUC06	
	ROUGE-2	ROUGE-SU	ROUGE-2	ROUGE-SU
DUC Best	0.0725	0.1316	0.09510	0.15470
SNMF	0.06043	0.12298	0.08549	0.13981
TMR	0.07147	0.13038	0.09132	0.15037
Wiki	0.07074	0.13002	0.08091	0.14022
MWDS	0.07311	0.13061	0.09296	0.14797

Table 3: Results on query-focused summarization.

proximate minimal dominating set of the sentence graph. On DUC05, our method achieves the best result; and on DUC06, our method outperforms all other systems except the best team in DUC. Note that our method based on the minimum dominating set is much simpler than other systems. Our method only depends on the distance to the query and has only one parameter (i.e., the threshold λ in generating the sentence graph).

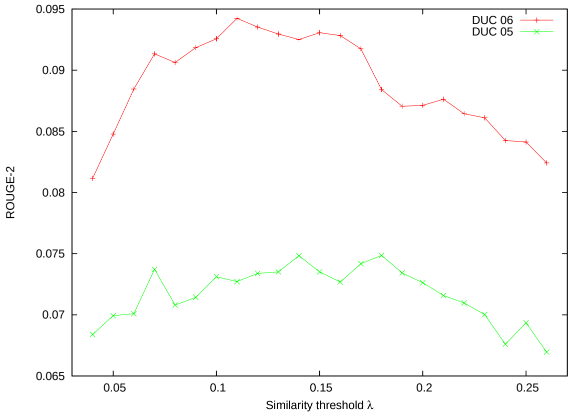


Figure 2: ROUGE-2 vs. threshold λ

We also conduct experiments to empirically evaluate the sensitivity of the threshold λ. Figure 2 shows the ROUGE-2 curve of our MWDS method on the two datasets when λ varies from 0.04 to 0.26. When λ is small, edges fail to represent the similarity of the sentences, while if λ is too large, the graph will be sparse. As λ is approximately in the range of $0.1 - 0.17$, ROUGE-2 value becomes stable and relatively high.

5.3 Update Summarization

Table 5 presents the experimental results on update summarization. In Table 5, 'TAC Best" and "TAC Median" represent the best and median results from the participants of TAC 2008 summarization track in the two tasks respectively according to the TAC 2008 report (Dang and Owczarzak, 2008). As seen from the results, the ROUGE scores of our methods are higher than the median results. The good results of the best team typically come from the fact that they utilize advanced natural language processing (NLP) techniques to resolve pronouns and other anaphoric expressions. Although we can spend more efforts on the preprocessing or language processing step, our goal here is to demonstrate the effectiveness of formalizing the update summarization problem using the minimum dominating set and hence we do not utilize advanced NLP techniques for preprocessing. The experimental results demonstrate that our simple update summarization method based on the minimum dominating set can lead to competitive performance for update summarization.

	TAC08 A		TAC08 B	
	ROUGE-2	ROUGE-SU	ROUGE-2	ROUGE-SU
TAC Best	0.1114	0.14298	0.10108	0.13669
TAC Median	0.08123	0.11975	0.06927	0.11046
MWDS	0.09012	0.12094	0.08117	0.11728

Table 5: Results on update summarization.

5.4 Comparative Summarization

We use the top six largest clusters of documents from TDT2 corpora to compare the summary generated by different comparative summarization methods. The topics of the six document clusters are as follows: topic 1: Iraq Issues; topic 2: Asia's economic crisis; topic 3: Lewinsky scandal; topic 4: Nagano Olympic Games; topic 5: Nuclear Issues in Indian and Pakistan; and topic 6: Jakarta Riot. From each of the topics, 30 documents are extracted

Topic	Complementary Dominating Set	Discriminative Sentence Selection	Dominating Set
1	$\cdots$ U.S. Secretary of State Madeleine Albright arrives to consult on the stand-off between the **United Nations and Iraq**.	**the U.S. envoy to the United Nations**, Bill Richardson, $\cdots$ play down China's refusal to support threats of **military force against Iraq**	The United States and Britain do not trust **President Saddam** and wants *cdots*warning of serious consequences if **Iraq** violates the accord.
2	**Thailand's currency**, the baht, **dropped through a key psychological level** of $\cdots$ amid a regional sell-off sparked by escalating **social unrest in Indonesia**.	Earlier, driven largely by **the declining yen**, **South Korea's stock market fell** by $\cdots$, while the **Nikkei 225 benchmark index dipped** below 15,000 in the morning $\cdots$	*In the fourth quarter, IBM Corp. earned $2.1 billion, up 3.4 percent from $2 billion a year earlier.*
3	$\cdots$ attorneys representing **President Clinton and Monica Lewinsky.**	The following night **Isikoff** $\cdots$, where he directly followed the recitation of the top-10 list: "Top 10 **White House Jobs That Sound Dirty.**"	In Washington, Ken Starr's grand jury continued its investigation of the **Monica Lewinsky matter**.
4	Eight women and six men were named Saturday night as the first **U.S. Olympic Snowboard Team** as their sport gets set to make its debut in **Nagano, Japan**.	*this tunnel is finland's cross country version of tokyo's alpine ski dome, and olympic skiers flock from russia, $\cdots$, france and austria this past summer to work out the kinks $\cdots$*	If the skiers the **men's super-G** and the **women's downhill** on Saturday, they will be back on schedule.
5	**U.S. officials** have announced **sanctions** Washington will impose on **India and Pakistan** for conducting **nuclear tests**.	The **sanctions** would stop all foreign aid except for humanitarian purposes, **ban military sales to India** $\cdots$	And **Pakistan's prime minister** says his country will sign the **U.N.'s comprehensive ban on nuclear tests** if **India** does, too.
6	$\cdots$ remain in force around **Jakarta**, and at the Parliament building where **thousands of students staged a sit-in** Tuesday $\cdots$.	"**President Suharto** has given much to his country over the past 30 years, raising **Indonesia's** standing in the world $\cdots$	*What were the students doing at the time you were there, and what was the reaction of the students to the troops?*

Table 4: A case study on comparative document summarization. Some unimportant words are skipped due to the space limit. The bold font is used to annotate the phrases that are highly related with the topics, and italic font is used to highlight the sentences that are not proper to be used in the summary.

randomly to produce a one-sentence summary. For comparison purpose, we extract the sentence with the maximal degree as the baseline. Note that the baseline can be thought as an approximation of the dominating set using only one sentence. Table 4 shows the summaries generated by our method (complementary dominating set (CDS)), discriminative sentence selection (DSS) (Wang et al., 2009a) and the baseline method. Our CDS method can extract discriminative sentences for all the topics. DSS can extract discriminative sentences for all the topics except topic 4. Note that the sentence extracted by DSS for topic 4 may be discriminative from other topics, but it is deviated from the topic Nagano Olympic Games. In addition, DSS tends to select long sentences which should not be preferred for summarization purpose. The baseline method may extract some general sentences, such as the sentence for topic 2 and topic 6 in Table 4.

6 Conclusion

In this paper, we propose a framework to model the multi-document summarization using the minimum dominating set and show that many well-known summarization tasks can be formulated using the proposed framework. The proposed framework leads to simple yet effective summarization methods. Experimental results show that our proposed methods achieve good performance on several multi-document document tasks.

7 Acknowledgements

This work is supported by NSF grants IIS-0549280 and HRD-0833093.

References

Chen, Y.P. and A.L. Liestman. 2002. Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks. In *Proceedings of International Symposium on Mobile Ad hoc Networking & Computing*. ACM.

Chvatal, V. 1979. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235.

Dang, H.T. and K Owczarzak. 2008. Overview of the TAC 2008 Update Summarization Task. In *Proceedings of the Text Analysis Conference (TAC)*.

Dang, H.T. 2007. Overview of DUC 2007. In *Document Understanding Conference*.

Daumé III, H. and D. Marcu. 2006. Bayesian query-focused summarization. In *Proceedings of the ACL-COLING*.

Erkan, G. and D.R. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In *Proceedings of EMNLP*.

Feige, U. 1998. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652.

Goldstein, J., V. Mittal, J. Carbonell, and M. Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *NAACL-ANLP 2000 Workshop on Automatic summarization*.

Guha, S. and S. Khuller. 1998. Approximation algorithms for connected dominating sets. *Algorithmica*, 20(4):374–387.

Haghighi, A. and L. Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of HLT-NAACL*.

Han, B. and W. Jia. 2007. Clustering wireless ad hoc networks with weakly connected dominating set. *Journal of Parallel and Distributed Computing*, 67(6):727–737.

Johnson, D.S. 1973. Approximation algorithms for combinatorial problems. In *Proceedings of STOC*.

Jurafsky, D. and J.H. Martin. 2008. *Speech and language processing*. Prentice Hall New York.

Kann, V. 1992. *On the approximability of NP-complete optimization problems*. PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm.

Lawrie, D., W.B. Croft, and A. Rosenberg. 2001. Finding topic words for hierarchical summarization. In *Proceedings of SIGIR*.

Lin, C.Y. and E. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of HLT-NAACL*.

Mani, I. 2001. Automatic summarization. *Computational Linguistics*, 28(2).

Nastase, V. 2008. Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation. In *Proceedings of EMNLP*.

Nenkova, A. and L. Vanderwende. 2005. The impact of frequency on summarization. *Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005-101*.

Radev, D.R., H. Jing, M. Styś, and D. Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40(6):919–938.

Raz, R. and S. Safra. 1997. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proceedings of STOC*.

Saggion, H., K. Bontcheva, and H. Cunningham. 2003. Robust generic and query-based summarisation. In *Proceedings of EACL*.

Tang, J., L. Yao, and D. Chen. 2009. Multi-topic based Query-oriented Summarization. In *Proceedings of SDM*.

Thai, M.T., N. Zhang, R. Tiwari, and X. Xu. 2007. On approximation algorithms of k-connected m-dominating sets in disk graphs. *Theoretical Computer Science*, 385(1-3):49–59.

Wan, X., J. Yang, and J. Xiao. 2007a. Manifold-ranking based topic-focused multi-document summarization. In *Proceedings of IJCAI*.

Wan, X., J. Yang, and J. Xiao. 2007b. Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction. In *Proceedings of ACL*.

Wang, D., T. Li, S. Zhu, and C. Ding. 2008. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *Proceedings of SIGIR*.

Wang, D., S. Zhu, T. Li, and Y. Gong. 2009a. Comparative document summarization via discriminative sentence selection. In *Proceeding of CIKM*.

Wang, D., S. Zhu, T. Li, and Y. Gong. 2009b. Multi-document summarization using sentence-based topic models. In *Proceedings of the ACL-IJCNLP*.

Wei, F., W. Li, Q. Lu, and Y. He. 2008. Query-sensitive mutual reinforcement chain and its application in query-oriented multi-document summarization. In *Proceedings of SIGIR*.

Wu, J. and H. Li. 2001. A dominating-set-based routing scheme in ad hoc wireless networks. *Telecommunication Systems*, 18(1):13–36.

Corpus-based Semantic Class Mining: Distributional vs. Pattern-Based Approaches

Shuming Shi[1] **Huibin Zhang**[2*] **Xiaojie Yuan**[2] **Ji-Rong Wen**[1]
[1] Microsoft Research Asia
[2] Nankai University
{shumings, jrwen}@microsoft.com
zhanghuibin@126.com; yuanxj@nankai.edu.cn

Abstract

Main approaches to corpus-based semantic class mining include *distributional similarity* (DS) and *pattern-based* (PB). In this paper, we perform an empirical comparison of them, based on a publicly available dataset containing 500 million web pages, using various categories of queries. We further propose a frequency-based rule to select appropriate approaches for different types of terms.

1 Introduction

Computing the semantic relationship between terms, which has wide applications in natural language processing and web search, has been a hot topic nowadays. This paper focuses on corpus-based semantic class mining (Lin 1998; Pantel and Lin 2002; Pasca 2004; Shinzato and Torisawa, 2005; Ohshima, et al., 2006; Zhang et al., 2009), where *peer terms* (or *coordinate terms*) are discovered from a corpus.

Existing approaches to semantic class mining could roughly be divided into two categories: *distributional similarity* (*DS*), and *pattern-based* (*PB*). The first type of work (Hindle, 1990; Lin 1998; Pantel and Lin 2002) is based on the distributional hypothesis (Harris, 1985), saying that terms occurring in analogous (lexical or syntactic) *contexts* tend to be similar. DS approaches basically exploit *second-order* co-occurrences to discover strongly associated concepts. In pattern-based approaches (Hearst 1992; Pasca 2004; Shinzato and Torisawa, 2005; Ohshima, et al., 2006; Zhang et al., 2009), patterns are applied to discover specific relationships between terms, from the general *first-order* co-occurrences. For example, "*NP such as NP, NP..., and NP*" is a popular and high-quality pattern for extracting peer terms (and also hyponyms). Besides the natural language patterns, some HTML tag tree patterns (e.g., the drop down list) are also effective in semantic class mining.

It is worth-noting that the word "*pattern*" also appears in some DS approaches (Pasca et al., 2006; Tanev and Magnini, 2006; Pennacchiotti and Pantel, 2009), to represent the *context* of a term or a term-pair, e.g., "(invent, subject-of)" for the term "Edison", and "- starring -" for the term-pair "(The Terminal, Tom Hanks)". Although "*patterns*" are utilized, we categorize them as DS approaches rather than PB, because they match the DS framework well. In this paper, *PB* only refers to the approaches that utilize patterns to exploit *first-order* co-occurrences. And the patterns in DS approaches are called *contexts* in the following part of this paper.

Progress has been made and promising results have been reported in the past years for both DS and PB approaches. However, most previous research work (some exceptions are discussed in related work) involves solely one category of approach. And there is little work studying the comparison of their performance for different types of terms (we use "*term*" to represent a single word or a phrase).

In this paper, we make an empirical study of this problem, based on a large-scale, publicly available dataset containing 500 million web pages. For each approach *P*, we build a *term-similarity graph* *G*(*P*), with vertices representing terms, and edges being the confidence that the two terms are peers. Approaches are compared by the quality of their corresponding term graphs.

[*] Work done during an internship at Microsoft

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 993–1001,
Beijing, August 2010

We measure the quality of a term graph by set expansion. Two query sets are adopted: One contains 49 semantic classes of named entities and 20220 trials (queries), collected by Pantel et al. (2009) from Wikipedia[2]; and the other contains 100 queries of five lexical categories (proper nouns, common nouns, verbs, adjectives, and adverbs), built in this paper for studying the performance comparison on different term types. With the dataset and the query sets, we study the comparison of DS and PB. Key observations and preliminary conclusions are,

- ***DS vs. PB***: DS approaches perform much better on common nouns, verbs, adjectives, and adverbs; while PB generates higher-quality semantic classes for proper nouns.
- ***Lexical vs. Html-tag patterns***: If only lexical patterns are adopted in PB, the performance drops significantly; while the performance only becomes slightly worse with only Html-tag patterns being included.
- ***Corpus-size***: For proper nouns, PB beats DS even based on a much smaller corpus; similarly, for other term types, DS performs better even with a smaller corpus.

Given these observations, we further study the feasibility of selecting appropriate approaches for different term types to obtain better results. A simple and effective frequency-based rule is proposed for approach-selection. Our online semantic mining system (*NeedleSeek*)[3] adopts both PB and DS to build semantic classes.

2 Related Work

Existing efforts for semantic class mining has been done upon various types of data sources, including text-corpora, search-results, and query logs. In corpus-based approaches (Lin 1998; Lin and Pantel 2001; Pantel and Lin 2002; Pasca 2004; Zhang et al., 2009), semantic classes are obtained by the offline processing of a corpus which can be unstructured (e.g., plain text) or semi-structured (e.g., web pages). Search-results-based approaches (Etzioni et al., 2004; Kozareva et al., 2008; Wang and Cohen, 2008) assume that multiple terms (or, less often, one term) in a semantic class have been provided as *seeds*. Other terms in the class are retrieved by sending queries (constructed according to the seeds) to a web search engine and mining the search results. Query logs are exploited in (Pasca 2007; Komachi and Suzuki, 2008; Yamaguchi 2008) for semantic class mining. This paper focuses on corpus-based approaches.

As has been mentioned in the introduction part, primarily two types of methodologies are adopted: DS and PB. Syntactic context information is used in (Hindle, 1990; Ruge, 1992; Lin 1998; Lin and Pantel, 2001; Pantel and Lin, 2002) to compute term similarities. The construction of syntactic contexts requires sentences to be parsed by a dependency parser, which may be extremely time-consuming on large corpora. As an alternative, lexical context (such as text window) has been studied (Pantel et al., 2004; Agirre et al., 2009; Pantel et al., 2009). In the pattern-based category, a lot of work has been done to discover term relations by sentence lexical patterns (Hearst 1992; Pasca 2004), HTML tag patterns (Shinzato and Torisawa, 2005), or both (Shi et al., 2008; Zhang et al., 2009). In this paper, our focus is not one specific methodology, but the comparison and combination of them.

A small amount of existing work is related to the comparison or combination of multiple methods. Pennacchiotti and Pantel (2009) proposed a feature combination framework (named ensemble semantic) to combine features generated by different extractors (distributional and "pattern-based") from various data sources. As has been discussed in the introduction, in our terminology, their "pattern-based" approaches are actually DS for term-pairs. In addition, their study is based on three semantic classes (actors, athletes, and musicians), all of which are proper nouns. Differently, we perform the comparison by classifying terms according to their lexical categories, based on which additional insights are obtained about the pros and cons of each methodology. Pantel et al., (2004) proposed, in the scenario of extracting *is-a* relations, one pattern-based approach and compared it with a baseline syntactic distributional similarity method (called *syntactic co-occurrence* in their paper). Differently, we study the comparison in a different scenario (semantic class mining). In addition, they did not differentiate the lexical types of terms in the study. The third difference is that we proposed a rule for method-selection while they did not. In (Pasca and Durme,

[2] http://www.wikipedia.org/

[3] http://needleseek.msra.cn/

2008), clusters of distributional similar terms were adopted to expand the labeled semantic classes acquired from the "such as | including" pattern. Although both patterns and distributional similarity were used in their paper, they did not do any comparison about their performance. Agirre et al. (2009) compared DS approaches with WordNet-based methods in computing word similarity and relatedness; and they also studied the combination of them. Differently, the methods for comparison in our paper are DS and PB.

3 Similarity Graph Construction

A key operation in corpus-based semantic class mining is to build a term similarity graph, with vertices representing terms, and edges being the similarity (or distance) between terms. Given the graph, a clustering algorithm can be adopted to generate the final semantic classes. Now we describe the state-of-the-art DS and PB approaches for computing term similarities.

3.1 Distributional Similarity

DS approaches are based on the distributional hypothesis (Harris, 1985), which says that terms appearing in analogous contexts tend to be similar. In a DS approach, a term is represented by a feature vector, with each feature corresponding to a context in which the term appears. The similarity between two terms is computed as the similarity between their corresponding feature vectors. Different approaches may have different ways of 1) defining a context, 2) assigning feature values, or 3) measuring the similarity between two feature vectors.

Contexts	Text window (window size: 2, 4)
	Syntactic
Feature value	PMI
Similarity measure	Cosine, Jaccard

Table 1. DS approaches implemented in this paper

Mainly two kinds of contexts have been extensively studied: *syntactic context* and *lexical context*. The construction of *syntactic contexts* relies on the syntactic parsing trees of sentences, which are typically the output of a syntactic parser. Given a syntactic tree, a syntactic context of a term *w* can be defined as the parent (or one child) of *w* in the tree together with their relationship (Lin, 1998; Pantel and Lin, 2002; Pantel et al., 2009). For instance, in the syntactic tree of sentence "*this is an interesting read for anyone studying logic*", one context of the word "logic" can be defined as "*study V:obj:N*". In this paper, we adopt Minipar (Lin, 1994) to parse sentences and to construct syntactic trees.

One popular *lexical context* is *text window*, where a context *c* for a term *w* in a sentence *S* is defined as a substring of the sentence containing but removing *w*. For example, for sentence "$\ldots w_1 w_2 w_3 w w_4 w_5 w_6 \ldots$", a text window context (with size 4) of *w* can be "$w_2 w_3 w_4 w_5$". It is typically time-consuming to construct the syntactic trees for a large-scale dataset, even with a lightweight syntactic parser like Minipar. The construction of lexical contexts is much more efficient because it does not require the syntactic dependency between terms. Both contexts are studied in this paper.

After defining contexts for a term *w*, the next step is to construct a feature vector for the term: $F(w)=(\mathrm{f}_{w1}, \mathrm{f}_{w2} \ldots, \mathrm{f}_{w,m})$, where *m* is the number of distinct contexts, and $\mathrm{f}_{w,c}$ is the feature value of context *c* with respect to term *w*. Among all the existing approaches, the dominant way of assigning feature values (or context values) is computing the pointwise mutual information (PMI) between the feature and the term,

$$\mathrm{f}_{w,c} = \mathrm{PMI}_{w,c} = \log \frac{F(w,c) \cdot F(*,*)}{F(w,*) \cdot F(*,c)} \qquad (3.1)$$

where $F(w,c)$ is the frequency of context *c* occurring for term *w*, $F(w,*)$ is the total frequency of all contexts for term *w*, $F(*,c)$ is the frequency of context *c* for all terms, and $\mathrm{F}(*,*)$ is the total frequency of all context for all terms. They are calculated as follows respectively,

$$\begin{aligned} F(w,*) &= \textstyle\sum_{j=1}^{m} F(w,j) \\ F(*,c) &= \textstyle\sum_{i=1}^{n} F(i,c) \qquad (3.2) \\ F(*,*) &= \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} F(i,j) \end{aligned}$$

where *m* and *n* are respectively the distinct numbers of contexts and terms.

Following state-of-the-art, we adopt PMI in this paper for context weighting.

Given the feature vectors of terms, the similarity of any two terms is naturally computed as the similarity of their corresponding feature vectors. Cosine similarity and Jaccard similarity (weighted) are implemented in our experiments,

$$Cosine(\vec{x}, \vec{y}) = \frac{\sum_i x_i y_i}{\sqrt{\sum_i {x_i}^2} \cdot \sqrt{\sum_i {y_i}^2}} \qquad (3.3)$$

$$Jaccard(\vec{x},\vec{y}) = \frac{\sum_i \min(x_i,y_i)}{\sum_i x_i + \sum_i y_i - \sum_i \min(x_i,y_i)} \quad (3.4)$$

Jaccard similarity is finally used in presenting our experimental results (in Section 6), because it achieves higher performance.

3.2 Pattern-based Approaches

In PB approaches, a list of carefully-designed (or automatically learned) patterns is exploited and applied to a text collection, with the hypothesis that the terms extracted by applying each of the patterns to a specific piece of text tend to be similar. Two categories of patterns have been studied in the literature: sentence lexical patterns, and HTML tag patterns. Table-2 lists some popular patterns utilized in existing semantic class mining work (Heast 1992; Pasca 2004; Kozareva et al., 2008; Zhang et al., 2009). In the table, "T" means a term (a word or a phrase). Exactly the same set of patterns is employed in implementing our pattern-based approaches in this paper.

Type	Pattern
Lexical	T {, T}*{,} (and|or) {other} T
	(such as | including) T (and|,|.)
	T, T, T {,T}*
Tag	<ul> <li> T </li> … <li> T </li> </ul>
	<ol> <li> T </li> … <li> T </li> </ol>
	<select> <option> T …<option> T </select>
	<table> <tr> <td> T </td> … <td> T </td> </tr> ... </table>
	Other Html-tag repeat patterns

Table 2. Patterns employed in this paper (Lexical: sentence lexical patterns; Tag: HTML tag patterns)

We call the set of terms extracted by applying a pattern one time as a *raw semantic class* (RASC). The term similarity graph needs to be built by aggregating the information of the extracted RASCs.

One basic idea of estimating term similarity is to count the number of RASCs containing both of them. This idea is extended in the state-of-the-art approaches (Zhang et al., 2009) to distinguish the reliability of different patterns and to punish term similarity contributions from the same domain (or site), as follows,

$$Sim(a,b) = \sum_{i=1}^{m} \log(1 + \sum_{j=1}^{k_i} w(P(C_{i,j}))) \quad (3.5)$$

where $C_{i,j}$ is a RASC containing both term a and term b, $P(C_{i,j})$ is the pattern via which the RASC is extracted, and $w(P)$ is the weight of pattern P. The above formula assumes all these RASCs belong to m sites (or domains) with $C_{i,j}$ extracted from a page in site i, and k_i being the number of RASCs corresponding to site i.

In this paper, we adopt an extension of the above formula which considers the frequency of a single term, as follows,

$$Sim^*(a,b) = Sim(a,b) \cdot \sqrt{IDF(a) \cdot IDF(b)} \quad (3.6)$$

where $IDF(a)$=log(1+N/$N(a)$), N is the total number of RASCs, and $N(a)$ is the number of RASCs containing a. In the experiments, we simply set the weight of every pattern type to be the same value (1.0).

4 Compare PB and DS

We compare PB and DS by the quality of the term similarity graphs they generated. The quality of a term graph is measured by set expansion: Given a list of *seed* terms (e.g., S={*lent*, *epiphany*}) belonging to a semantic class, our task is to find other members of this class, such as *advent*, *easter*, and *christmas*.

In this section, we first describe our set expansion algorithm adopted in our study. Then DS and PB are compared in terms of their set-expansion performance. Finally we discuss ways of selecting appropriate approaches for different types of seeds to get better expansion results.

4.1 Set Expansion Algorithm

Having at hand the similarity graph, set expansion can be implemented by selecting the terms most similar to the seeds. So given a query Q={s_1, s_2, …, s_k}, the key is to compute $f(t,Q)$, the similarity between a term t and the seed-set Q. Naturally, we define it as the weighted average similarity between t and every seed in Q,

$$f(t,Q) = \sum_{i=1}^{k} w_i \cdot Sim(t,s_i) \quad (4.1)$$

where w_i is the weight of seed s_i, which can be a constant value, or a function of the frequency of term s_i in the corpus. Although Formula 3.6 can be adopted directly for calculating $Sim(t,s_i)$, we use the following rank-based formula because it generate better expansion results.

$$Sim(t,s_i) = \frac{1}{\log(\lambda + r(t,s_i))} \quad (4.2)$$

where $r(t,s_i)$ is the rank of term t among the neighbors of s_i.

In our experiments, we fix w_i=1 and λ=10.

4.2 Compare DS with PB

In order to have a comprehensive comparison of the two approaches, we intentionally choose terms of diverse types and do experiments based on various data scales. We classify terms into 5 types by their lexical categories: proper nouns, common nouns, verbs, adjectives, and adverbs. The data scales for experiments are from one million to 500 million web pages. Please refer to sections 5.1 and 5.2 for more details about the corpora and seeds used for experiments.

Experimental results (refer to Section 6) will show that, for proper nouns, the ranking of approaches (in terms of performance) is:

$$PB > PB\text{-}HtmlTag > DS \approx PB\text{-}Lexical$$

While for common nouns, verbs, adjectives, and adverbs, we have:

$$DS > PB$$

Here "PB-lexical" means only the lexical patterns of Table 2 are adopted. Similarly, "PB-HtmlTag" represents the PB approach with only Html-tag patterns being utilized.

Please pay attention that this paper by no means covers all PB or DS approaches (although we have tried our best to include the most popular ones). For PB, there are of course other kinds of patterns (e.g., patterns based on deeper linguistic analysis). For DS, other types of contexts may exist in addition to those listed in Table 1. So in interpreting experimental results, making observations, and drawing preliminary conclusions, we only means the patterns in Table 2 for PB and Table 1 for DS. It will be an interesting future work to include more DS and PB approaches in the study.

In order to understand why PB performs so well in dealing with proper nouns while so badly for other term categories, we calculated the frequency of each seed term in the extracted *RASCs*, the output of the pattern-matching algorithm. We define the *normalized frequency* of a term to be its frequency in the RASCs divided by the frequency in the sentences of the original documents (with duplicate sentences merged). Then we define the *mean normalized frequency* (MNF) of a seed set *S*, as follows,

$$MNF(S) = \frac{\sum_{t \in S} F_{norm}(t)}{|S|} \tag{4.3}$$

where $F_{norm}(t)$ is the normalized frequency of *t*.

The MNF values for the five seed sets are listed in Table 3, where we can see that proper nouns have the largest MNF values, followed by common nouns. In other words, the patterns in Table 2 capture the relations of more proper nouns than other term categories.

Seed Categories	Terms	MNF
Proper nouns	40	0.2333
Common nouns	40	0.0716
Verbs	40	0.0099
Adjectives	40	0.0126
Adverbs	40	0.0053

Table 3. MNF values of different seed categories

As mentioned in the introduction, the PB and DS approaches we studied capture first-order and second-order term co-occurrences respectively. Some existing work (e.g., Edmonds, 1997) showed that second-order co-occurrence leads to better results for detecting synonymy. Considering that a high proportion of coordinate terms of verbs, adjectives, and adverbs are their synonyms and antonyms, it is reasonable that DS behaves better for these term types because it exploits second-order co-occurrence. For PB, different from the standard way of dealing with first-order co-occurrences where statistics are performed on all pairs of near terms, a *subset* of co-occurred terms are *selected* in PB by specific patterns. The patterns in Table-2 help detecting coordinate proper nouns, because they are frequently occurred together obeying the patterns in sentences or web pages. But it is not the case for other term types. It will be interesting to study the performance of PB when more pattern types are added.

4.3 Approach Selection

Having observed that the two approaches perform quite differently on every type of queries we investigated, we hope we can improve the expansion performance by smartly selecting an approach for each query. In this section, we propose and study several approach-selection methods, by which we hope to gain some insights about the possibility and effectiveness of combining DS and PB for better set expansion.

Oracle selection: In order to get an insight about the upper bound that we could obtain when combing the two methods, we implement an oracle that chooses, for each query, the approach that generates better expansion results.

Frequency-based selection: It is shown in Table 3 that the *mean normalized frequency* of proper nouns is much larger than other terms. Motivated by this observation, we select a set expansion methodology for each query as follows: Select PB if the normalized frequency values of all terms in the query are larger than 0.1; otherwise choose DS.

We demonstrate, in Section 6.3, the effectiveness of the above selection methods.

5 Experimental Setup

5.1 Dataset and Exp. Environment

We adopt a public-available dataset in our experiments: ClueWeb09[4]. This is a very large dataset collected by Carnegie Mellon University in early 2009 and has been used by several tracks of the Text Retrieval Conference (TREC)[5]. The whole dataset consists of 1.04 billion web pages in ten languages while only those in English, about 500 million pages, are used in our experiments. The reason for selecting such a dataset is twofold: First, it is a corpus large enough for conducting web-scale experiments and getting meaningful results. Second, since it is publicly available, it is possible for other researchers to reproduce the experiments in the paper.

Corpora	Docs (millions)	Sentences (millions)	Description
Clue500	500	13,000	All En pages in ClueWeb09
Clue050	50	1,600	ClueWeb09 category B
Clue010	10	330	Sampling from Clue050
Clue001	1	42	Sampling from Clue050

Table 4. Corpora used in experiments

To test the impact of corpus size on set expansion performance, four corpora are derived from the dataset, as outlined in Table 4. The Clue500 corpus contains all the 500 million English web pages in the dataset; while Clue050 is a subset of ClueWeb09 (named category B) containing 50 million English web pages. The remaining two corpora are respectively the 1/5 and 1/50 random sampling of web pages from Clue050.

Documents in the corpora are stored and processed in a cluster of 40 four-core machines.

5.2 Query Sets

We perform our study using two query sets.

WikiGold: It was collected by Pantel et al. (2009) from the “List of” pages in Wikipedia and used as the gold standard in their paper. This gold standard consists of 49 entity sets, and 20220 trials (used as queries) of various numbers of seeds. Most seeds in the query set are named entities. Please refer to Pantel et al. (2009) for details of the gold standard.

Mix100: This query set consists of 100 queries in five categories: verbs, adjectives, adverbs, common nouns, and proper nouns. There are 20 queries in every category and two seeds in every query. The query set was built by the following steps: First, 20 terms of each category were randomly selected from a term list (which is constructed by part-of-speech tagging the Clue050 corpus and removing low-frequency terms), and were treated as the first seed of the each query. Then, we manually added one additional seed for each query. The reason for utilizing two seeds instead of one is the observation that a large portion of the terms selected in the previous step belong to multiple categories. For example, “*colorful*” is both an adjective and a proper noun (a Japanese manga).

5.3 Results Labeling

No human labeling efforts are needed for the expansion results of the WikiGold query set. Every returned term is automatically judged to be “Good” (otherwise “Bad”) if it appears in the corresponding gold standard entity set.

For Mix100, the search results of various approaches are merged and labeled by three human labelers. Each labeler assigns each term in the search results a label of “Good”, “Fair” or “Bad”. The labeling agreement values (measured by percentage agreement) between labelers I and II, I and III, II and III are respectively 0.82, 0.81, and 0.81. The ultimate judgment of each result term is obtained from the three labelers by majority voting. In the case of three labelers giving mutually different results (i.e., one “Good”, one “Fair” and one “Bad”), the ultimate judgment is set to “Fair” (the average).

5.4 Evaluation Metrics

After removing seeds from the expansion results, we adopt the following metrics to evaluate the

[4] http://boston.lti.cs.cmu.edu/Data/clueweb09/

[5] http://trec.nist.gov/

results of each query. The evaluation score on a query set is the average over all the queries.

Precision@*k*: The percentage of relevant (good or fair) terms in the top-*k* expansion results (terms labeled as "Fair" are counted as 0.5)

Recall@*k*: The ratio of relevant terms in the top-*k* results to the total number of relevant terms

R-Precision: Precision@R where *R* is the total number of terms labeled as "Good"

Mean average precision (MAP): The average of precision values at the positions of all good or fair results

6 Experimental Results

6.1 Overall Performance Comparison

Table 5 lists the performance (measured by MAP, R-precision, and the precisions at ranks 25, 50, and 100) of some key approaches on corpus Clue050 and query set WikiGold. The results of query set Mix100 are shown in Table 6. In the results, *TWn* represents the DS approach with text-window of size *n* as contexts, *Syntactic* is the DS approach with syntactic contexts, *PB-Lexical* means only the lexical patterns of Table 2 are adopted, and *PB-HtmlTag* represents the PB approach with only Html-tag patterns utilized.

Approach	MAP	R-Prec	P@25	P@50	P@100
TW2	0.218	0.287	0.359	0.278	0.204
TW4	0.152	0.210	0.325	0.244	0.173
Syntactic	0.170	0.247	0.314	0.242	0.178
PB-Lexical	0.227	0.276	0.352	0.272	0.190
PB-HtmlTag	0.354	0.417	0.513	0.413	0.311
PB	**0.362**	**0.424**	**0.520**	**0.418**	**0.314**
Pantel-24M	N/A	0.264	0.353	0.298	0.239
Pantel-120M	N/A	0.356	0.377	0.319	0.250
Pantel-600M	N/A	0.404	0.407	0.347	0.278

Table 5. Performance comparison on the Clue050 corpus (query set: WikiGold)

It is shown that PB gets much higher evaluation scores than other approaches on the *WikiGold* query set and the proper-nouns category of *Mix100*. While for other seed categories in *Mix100*, TW2 return significantly better results. We noticed that most seeds in *WikiGold* are proper nouns. So the experimental results tend to indicate that the performance comparison between state-of-the-art DS and PB approaches depends on the types of terms to be mined, specifically, DS approaches perform better in mining semantic classes of common nouns, verbs, adjectives, and adverbs; while state-of-the-art PB approaches are more suitable for mining semantic classes of proper nouns. The performance of PB is low in dealing with other types of terms (especially adverbs). The performance of PB drops significantly if only lexical patterns are used; and the HtmlTag-only version of PB performs only slightly worse than PB.

The observations are verified by the precision-recall graph in Figure 1 on Clue500. The results of the *syntactic* approach on Clue500 are not included, because it is too time-consuming to parse all the 500 million web pages by a dependency parser (even using a high-performance parser like Minipar). It took overall about 12,000 CPU-hours to parse all the sentences in Clue050 by Minipar.

Query types & Approaches		MAP	P@5	P@10	P@20
Proper Nouns	TW2	0.302	0.835	0.810	0.758
	PB	**0.336**	**0.920**	**0.838**	**0.813**
Common Nouns	TW2	**0.384**	**0.735**	**0.668**	**0.595**
	PB	0.212	0.640	0.548	0.485
Verbs	TW2	**0.273**	**0.655**	**0.543**	**0.465**
	PB	0.176	0.415	0.373	0.305
Adjectives	TW2	**0.350**	**0.655**	**0.563**	**0.473**
	PB	0.120	0.335	0.285	0.234
Adverbs	TW2	**0.432**	**0.605**	**0.505**	**0.454**
	PB	0.043	0.100	0.095	0.089

Table 6. Performance comparison on different query types (Corpus: Clue050; query set: Mix100)

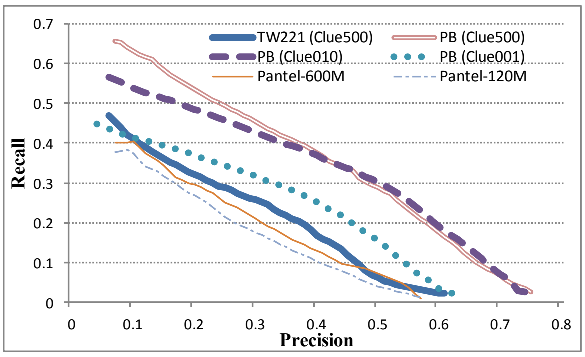


Figure 1. Precision and recall of various approaches (query set: WikiGold)

The methods labeled Pantel-24M etc. (in Table 5 and Figure 1) are the approaches presented in (Pantel et al., 2009) on their corpus (called Web04, Web20, and Web100 in the paper) containing respectively 24 million, 120 million, and 600 million web pages. Please pay attention that their results and ours may not be directly comparable, because different corpora and set-expansion algorithms were used. Their results are listed here for reference purpose only.

6.2 Corpus Size Effect

Table 7 shows the performance (measured by MAP) of two approaches on query set *Mix100*, by varying corpus size. We observed that the performance of TW2 improves rapidly along with the growth of corpus size from one million to 50 million documents. From Clue050 to Clue500, the performance is slightly improved.

Query types & Approaches		Clue001	Clue010	Clue050	Clue500
Proper Nouns	TW2	0.209	0.265	0.302	**0.311**
	PB	**0.355**	0.351	0.336	0.327
Common Nouns	TW2	0.259	0.348	0.384	**0.393**
	PB	0.200	**0.234**	0.212	0.205
Verbs	TW2	0.224	0.268	0.273	**0.278**
	PB	0.101	0.134	**0.176**	0.148
Adjectives	TW2	0.309	0.326	0.350	**0.353**
	PB	0.077	**0.158**	0.120	0.129
Adverbs	TW2	0.413	0.423	0.432	**0.437**
	PB	0.028	0.058	0.043	**0.059**

Table 7. Effect of different corpus size (query set: Mix100; metric: MAP)

For PB, however, the performance change is not that simple. For proper nouns, the best performance (in terms of MAP) is got on the two small corpora Clue001 and Clue010; and the score does not increase when corpus size grows. Different observations are made on WikiGold (see Figure 1), where the performance improves a lot with the data growth from Clue001 to Clue010, and then stabilizes (from Clue010 to Clue500). For other term types, the MAP scores do not grow much after Clue010. To our current understanding, the reason may be due to the two-fold effect of incorporating more data in mining: *bringing useful information as well as noise*. Clue001 contains enough information, which is fully exploited by the PB approach, for expanding the proper-nouns in Mix100. So the performance of PB on Clue001 is excellent. The named entities in WikiGold are relatively rare, which requires a larger corpus (Clue010) for extracting peer terms from. But when the corpus gets larger, we may not be able to get more useful information to further improve results quality.

Another interesting observation is that, for proper nouns, the performance of PB on Clue001 is even much better than that of TW2 on corpus Clue500. Similarly, for other query types (common nous, verbs, adjectives, and adverbs), TW2 easily beats PB even with a much smaller corpus.

6.3 Approach Selection

Here we demonstrate the experimental results of combining DS and PB with the methods we proposed in Section 4.3. Table 8 shows the combination of PB and TW2 on corpus Clue050 and query set Mix100. The overall performance relies on the number (or percentage) of queries in each category. Two ways of mixing the queries are tested: avg(4:1:1:1:1) and avg(1:1:1:1:1), where the numbers are the proportion of proper nouns, common nouns, verbs, adjectives, and adverbs.

Approach	Avg (1:1:1:1:1)			Avg (4:1:1:1:1)		
	P@5	P@10	P@20	P@5	P@10	P@20
TW2	0.697	0.618	0.548	0.749	0.690	0.627
PB	0.482	0.428	0.385	0.646	0.581	0.545
Oracle	0.759	0.663	0.591	0.836	0.759	0.695
Freq-based	0.721	0.633	0.570	0.799	0.723	0.671

Table 8. Experiments of combining both approaches (Corpus: Clue050; query set: Mix100)

The expansion performance is improved a lot with our frequency-based combination method. As expected, oracle selection achieves great performance improvement, which shows the large potential of combining DS and PB. Similar results (omitted due to space limitations) are observed on the other corpora.

Our online semantic mining system (*NeedleSeek*, http://needleseek.msra.cn) adopts both PB and DS for semantic class construction.

7 Conclusion

We compared two mainstream methods (DS and PB) for semantic class mining, based on a dataset of 500 million pages and using five term types. We showed that PB is clearly adept at extracting semantic classes of proper nouns; while DS is relatively good at dealing with other types of terms. In addition, a small corpus is sufficient for each approach to generate better semantic classes of its "favorite" term types than those obtained by its counterpart on a much larger corpus. Finally, we tried a frequency-based method of combining them and saw apparent performance improvement.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, Aitor Soroa. A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. NAACL-HLT 2009.

Philip Edmonds. 1997. Choosing the Word most Typical in Context Using a Lexical Co-Occurrence Network. *ACL'97*, pages 507-509.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel Weld, and Alexander Yates. 2004. Web-Scale Information Extraction in KnowItAll. *WWW'04*, pages 100–110, New York.

Zelig S. Harris. 1985. Distributional Structure. *The Philosophy of Linguistics*. New York: Oxford University Press.

Marti A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. *COLING'92*, Nantes, France.

Donald Hindle. 1990. Noun Classification from Predicate-Argument Structures. In ACL'90, pages 268–275, Pittsburg, Pennsylvania, June.

Mamoru Komachi and Hisami Suzuki. Minimally Supervised Learning of Semantic Knowledge from Query Logs. *IJCNLP 2008*, pages 358–365, 2008.

Zornitsa Kozareva, Ellen Riloff, Eduard Hovy. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. ACL'08: HLT.

Dekang Lin. 1994. Principar - an Efficient, Broad-Coverage, Principle-based Parser. *COLING'94*, pp. 482-488.

Dekang Lin. 1998. Automatic Retrieval and Clustering of Similar Words. *COLING-ACL'98*, pages 768-774.

Dekang Lin and Patrick Pantel. 2001. Induction of Semantic Classes from Natural Language Text. *SIGKDD'01*, pages 317-322.

Hiroaki Ohshima, Satoshi Oyama and Katsumi Tanaka. 2006. Searching Coordinate Terms with their Context from the Web. *WISE'06*.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu and Vishnu Vyas. 2009. Web-Scale Distributional Similarity and Entity Set Expansion. *EMNLP'09.* Singapore.

Patrick Pantel and Dekang Lin. 2002. Discovering Word Senses from Text. *SIGKDD'02*.

Patric Pantel, Deepak Ravichandran, and Eduard Hovy. 2004. Towards Terascale Knowledge Acquisition. *COLING'04*, Geneva, Switzerland.

Marius Pasca. 2004. Acquisition of Categorized Named Entities for Web Search. *CIKM'04*.

Marius Pasca. 2007. Weakly-Supervised Discovery of Named Entities Using Web Search Queries. *CIKM'07*. pp. 683-690.

Marius Pasca and Benjamin Van Durme. 2008. Weakly-supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs. *ACL'08*.

Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Organizing and Searching the World Wide Web of Facts - Step One: The One-Million Fact Extraction Challenge. In AAAI'06.

Marco Pennacchiotti and Patrick Pantel. 2009. Entity Extraction via Ensemble Semantics. EMNLP'09.

Gerda Ruge. 1992. Experiments on Linguistically-Based Term Associations. *Information Processing & Management*, 28(3): 317-32.

Keiji Shinzato and Kentaro Torisawa. 2005. A Simple WWW-based Method for Semantic Word Class Acquisition. *Recent Advances in Natural Language Processing (RANLP'05)*, Borovets, Bulgaria.

Shuming Shi, Xiaokang Liu, Ji-Rong Wen. 2008. Pattern-based Semantic Class Discovery with Multi-Membership Support. *CIKM'08*, Napa Valley, California, USA.

Hristo Tanev and Bernardo Magnini. 2006. Weakly Supervised Approaches for Ontology Population. EACL'2006, Trento, Italy.

Richard C. Wang and William W. Cohen. 2008. Iterative Set Expansion of Named Entities Using the Web. *ICDM'08*, pages 1091–1096.

Masashi Yamaguchi, Hiroaki Ohshima, Satoshi Oyama, and Katsumi Tanaka. Unsupervised Discovery of Coordinate Terms for Multiple Aspects from Search Engine Query Logs. *The 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*.

Huibin Zhang, Mingjie Zhu, Shuming Shi, and Ji-Rong Wen. 2009. Employing Topic Models for Pattern-based Semantic Class Discovery. *ACL'09*, Singapore.

Metaphor Identification Using Verb and Noun Clustering

Ekaterina Shutova, Lin Sun and Anna Korhonen
Computer Laboratory, University of Cambridge
es407,ls418,alk23@cam.ac.uk

Abstract

We present a novel approach to automatic metaphor identification in unrestricted text. Starting from a small seed set of manually annotated metaphorical expressions, the system is capable of harvesting a large number of metaphors of similar syntactic structure from a corpus. Our method is distinguished from previous work in that it does not employ any hand-crafted knowledge, other than the initial seed set, but, in contrast, captures metaphoricity by means of verb and noun clustering. Being the first to employ unsupervised methods for metaphor identification, our system operates with the precision of 0.79.

1 Introduction

Besides enriching our thought and communication with novel imagery, the phenomenon of metaphor also plays a crucial structural role in our use of language. Metaphors arise when one concept is viewed in terms of the properties of the other. Below are some examples of metaphor.

(1) How can I *kill* a process? (Martin, 1988)

(2) Inflation has *eaten up* all my savings. (Lakoff and Johnson, 1980)

(3) He *shot down* all of my arguments. (Lakoff and Johnson, 1980)

(4) And then my heart with pleasure *fills*,
And *dances* with the daffodils.[1]

In metaphorical expressions seemingly unrelated features of one concept are associated with another concept. In the computer science metaphor in (1) the *computational process* is viewed as something *alive* and, therefore, its forced termination is associated with the act of killing. Lakoff and Johnson (1980) explain metaphor as a systematic association, or a *mapping*, between two concepts or conceptual domains: the *source* and the *target*. The metaphor in (3) exemplifies a mapping of a concept of *argument* to that of *war*. The *argument*, which is the target concept, is viewed in terms of a *battle* (or a *war*), the source concept. The existence of such a link allows us to talk about arguments using the war terminology, thus giving rise to a number of metaphors.

Characteristic to all areas of human activity (from poetic to ordinary to scientific) and, thus, to all types of discourse, metaphor becomes an important problem for natural language processing (NLP). In order to estimate the frequency of the phenomenon, Shutova and Teufel (2010) conducted a corpus study on a subset of the British National Corpus (BNC) (Burnard, 2007) representing various genres. They manually annotated metaphorical expressions in this data and found that 241 out of 761 sentences contained a metaphor, whereby in 164 phrases metaphoricity was introduced by a verb. Due to such a high frequency of their use, a system capable of recognizing and interpreting metaphorical expressions in unrestricted text would become an invaluable component of any semantics-oriented NLP application.

Automatic processing of metaphor can be clearly divided into two subtasks: *metaphor identification* (distinguishing between literal and metaphorical language in text) and *metaphor interpretation* (identifying the intended literal meaning of a metaphorical expression). Both of them have been repeatedly attempted in NLP.

To date the most influential account of metaphor identification is that of Wilks (1978).

[1] "I wandered lonely as a cloud", William Wordsworth, 1804.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1002–1010,
Beijing, August 2010

According to Wilks, metaphors represent a violation of selectional restrictions in a given context. Consider the following example.

(5) My car *drinks* gasoline. (Wilks, 1978)

The verb *drink* normally takes an *animate* subject and a *liquid* object. Therefore, *drink* taking a *car* as a subject is an anomaly, which may as well indicate metaphorical use of *drink*.

This approach was automated by Fass (1991) in his met* system. However, Fass himself indicated a problem with the method: it detects any kind of non-literalness or anomaly in language (metaphors, metonymies and others), i.e., it overgenerates with respect to metaphor. The techniques met* uses to differentiate between those are mainly based on hand-coded knowledge, which implies a number of limitations. In a similar manner manually created knowledge in the form of WordNet (Fellbaum, 1998) is employed by the system of Krishnakumaran and Zhu (2007), which essentially differentiates between highly lexicalized metaphors included in WordNet, and novel metaphorical senses.

Alternative approaches (Gedigan et al., 2006) search for metaphors of a specific domain defined a priori (e.g. MOTION metaphors) in a specific type of discourse (e.g. Wall Street Journal). In contrast, the scope of our experiments is the whole of the British National Corpus (BNC) (Burnard, 2007) and the domain of the expressions we identify is unrestricted. However, our technique is also distinguished from the systems of Fass (1991) and Krishnakumaran and Zhu (2007) in that it does not rely on any hand-crafted knowledge, but rather captures metaphoricity in an unsupervised way by means of verb and noun clustering.

The motivation behind the use of clustering methods for metaphor identification task lies in the nature of metaphorical reasoning based on association. Compare, for example, the target concepts of *marriage* and *political regime*. Having quite distinct meanings, both of them are cognitively mapped to the source domain of *mechanism*, which shows itself in the following examples:

(6) Our relationship is not really *working*.

(7) Diana and Charles did not succeed in *mending* their marriage.

(8) The *wheels* of Stalin's regime were *well oiled* and already *turning*.

We expect that such relatedness of distinct target concepts should manifest itself in the examples of language use, i.e. target concepts that are associated with the same source concept should appear in similar lexico-syntactic environments. Thus, clustering concepts using grammatical relations (GRs) and lexical features would allow us to capture their relatedness **by association** and harvest a large number of metaphorical expressions beyond our seed set. For example, the sentence in (6) being part of the seed set should enable the system to identify metaphors in both (7) and (8).

In summary, our system (1) starts from a seed set of metaphorical expressions exemplifying a range of source–target domain mappings; (2) performs unsupervised noun clustering in order to harvest various target concepts associated with the same source domain; (3) by means of unsupervised verb clustering creates a source domain verb lexicon; (4) searches the BNC for metaphorical expressions describing the target domain concepts using the verbs from the source domain lexicon.

We tested our system starting with a collection of metaphorical expressions representing verb-subject and verb-object constructions, where the verb is used metaphorically. We evaluated the precision of metaphor identification with the help of human judges. In addition to this we compared our system to a baseline built upon WordNet, whereby we demonstrated that our method goes far beyond synonymy and captures metaphors not directly related to any of those seen in the seed set.

2 Experimental Data

2.1 Seed Phrases

We used the dataset of Shutova (2010) as a seed set. Shutova (2010) annotated metaphorical expressions in a subset of the BNC sampling various genres: literature, newspaper/journal articles, essays on politics, international relations and history, radio broadcast (transcribed speech). The dataset consists of 62 phrases that are single-word metaphors representing verb-subject and verb-object relations, where a verb is used metaphorically. The seed phrases include e.g. *stir excitement, reflect enthusiasm, accelerate change, grasp theory, cast doubt, suppress memory, throw remark* (verb - direct object constructions) and *campaign surged, factor shaped [..], tension mounted, ideology embraces, changes operated, approach focuses, example illustrates* (subject - verb constructions).

2.2 Corpus

The search space for metaphor identification was the British National Corpus (BNC) that was parsed using the RASP parser of Briscoe et al. (2006). We used the grammatical relations output of RASP for BNC created by Andersen et al. (2008). The system searched the corpus for the source and target domain vocabulary within a particular grammatical relation (verb-object or verb-subject).

3 Method

Starting from a small seed set of metaphorical expressions, the system implicitly captures the associations that underly their production and comprehension. It generalizes over these associations by means of unsupervised verb and noun clustering. The obtained clusters then represent potential source and target concepts between which metaphorical associations hold. The knowledge of such associations is then used to annotate metaphoricity in a large corpus.

3.1 Clustering Motivation

Abstract concepts that are associated with the same source domain are often related to each other on an intuitive and rather structural level, but their meanings, however, are not necessarily synonymous or even semantically close. The results of previous research on corpus-based lexical semantics suggest that the linguistic environment in which a lexical item occurs can shed light on its meaning. A number of works have shown that it is possible to automatically induce semantic word classes from corpus data via clustering of contextual cues (Pereira et al., 1993; Lin, 1998; Schulte im Walde, 2006). The consensus is that the lexical items exposing similar behavior in a large body of text most likely have the same meaning. However, the concepts of *marriage* and *political regime*, that are also observed in similar lexico-syntactic environments, albeit having quite distinct meanings are likewise assigned by such methods to the same cluster. In contrast to concrete concepts, such as *tea, water, coffee, beer, drink, liquid*, that are clustered together due to **meaning similarity**, abstract concepts tend to be clustered together **by association** with the same source domain. It is the presence of this association that explains the fact that they share common contexts. We exploit this idea for identification of new target domains associated with the same source domain. We then use unsupervised verb clustering to collect source domain vocabulary, which in turn allows us to harvest a large number of new metaphorical expressions.

3.2 Verb and Noun Clustering

Since Levin (1993) published her classification, there have been a number of attempts to automatically classify verbs into semantic classes using supervised and unsupervised approaches (Lin, 1998; Brew and Schulte im Walde, 2002; Korhonen et al., 2003; Schulte im Walde, 2006; Joanis et al., 2008; Sun and Korhonen, 2009). Similar methods were also applied to acquisition of noun classes from corpus data (Rooth et al., 1999; Pantel and Lin, 2002; Bergsma et al., 2008).

We adopt a recent verb clustering approach of Sun and Korhonen (2009), who used rich syntactic and semantic features extracted using a shallow parser and a clustering method suitable for the resulting high dimensional feature space. When Sun and Korhonen evaluated their approach on 204 verbs from 17 Levin classes, they obtained 80.4 F-measure (which is high in particular for an unsupervised approach). We apply this approach to a much larger set of 1610 verbs: all the verb forms appearing in VerbNet (Kipper et al., 2006) with the exception of highly infrequent ones. In addition, we adapt the approach to noun clustering.

3.2.1 Feature Extraction

Our verb dataset is a subset of VerbNet compiled as follows. For all the verbs in VerbNet we extracted their occurrences (up to 10,000) from the raw corpus data collected originally by Korhonen et al. (2006) for construction of VALEX lexicon. Only the verbs found in this data more than 150 times were included in the experiment.

For verb clustering, we adopted the best performing features of Sun and Korhonen (2009): automatically acquired verb subcategorization frames (SCFs) parameterized by their selectional preferences (SPs). We obtained these features using the SCF acquisition system of Preiss et al. (2007). The system tags and parses corpus data using the RASP parser and extracts SCFs from the resulting GRs using a rule-based classifier which identifies 168 SCF types for English verbs. It produces a lexical entry for each verb and SCF combination occurring in corpus data. We obtained SPs by clustering argument heads appearing in the subject and object slots of verbs in the resulting lexicon.

Our noun dataset consists of 2000 most frequent nouns in the BNC. Following previous works on semantic noun classification (Pantel and Lin, 2002; Bergsma et al., 2008), we used GRs as features for noun clustering. We employed all the argument heads and verb lemmas appearing in the subject, direct object and indirect object relations in the RASP-parsed BNC.

The feature vectors were first constructed from the corpus counts, and subsequently normalized by the sum of the feature values before applying clustering.

3.2.2 Clustering Algorithm

We use spectral clustering (SPEC) for both verbs and nouns. This technique has proved to be effective in previous verb clustering works (Brew and Schulte im Walde, 2002; Sun and Korhonen, 2009) and in related NLP tasks involving high dimensional data (Chen et al., 2006). We use the MNCut algorithm for SPEC which has a wide applicability and a clear probabilistic interpretation (Meila and Shi, 2001).

The task is to group a given set of words $W = \{w_n\}_{n=1}^{N}$ into a disjoint partition of K classes. SPEC takes a similarity matrix as input. We construct it using the Jensen-Shannon divergence (JSD) as a measure. The JSD between two feature vectors w and w' is $d_{jsd}(w, w') = \frac{1}{2}D(w||m) + \frac{1}{2}D(w'||m)$ where D is the Kullback-Leibler divergence, and m is the average of the w and w'.

The similarity matrix S is constructed where $S_{ij} = \exp(-d_{jsd}(w, w'))$. In SPEC, the similarities S_{ij} are viewed as weights on the edges ij of a graph G over W. The similarity matrix S is thus the adjacency matrix for G. The degree of a vertex i is $d_i = \sum_{j=1}^{N} S_{ij}$. A cut between two partitions A and A' is defined to be $\text{Cut}(A, A') = \sum_{m \in A, n \in A'} S_{mn}$.

The similarity matrix S is then transformed into a stochastic matrix P.

$$P = D^{-1}S \tag{1}$$

The degree matrix D is a diagonal matrix where $D_{ii} = d_i$.

It was shown by Meila and Shi (2001) that if P has the K leading eigenvectors that are piecewise constants[2] with respect to a partition I^* and their eigenvalues are not zero, then I^* minimizes the multiway normalized cut (MNCut):

$$\text{MNCut}(I) = K - \sum_{k=1}^{K} \frac{\text{Cut}(I_k, I_k)}{\text{Cut}(I_k, I)}$$

P_{mn} can be interpreted as the transition probability between the vertexes m, n. The criterion can thus be expressed as $\text{MNCut}(I) = \sum_{k=1}^{K}(1 - P(I_k \to I_k|I_k))$ (Meila, 2001), which is the sum of transition probabilities across different clusters. This criterion finds the partition where random walks are most likely to happen within the same cluster. In practice, the leading eigenvectors of P are not piecewise constants. However, we can extract the partition by finding the approximately equal elements in the eigenvectors using a clustering algorithm, such as K-Means.

Since SPEC has elements of randomness, we ran the algorithm multiple times and the partition that minimizes the distortion (the distances to cluster centroid) is reported. Some of the clusters obtained as a result of applying the algorithm to our noun and verb datasets are demonstrated in Figures 1 and 2 respectively. The noun clusters represent target concepts that we expect to be associated with the same source concept (some suggested source concepts are given in Figure 1, although the system only captures those implicitly).

[2] An eigenvector v is piecewise constant with respect to I if $v(i) = v(j) \forall i, j \in I_k$ and $k \in 1, 2...K$

Source: MECHANISM
Target Cluster: consensus relation tradition partnership resistance foundation alliance friendship contact reserve unity link peace bond myth identity hierarchy relationship connection balance marriage democracy defense faith empire distinction coalition regime division
Source: STORY; JOURNEY
Target Cluster: politics practice trading reading occupation profession sport pursuit affair career thinking life
Source: LOCATION; CONTAINER
Target Cluster: lifetime quarter period century succession stage generation decade phase interval future
Source: LIVING BEING; END
Target Cluster: defeat fall death tragedy loss collapse decline disaster destruction fate

Figure 1: Clustered target concepts

Source Cluster: sparkle glow widen flash flare gleam darken narrow flicker shine blaze bulge
Source Cluster: gulp drain stir empty pour sip spill swallow drink pollute seep flow drip purify ooze pump bubble splash ripple simmer boil tread
Source Cluster: polish clean scrape scrub soak
Source Cluster: kick hurl push fling throw pull drag haul
Source Cluster: rise fall shrink drop double fluctuate dwindle decline plunge decrease soar tumble surge spiral boom

Figure 2: Clustered verbs (source domains)

The verb clusters contain coherent lists of source domain vocabulary.

3.3 Selectional Preference Strength Filter

Following Wilks (1978), we take metaphor to represent a violation of selectional restrictions. However, not all verbs have an equally strong capacity to constrain their arguments, e.g. *remember, accept, choose* etc. are weak in that respect. We suggest that for this reason not all the verbs would be equally prone to metaphoricity, but only the ones exhibiting strong selectional preferences. We test this hypothesis experimentally and expect that placing this criterion would enable us to filter out a number of candidate expressions, that are less likely to be used metaphorically.

We automatically acquired selectional preference distributions for `Verb-Subject` and `Verb-Object` relations from the BNC parsed by RASP. We first clustered 2000 most frequent nouns in the BNC into 200 clusters using SPEC as described in the previous section. The obtained clusters formed our selectional preference classes. We adopted the selectional preference measure proposed by Resnik (1993) and successfully applied to a number of tasks in NLP including word sense disambiguation (Resnik, 1997). Resnik models selectional preference of a verb in probabilistic terms as the difference between the posterior distribution of noun classes in a particular relation with the verb and their prior distribution in that syntactic position regardless of the identity of the predicate. He quantifies this difference using the relative entropy (or Kullback-Leibler distance), defining the *selectional preference strength* (SPS) as follows.

$$S_R(v) = D(P(c|v)||P(c)) = \sum_c P(c|v) \log \frac{P(c|v)}{P(c)}, \quad (2)$$

where $P(c)$ is the prior probability of the noun class, $P(c|v)$ is the posterior probability of the noun class given the verb and R is the grammatical relation in question. SPS measures how strongly the predicate constrains its arguments.

We use this measure to filter out the verbs with weak selectional preferences. The optimal SPS threshold was set experimentally on a small held-out dataset and approximates to 1.32. We excluded expressions containing the verbs with preference strength below this threshold from the set of candidate metaphors.

4 Evaluation and Discussion

In order to prove that our metaphor identification method generalizes well over the seed set and goes far beyond synonymy, we compared its output to that of a baseline taking WordNet synsets to represent source and target domains. We evaluated the quality of metaphor tagging in terms of precision with the help of human judges.

4.1 Comparison against WordNet Baseline

The baseline system was implemented using synonymy information from WordNet to expand on the seed set. Assuming all the synonyms of the verbs and nouns in seed expressions to represent the source and target vocabularies respectively, the system searches for phrases composed of lexical items belonging to those vocabularies. For example, given a seed expression *stir excitement*, the baseline finds phrases such as *arouse fervour,* *stimulate agitation, stir turmoil* etc. However, it is not able to generalize over the concepts to broad semantic classes, e.g. it does not find other *feelings* such as *rage, fear, anger, pleasure* etc., which is necessary to fully characterize the target domain. The same deficiency of the baseline system manifests itself in the source domain vocabulary: the system has only the knowledge of direct synonyms of *stir*, as opposed to other verbs characteristic to the domain of *liquids*, e.g. *pour, flow, boil* etc., successfully identified by means of clustering.

To compare the **coverage** achieved by unsupervised clustering to that of the baseline in quantitative terms, we estimated the number of WordNet synsets, i.d. different word senses, in the metaphorical expressions captured by the two systems. We found that the baseline system covers only 13% of the data identified using clustering and does not go beyond the concepts present in the seed set. In contrast, most metaphors tagged by our method are novel and represent a considerably wider range of meanings, e.g. given the seed metaphors *stir excitement, throw remark, cast doubt* the system identifies previously unseen expressions *swallow anger, hurl comment, spark enthusiasm* etc. as metaphorical.

4.2 Comparison with Human Judgements

In order to access the **quality** of metaphor identification by both systems we used the help of human annotators. The annotators were presented with a set of randomly sampled sentences containing metaphorical expressions as annotated by the system and by the baseline. They were asked to mark the tagged expressions that were metaphorical in their judgement as correct.

The annotators were encouraged to rely on their own intuition of metaphor. However, we also provided some guidance in the form of the following definition of metaphor[3]:

1. For each verb establish its meaning in context and try to imagine a more basic meaning of this verb on other contexts. Basic meanings normally are: (1) more concrete; (2) related to bodily action; (3) more precise (as opposed to vague); (4) historically older.
2. If you can establish the basic meaning that is distinct from the meaning of the verb in this context, the verb is likely to be used metaphorically.

CKM 391 Time and time again he would stare at the ground, hand on hip, if he thought he had received a bad call, and then **swallow his anger** and play tennis.
AD9 3205 He tried to **disguise the anxiety** he felt when he found the comms system down, but Tammuz was nearly hysterical by this stage.
AMA 349 We will **halt the reduction** in NHS services for long-term care and community health services which support elderly and disabled patients at home.
ADK 634 **Catch their interest** and **spark their enthusiasm** so that they begin to see the product's potential.
K2W 1771 The committee heard today that gangs regularly **hurled** abusive **comments** at local people, making an unacceptable level of noise and leaving litter behind them.

Figure 3: Sentences tagged by the system (metaphors in bold)

We had 5 volunteer annotators who were all native speakers of English and had no or sparse linguistic knowledge. Their agreement on the task was 0.63 in terms of κ (Siegel and Castellan, 1988), whereby the main source of disagreement was the presence of highly lexicalized metaphors, e.g. verbs such as *adopt, convey, decline* etc. We then evaluated the system performance against their judgements in terms of *precision*. Precision measures the proportion of metaphorical expressions that were tagged correctly among the ones that were tagged. We considered the expressions tagged as metaphorical by at least three annotators to be correct. As a result our system identifies metaphor with the precision of 0.79, whereas the baseline only attains 0.44. Some examples of sentences annotated by the system are shown in Figure 3.

Such a striking discrepancy between the performance levels of the clustering approach and the baseline can be explained by the fact that a large number of metaphorical senses are included in WordNet. This means that in WordNet synsets source domain verbs are mixed with more abstract terms. For example, the metaphorical sense of *shape* in *shape opinion* is part of the synset (*de-*

[3] taken from the annotation procedure of Shutova and Teufel (2010) that is in turn partly based on the work of Pragglejaz Group (2007).

termine, shape, mold, influence, regulate). This results in the baseline system tagging literal expressions as metaphorical, erroneously assuming that the verbs from the synset belong to the source domain.

The main source of confusion in the output of our clustering method was the conventionality of some metaphorical expressions, e.g. *hold views, adopt traditions, tackle a problem*. The system is capable of tracing metaphorical etymology of conventional phrases, but their senses are highly lexicalized. This lexicalization is reflected in the data and affects clustering in that conventional metaphors are sometimes clustered together with literally used terms, e.g. *tackle a problem* and *resolve a problem*, which may suggest that the latter are metaphorical. It should be noted, however, that such errors are rare.

Since there is no large metaphor-annotated corpus available, it was impossible for us to reliably evaluate the *recall* of the system. However, the system identified a total number of 4456 metaphorical expressions in the BNC starting with a seed set of only 62, which is a promising result.

5 Related Work

One of the first attempts to identify and interpret metaphorical expressions in text automatically is the approach of Fass (1991). Fass developed a system called met*, capable of discriminating between literalness, metonymy, metaphor and anomaly. It does this in three stages. First, literalness is distinguished from non-literalness using selectional preference violation as an indicator. In the case that non-literalness is detected, the respective phrase is tested for being a metonymic relation using hand-coded patterns (such as CONTAINER-for-CONTENT). If the system fails to recognize metonymy, it proceeds to search the knowledge base for a *relevant analogy* in order to discriminate metaphorical relations from anomalous ones. E.g., the sentence in (5) would be represented in this framework as (*car,drink,gasoline*), which does not satisfy the preference (*animal,drink,liquid*), as *car* is not a hyponym of *animal*. met* then searches its knowledge base for a triple containing a hypernym of both the actual argument and the desired argument and finds (*thing,use,energy_source*), which represents the metaphorical interpretation.

Birke and Sarkar (2006) present a sentence clustering approach for non-literal language recognition implemented in the TroFi system (Trope Finder). This idea originates from a similarity-based word sense disambiguation method developed by Karov and Edelman (1998). The method employs a set of seed sentences, where the senses are annotated, computes similarity between the sentence containing the word to be disambiguated and all of the seed sentences and selects the sense corresponding to the annotation in the most similar seed sentences. Birke and Sarkar (2006) adapt this algorithm to perform a two-way classification: literal vs. non-literal, and they do not clearly define the kinds of tropes they aim to discover. They attain a performance of 53.8% in terms of f-score.

The method of Gedigan et al. (2006) discriminates between literal and metaphorical use. They trained a maximum entropy classifier for this purpose. They obtained their data by extracting the lexical items whose frames are related to MOTION and CURE from FrameNet (Fillmore et al., 2003). Then they searched the PropBank Wall Street Journal corpus (Kingsbury and Palmer, 2002) for sentences containing such lexical items and annotated them with respect to metaphoricity. They used PropBank annotation (arguments and their semantic types) as features to train the classifier and report an accuracy of 95.12%. This result is, however, only a little higher than the performance of the naive baseline assigning majority class to all instances (92.90%). These numbers can be explained by the fact that 92.00% of the verbs of MOTION and CURE in the Wall Street Journal corpus are used metaphorically, thus making the dataset unbalanced with respect to the target categories and the task notably easier.

Both Birke and Sarkar (2006) and Gedigan et al. (2006) focus only on metaphors expressed by a verb. As opposed to that the approach of Krishnakumaran and Zhu (2007) deals with verbs, nouns and adjectives as parts of speech. They use hyponymy relation in WordNet and word bigram counts to predict metaphors at the sentence level. Given an IS-A metaphor (e.g. *The world is a stage*[4]) they verify if the two nouns involved are in hyponymy relation in WordNet, and if this is not the case then this sentence is tagged as containing a metaphor. Along with this they consider expressions containing a verb or an adjective used metaphorically (e.g. *He planted good ideas in their minds* or *He has a fertile imagination*). Hereby they calculate bigram probabilities of verb-noun and adjective-noun pairs (including the hyponyms/hypernyms of the noun in question). If the combination is not observed in the data with sufficient frequency, the system tags the sentence containing it as metaphorical. This idea is a modification of the selectional preference view of Wilks. However, by using bigram counts over verb-noun pairs as opposed to verb-object relations extracted from parsed text Krishnakumaran and Zhu (2007) loose a great deal of information. The authors evaluated their system on a set of example sentences compiled from the Master Metaphor List (Lakoff et al., 1991), whereby highly conventionalized metaphors (they call them *dead metaphors*) are taken to be negative examples. Thus, they do not deal with literal examples as such: essentially, the distinction they are making is between the senses included in WordNet, even if they are conventional metaphors, and those not included in WordNet.

6 Conclusions and Future Directions

We presented a novel approach to metaphor identification in unrestricted text using unsupervised methods. Starting from a limited set of metaphorical seeds, the system is capable of capturing the regularities behind their production and annotating a much greater number and wider range of previously unseen metaphors in the BNC.

Our system is the first of its kind and it is capable of identifying metaphorical expressions with a high precision (0.79). By comparing its coverage to that of a WordNet baseline, we proved that our method goes far beyond synonymy and generalizes well over the source and target domains. Although at this stage we tested our system on verb-subject and verb-object metaphors only, we are convinced that the described identification techniques can be similarly applied to a wider range of syntactic constructions. Extending the system to deal with more parts of speech and types of phrases is part of our future work.

One possible limitation of our approach is that it is seed-dependent, which makes the recall of the system questionable. Thus, another important future research avenue is the creation of a more diverse seed set. We expect that a set of expressions representative of the whole variety of common metaphorical mappings, already described in linguistics literature, would enable the system to attain a very broad coverage of the corpus. Master Metaphor List (Lakoff et al., 1991) and other existing metaphor resources could be a sensible starting point on a route to such a dataset.

Acknowledgments

We are very grateful to our anonymous reviewers for their useful feedback on this work and the volunteer annotators for their interest, time and help. This research is funded by generosity of Cambridge Overseas Trust (Katia Shutova), Dorothy Hodgkin Postgraduate Award (Lin Sun) and the Royal Society, UK (Anna Korhonen).

References

Andersen, O. E., J. Nioche, E. Briscoe, and J. Carroll. 2008. The BNC parsed with RASP4UIMA. In *Proceedings of LREC 2008*, Marrakech, Morocco.

Bergsma, S., D. Lin, and R. Goebel. 2008. Discriminative learning of selectional preference from unlabeled text. In *Proceedings of the EMNLP*.

Birke, J. and A. Sarkar. 2006. A clustering approach for the nearly unsupervised recognition of nonliteral language. In *In Proceedings of EACL-06*, pages 329–336.

Brew, C. and S. Schulte im Walde. 2002. Spectral clustering for German verbs. In *Proceedings of EMNLP*.

Briscoe, E., J. Carroll, and R. Watson. 2006. The second release of the rasp system. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 77–80.

Burnard, L. 2007. *Reference Guide for the British National Corpus (XML Edition).*

[4] William Shakespeare

Chen, J., D. Ji, C. Lim Tan, and Z. Niu. 2006. Unsupervised relation disambiguation using spectral clustering. In *Proceedings of COLING/ACL*.

Fass, D. 1991. met*: A method for discriminating metonymy and metaphor by computer. *Computational Linguistics*, 17(1):49–90.

Fellbaum, C., editor. 1998. *WordNet: An Electronic Lexical Database (ISBN: 0-262-06197-X)*. MIT Press, first edition.

Fillmore, C. J., C. R. Johnson, and M. R. L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16(3):235–250.

Gedigan, M., J. Bryant, S. Narayanan, and B. Ciric. 2006. Catching metaphors. In *In Proceedings of the 3rd Workshop on Scalable Natural Language Understanding*, pages 41–48, New York.

Joanis, E., S. Stevenson, and D. James. 2008. A general feature space for automatic verb classification. *Natural Language Engineering*, 14(3):337–367.

Karov, Y. and S. Edelman. 1998. Similarity-based word sense disambiguation. *Computational Linguistics*, 24(1):41–59.

Kingsbury, P. and M. Palmer. 2002. From TreeBank to PropBank. In *Proceedings of LREC-2002*, Gran Canaria, Canary Islands, Spain.

Kipper, K., A. Korhonen, N. Ryant, and M. Palmer. 2006. Extensive classifications of English verbs. In *Proceedings of the 12th EURALEX International Congress*.

Korhonen, A., Y. Krymolowski, and Z. Marx. 2003. Clustering polysemic subcategorization frame distributions semantically. In *Proceedings of ACL 2003*, Sapporo,Japan.

Korhonen, A., Y. Krymolowski, and T. Briscoe. 2006. A large subcategorization lexicon for natural language processing applications. In *Proceedings of LREC 2006*.

Krishnakumaran, S. and X. Zhu. 2007. Hunting elusive metaphors using lexical resources. In *Proceedings of the Workshop on Computational Approaches to Figurative Language*, pages 13–20, Rochester, NY.

Lakoff, G. and M. Johnson. 1980. *Metaphors We Live By*. University of Chicago Press, Chicago.

Lakoff, G., J. Espenson, and A. Schwartz. 1991. The master metaphor list. Technical report, University of California at Berkeley.

Levin, B. 1993. *English Verb Classes and Alternations*. University of Chicago Press, Chicago.

Lin, D. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774.

Martin, J. H. 1988. Representing regularities in the metaphoric lexicon. In *Proceedings of the 12th conference on Computational linguistics*, pages 396–401.

Meila, M. and J. Shi. 2001. A random walks view of spectral segmentation. In *AISTATS*.

Meila, M. 2001. The multicut lemma. Technical report, University of Washington.

Pantel, P. and D. Lin. 2002. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619. ACM.

Pereira, F., N. Tishby, and L. Lee. 1993. Distributional clustering of English words. In *Proceedings of ACL-93*, pages 183–190, Morristown, NJ, USA.

Pragglejaz Group. 2007. MIP: A method for identifying metaphorically used words in discourse. *Metaphor and Symbol*, 22:1–39.

Preiss, J., T. Briscoe, and A. Korhonen. 2007. A system for large-scale acquisition of verbal, nominal and adjectival subcategorization frames from corpora. In *Proceedings of ACL-2007*, volume 45, page 912.

Resnik, P. 1993. *Selection and Information: A Class-based Approach to Lexical Relationships*. Ph.D. thesis, Philadelphia, PA, USA.

Resnik, P. 1997. Selectional preference and sense disambiguation. In *ACL SIGLEX Workshop on Tagging Text with Lexical Semantics*, Washington, D.C.

Rooth, M., S. Riezler, D. Prescher, G. Carroll, and F. Beil. 1999. Inducing a semantically annotated lexicon via EM-based clustering. In *Proceedings of ACL 99*, pages 104–111.

Schulte im Walde, S. 2006. Experiments on the automatic induction of German semantic verb classes. *Computational Linguistics*, 32(2):159–194.

Shutova, E. and S. Teufel. 2010. Metaphor corpus annotated for source - target domain mappings. In *Proceedings of LREC 2010*, Malta.

Shutova, E. 2010. Automatic metaphor interpretation as a paraphrasing task. In *Proceedings of NAACL 2010*, Los Angeles, USA.

Siegel, S. and N. J. Castellan. 1988. *Nonparametric statistics for the behavioral sciences*. McGraw-Hill Book Company, New York, USA.

Explore the Structure of Social Tags by Subsumption Relations

Xiance Si, Zhiyuan Liu, Maosong Sun
Department of Computer Science and Technology
State Key Lab on Intelligent Technology and Systems
National Lab for Information Science and Technology
Tsinghua University
{sixiance,lzy.thu}@gmail.com, sms@tsinghua.edu.cn

Abstract

Thanks to its simplicity, social tagging system has accumulated huge amount of user contributed tags. However, user contributed tags lack explicit hierarchical structure, while many tag-based applications would benefit if such a structure presents. In this work, we explore the structure of tags with a directed and easy-to-evaluate relation, named as the subsumption relation. We propose three methods to discover the subsumption relation between tags. Specifically, the tagged document's content is used to find the relations, which leads to better result. Besides relation discovery, we also propose a greedy algorithm to eliminate the redundant relations by constructing a Layered Directed Acyclic Graph (Layered-DAG) of tags. We perform quantitative evaluations on two real world data sets. The results show that our methods outperform hierarchical clustering-based approach. Empirical study of the constructed Layered-DAG and error analysis are also provided.

1 Introduction

In this work, we aim at exploring the structure of social tags. Social tagging is widely used in Web-based services, in which a user could use any word to annotate an object. Thanks to its simplicity, services with social tagging features have attracted a lot of users and have accumulated huge amount of annotations. However, comparing to taxonomies, social tagging has an inherent shortcoming, that there is no explicit hierarchical relations between tags. Figure 1 (a) shows an example of the commonly used flat tag cloud, in which only the popularity of a tag is concerned. Kome et al. (2005) argued that implicit hierarchical relations exist in social tags. Previous literature shows that organizing tags in hierarchical structures will help tag-based Information Retrieval applications (Begelman et al., 2006; Brooks and Montanez, 2006).

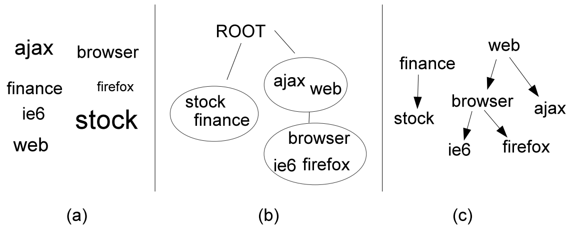


Figure 1: Examples of (a) flat tag cloud, (b) hierarchical clusters, and (c) subsumption relations.

Hierarchical clustering could reveal the similarity relations of tags. Figure 1 (b) shows an example of a typical hierarchical clustering of tags. While clusters can capture similarity between tags, problems still remain: First, clusters mix different relations, such as synonyms and hypernyms. Second, clusters also ignore the direction of relations, for example, the direction in browser → firefox. Third, it is hard to evaluate the correctness of clustering. Specifically, it is hard to tell if two tags are similar or not. In practice, directed and easy-to-evaluate relations between tags are preferred, such as Figure 1 (c).

In this work, we explore the structure of social tags by discovering a directed and easy-to-evaluate relation between tags, named *subsumption relation*. A tag t_a subsumes t_b, if and only if wherever t_b is used, we can also replace it

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1011–1019,
Beijing, August 2010

with t_a. Unlike *similar-to*, subsumption relation is asymmetric, and its correctness is easier to assess. Then, we propose three ways to discover the subsumption relations, through tag-tag, tag-word and tag-reason co-occurrences respectively. In the third way, A tag's *reason* is defined as the word in the content that explains the using of the tag. We employ the Tag Allocation Model (TAM) proposed by Si et al. (2010) to find the reason for each tag. Besides subsumption relation discovery, we also propose a greedy algorithm to remove the redundant relations. The removal is done by constructing a Layered Directed Acyclic Graph (Layered-DAG) of tags with the subsumption relations.

We carried out the experiments on two real world data sets. The results of quantitative evaluation showed that tag-reason based approach outperformed other two methods and a commonly used hierarchical clustering-based method. We also do empirical study on the output of Layered-DAG construction.

The contribution of this paper can be summarized as follows:

1. We explore the structure of social tags by a clearly defined subsumption relation. We propose methods to discover the subsumption relation automatically, leveraging both the co-occurred tags and the content of annotated document.
2. We propose an algorithm to eliminate the redundant relations by constructing a Layered-DAG of tags.
3. We perform both empirical and quantitative evaluation of proposed methods on two real world data sets.

The rest of the paper is organized as follows: Section 2 surveys the related work; Section 3 defines the subsumption relation we used, and proposes methods for relation discovery; Section 4 proposes a greedy algorithm for Layered-DAG construction; Section 5 explains the experimental settings and shows the evaluation results. Section 6 concludes the paper.

2 Related Work

To explore the hierarchical relations between tags, an intuitive way is to cluster the tags into hierarchical clusters. Wu et al. (2006b) used a factorized model, namely Latent Semantic Analysis, to group tags into non-hierarchical topics for better recommendation. Brooks et al. (2006) argued that performing Hierarchical Agglomerative Clustering (HAC) on tags can improve the collaborative tagging system. Later, HAC on tags was also used for improving personalized recommendation (Shepitsen et al., 2008). Heymann et al. (2006) clustered tags into a tree by a similarity-based greedy tree-growing method. They evaluated the obtained trees empirically, and reported that the method is simple yet powerful for organizing tags with hierarchies. Based on Heymann et al.'s work, Schwarzkopf et al. (2007) proposed an approach for modeling users with the hierarchy of tags. Begelman et al. (2006) used top-down hierarchical clustering, instead of bottom-up HAC, to organize tags, and argued that tag hierarchies improve user experiences in their system. Most of the hierarchical clustering algorithms rely on the symmetric similarity between tags, while the discovered relations are hard to evaluate quantitatively, since one cannot distinguish similar from not-similar with a clear boundary.

People have also worked on bridging social tagging systems and ontologies. An ontology defines relations between entities. Peter Mika (2005) proposed an extended scheme of social tagging that includes actors, concepts and objects, and used tag co-occurrences to construct an ontology from social tags. Wu et al. (2006a) used hierarchical clustering to build ontology from tags that also use similar-to relations. Later, ontology schemes that fits social tagging system were proposed, such as (Van Damme et al., 2007) and (Echarte et al., 2007), which mainly focused on the relation between tags, objects and users, rather than between tags themselves. Alexandre Passant (2007) mapped tags to domain ontologies manually to improve information retrieval in social media. To construct tag ontology automatically, Angeletou et al. (2007) used ontologies built by domain experts to find relations between tags, but observed a very low coverage. Specia et al. (2007) proposed an integrated framework for organizing tags by existing ontologies, but no experiment was performed. Kim et al. (2008) summarized the state-of-the-art methods to model tags with semantic annotations.

Before social tagging was invented, Sanderson et al. (1999) proposed to use *subsumption* relation to organize words in text hierarchically. Schmitz et al. (2006) followed the idea to use subsumption relation for organizing Flickr [1] tag, where tag-tag co-occurrences are used for discover the relations. We follow the idea of subsumption relation in this paper, and explore alternative ways for relation discovery.

3 Subsumption Relations in Tags

In this section, we define the subsumption relation used in our study, and propose three methods to discover the subsumption relations.

3.1 Definitions

First, we introduce the symbols used through out the paper: A tag is denoted as $t \in T$, where T is the set of all tags. To distinguish from words, we use `fixed-width` to represent the example tags. An annotated document is denoted as $d \in D$, where D is the set of all documents. The words in d are denoted as a set $\{w_{d_i}\}$, where $i \in [1, |d|]$, and $|d|$ is the number of words in d.

Inspired by (Sanderson and Croft, 1999), we define the subsumption relation between t_a and t_b as follows: t_a **subsumes** t_b**, means that wherever the tag** t_b **is used,** t_a **can also be used without ambiguity**. The subsumption relation between t_a and t_b is denoted as $t_a \rightarrow_s t_b$.

Subsumption relation is directional, that is, $t_a \rightarrow_s t_b$ does not imply $t_b \rightarrow_s t_a$. For example, `literature` $\rightarrow_s$ `chineseliterature`, since for any document annotated with `chineseliterature`, we can also annotate it with `literature`. However, if we swapped the two tags, the statement would not hold.

Subsumption relation is more strict than similarity. For example, during the time of Haiti earthquake, the tag `earthquake` is close to `haiti` in similarity, but none of them implies the use of the other one: document annotated with `earthquake` may refer to the earthquake in China, while document annotated with `haiti` may mean the traveling experience in Haiti.

Note that the subsumption has transitivity property, that $t_a \rightarrow_s t_b$ and $t_b \rightarrow_s t_c$ means $t_a \rightarrow_s t_c$, which corresponds to our intuition. For instance, `naturaldisaster` $\rightarrow_s$ `earthquake` and `disaster`$\rightarrow_s$`naturaldisaster` means `disaster` $\rightarrow_s$`earthquake`.

3.2 Discover Subsumption Relation

We discover the subsumption relations by estimating the probability $p(t_a|t_b)$. The motivation is, if $t_a \rightarrow_s t_b$ and t_b is used, it would be more likely to see t_a. So, by sorting all (t_a, t_b) pairs by $p(t_a|t_b)$ in descending order, top-ranked pairs are more likely to have subsumption relations.

In this work, we present three methods to estimate the probability $p(t_a|t_b)$, using tag-tag, tag-word and tag-reason co-occurrences respectively. By using tag-word and tag-reason co-occurrences, we leverage the content of the annotated document for subsumption relation discovery.

3.2.1 Tag-Tag Co-occurrences Approach

The most intuitive way to estimate $p(t_a|t_b)$ is via tag-tag co-occurrences. Specifically, we use the following formula:

$$p(t_a|t_b) = \frac{N_d(t_a, t_b)}{N_d(t_b)}, \qquad (1)$$

where $N_d(t_a, t_b)$ is the number of documents that are annotated by both t_a and t_b, and $N_d(t_b)$ is the number of documents annotated by t_b. We denote the tag-tag co-occurrences approach as TAG-TAG.

The use of TAG-TAG can be found in previous literature for organizing tags for photos(Schmitz, 2006). One of TAG-TAG's benefits is that it does not rely on the content of the annotated document, thus it can be applied to tags for non-text objects, such as images and music. However, when coming to text documents, this benefit is also a shortcoming, that TAG-TAG makes no use of the content when it is available.

Using TAG-TAG for subsumption relation discovery relies on an implication, that if a user has annotated d with t_b, he would also annotate all tags that subsumes t_b. The implication may not always hold in real world situations. For example,

[1] http://www.flickr.com. An image sharing site that allows users to annotate images with tags

a novel reader would use tags such as scifi and mystery to organize his collections, but he is not likely to annotate each of his collection as novel or book, since they are too obvious for him. We name the problem as the *omitted-tag problem*.

3.2.2 Tag-Word Co-occurrences Approach

When the content of the annotated document is available, using it for estimating $p(t_a|t_b)$ is a natural thought. The content is expected to be complete and information-rich whether or not the user has omitted any tags. We use the following formula to estimate $p(t_a|t_b)$ by tag-word co-occurrences:

$$\begin{aligned} p(t_a|t_b) &= \sum_{w \in W} p(t_a|w)p(w|t_b) \\ &= \sum_{w \in W} \frac{N_d(t_a, w)}{N_d(w)} \frac{N_d(t_b, w)}{N_d(t_b)}, \end{aligned} \quad (2)$$

where $N_d(t_a, w)$ is the number of documents that contains both tag t_a and word w, and $N_d(w)$ is the number of documents that contains the word w. We denote this approach as TAG-WORD.

Instead of computing tag-tag co-occurrences directly, TAG-WORD uses words in the document as a bridge to estimate $p(t_a|t_b)$. By introducing words, the estimation is less affected by the omitted-tag problem, Take the novel reader example again: Although he does not use the tag novel too often, the words in book descriptions would suggest the using of novel, according to all other documents annotated by novel.

While using the content may weaken the omitted-tag problem, it also brings the noise in text to the estimation. Not every word in the content is related to one of the tags. To the opposite, most words are functional words or that about other aspects of the document. $p(t_a|t_b)$ estimated by using all words may largely depends on these irrelevant words.

3.2.3 Tag-Reason Co-occurrences Approach

To focus on the words that are highly relevant to the interested tags, we propose the third method that uses tag-reason co-occurrences. The *reason* is defined as the word(s) that can explain the using of a tag in the document. For example, the tag scifi for a book could be explained by the words "robot", "Asimov" in the book description. If the reason of each tag could be identified, the noise in content-based $p(t_a|t_b)$ could be reduced.

Si et al. (2010) proposed a probabilistic model for content-based social tags, named Tag Allocation Model (TAM). TAM introduces a latent variable r for each tag in the data set, known as the reason variable. The value of r can be a word in the corresponding document, or a global noise variable μ. Allowing the reason of tags to be a global noise makes TAM deal with content-irrelevant tags and mistakenly annotated tags effectively. The likelihood that a document d is annotated by tag t is given as:

$$\begin{aligned} p(t|d) &= \sum_{w \in d} p(t|r = w)p(r = w|d)p(s = 0) \\ &+ p(t|\mu)p(r = \mu)p(s = 1), \end{aligned} \quad (3)$$

where r is the reason of the tag t, $r \in \{w_{di}|i \in [0, |d|]\} \cup \{\mu\}$, μ is the global noise variable. s is the source of reason t, $s = 0$ means the source is the content of the document, while $s = 1$ means the source is the global noise variable μ. TAM can be trained use Gibbs sampling method. For the details of TAM, please refer to (Si and Sun, 2010).

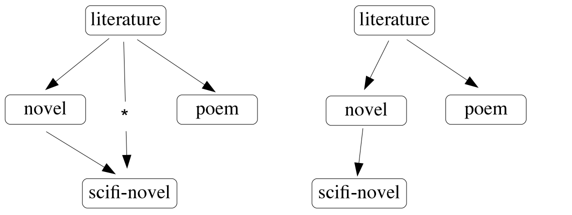
With a trained TAM, we can infer $p(t|r)$, the probability of seeing a tag t when using r as the reason, and $p(r|t)$, the probability of choosing r as the reason for tag t. With these probabilities, we can estimate $p(t_a|t_b)$ by

$$p(t_a|t_b) = \sum_{r \in W} p(t_a|r)p(r|t_b). \quad (4)$$

Note that we use only word reasons ($r \in W$), ignoring the noise reason μ completely. We denote this approach as TAG-REASON.

With the help of TAM, TAG-REASON covers the problems of the TAG-WORD method in two aspects: First, instead of using all words, TAG-REASON emphasizes on the really relevant words, which are the reasons identified by TAM. Second, by ignoring the noise variable μ, TAG-REASON is less affected by the content-irrelevant noise tags, such as thingstodo or myown.

After $p(t_a|t_b)$ is estimated for each $(t_a, t_b) \in T \times T$, we use the top-$n$ pairs with largest $p(t_a|t_b)$



(a) Directed Acyclic Graph (b) Layered Directed Acyclic Graph

Figure 2: DAG and Layered-DAG

as the final set of discovered subsumption relations.

4 Remove Redundancy with Layered-DAG Construction

The discovered subsumption relations connect all tags into a directed graph $G = \{V, E\}$, where V is the set of nodes, with each node is a tag; E is the set of edges, an edge e_{t_a,t_b} from t_a to t_b means $t_a \rightarrow_s t_b$. Furthermore, we define the weight of each edge w_e as the probability $p(t_a|t_b)$.

Recalling that subsumption relation has transitivity property, to avoid the cyclic references in G, we would like to turn G into a Directed Acyclic Graph (DAG). Further, DAG may also contains redundant information. Figure 2 (a) shows a part of a DAG. Note the edge marked as "*", which is perfectly correct, but does not provide extra information, since `literature` $\rightarrow_s$ `novel` and `novel` $\rightarrow_s$ `scifi-novel` have already implied that `literature` $\rightarrow_s$ `novel`. We would like to remove these redundant relations, turning a DAG into the form of Figure 2 (b).

We define Layered-DAG formally as follows: For a DAG G, when given any pair of nodes, if every path that can connect them has equal length, G is a Layered-DAG. Layered-DAG prohibits edges that link cross layers, such like edge "*" in Figure 2 (a). Constructing a Layered-DAG from the discovered relations can eliminate the redundant information.

Given a set of subsumption relations, multiple Layered-DAGs may be constructed. In particular, we want to find the Layered-DAG that maximizes the sum of all edges' weights. Weight maximization implies two concerns: First, when we need to remove a relation to resolve the conflicts or redundancy, the one with lower weight is prefered. Second, when more than one valid Layered-DAGs are available, we want to use the one that contains as many edges as possible.

```
Layered-DAG Construction Algorithm
Input: A set of weighted relations, $R = \{t_a \rightarrow_s t_b | t_a \in T, t_b \in T\}$,
$w_{t_a \rightarrow_s t_b} > 0$
Output: A Layered-DAG of tags $G^* = \{V^*, E^*\}$
 1:  $V^* = \{\}$
 2:  while $R \neq \varnothing$
 3:    if $V^* = \varnothing$
 4:      choose $t_a \rightarrow_s t_b \in R$ with highest weight.
 5:      $E^* \Leftarrow t_a \rightarrow_s t_b$
 6:      $V^* \Leftarrow t_a, V^* \leftarrow t_b$.
 7:      remove $t_a \rightarrow_s t_b$ from $R$.
 8:    else
 9:      $C \Leftarrow \{t_a \rightarrow_s t_b | t_a \rightarrow_s t_b \in R, \{t_a, t_b\} \cap V^* \neq \varnothing\}$
10:      for $t_a \rightarrow_s t_b \in C$ in descending weight order
11:        if adding $t_a \rightarrow_s t_b$ to $G^*$ keeps $G^*$ a Layered-DAG.
12:          $E^* \Leftarrow t_a \rightarrow_s t_b$
13:          $V^* \Leftarrow t_a, V^* \leftarrow t_b$.
14:          break
15:        endif
16:        remove $t_a \rightarrow_s t_b$ from $R$.
17:      endfor
18:    endif
19:  endwhile
20:  output $G^*$
```

Figure 3: A greedy algorithm for constructing Layered-DAG of tags

Finding and proving an optimal algorithm for maximum Layered-DAG construction are beyond the scope of this paper. Here we present a greedy algorithm that works well in practice, as described in Figure 3.

The proposed algorithm starts with a minimal Layered-DAG G^* that contains only the highest weighted relation in R (Steps 1-8). Then, it moves an edge in G to G^* once a time, ensuring that adding the new edge still keeps G^* a valid Layered-DAG (Step 11), and the new edge has the highest weights among all valid candidates (Steps 9-10).

5 Experiments

In this section, we show the experimental results of proposed methods. Specifically, we focus on the following points:

- The quality of discovered subsumption relations by different methods.
- The characteristics of wrong subsumption relations discovered.
- The effect of Layered-DAG construction on the quality of relations.
- Empirical study of the resulted Layered-DAG.

Name	N	$\bar{N}_{tag}$	$\bar{N}_{content}$
BLOG	100,192	2.78	332.87
BOOK	110,371	8.51	204.76

Table 1: Statistics of the data sets. N is the number of documents. $\bar{N}_{tag}$ is the mean number of tags per document. $\bar{N}_{content}$ is the mean number of words per document.

5.1 Data Sets

We use two real world social tagging data sets. The first data set, named BLOG, is a collection of blog posts annotated by blog authors, which is crawled from the web. The second data set, named BOOK, is from a book collecting and sharing site[2], which contains description of Chinese books and user contributed tags. Table 1 lists the basic statistics of the data sets.

The two data sets have different characteristics. Documents in BLOG are longer, not well written, and the number of tags per document is small. To the opposite, documents in BOOK are shorter but well written, and there are more tags for each document.

5.2 Discovered Subsumption Relations

5.2.1 Experimental Settings

For BLOG, we use the tags that have been used more than 10 times; For BOOK, we use the tags that have been used more than 50 times. We perform 100 iterations of Gibbs sampling when training the TAM model, with first 50 iterations as the burn-in iterations. All the estimation methods require proper smoothing. Here we use additive smoothing for all methods, which adds a very small number (0.001 in our case) to all raw counts. Sophisticated smoothing method could be employed, but is out of the scope of this paper.

5.2.2 Evaluation

We use *precision* and *coverage* to evaluate the discovered relations at any given cut-off threshold n. First, we sort the discovered relations by their weights in descending order. Then, we take the top-n relations, discarding the others. For the remaining relations, precision is computed as N_c/n, N_c is the number of correct relations in the top-n list; coverage is computed as $N_t/|T|$, where N_t is the number of unique tags appeared in the top-n list, and $|T|$ is the total number of tags.

To get N_c, the number of correct relations, we need a standard judgement of the correctness of relations, which involves human labeling. To minimize the bias in human assessment, we use **pooling**, which is a widely accepted method in Information Retrieval research (Voorhees and Harman, 2005). Pooling works as follows: First, relations obtained by different methods are mixed together, creating a pool of relations. Second, the pool is shuffled, so that the labeler cannot identify the source of a single relation. Third, annotators are requested to label the relations in the pool as correct or incorrect, based on the definition of subsumption relation. After all relations in the pool are labeled, we use them as the standard judgement to evaluate each method's output.

Precision measures the proportion of correct relations, while coverage measures the proportion of tags that are connected by the relations. The cutoff threshold n affects both precision and coverage: the larger the n, the lower the precision, and the higher the coverage.

5.2.3 Baseline methods

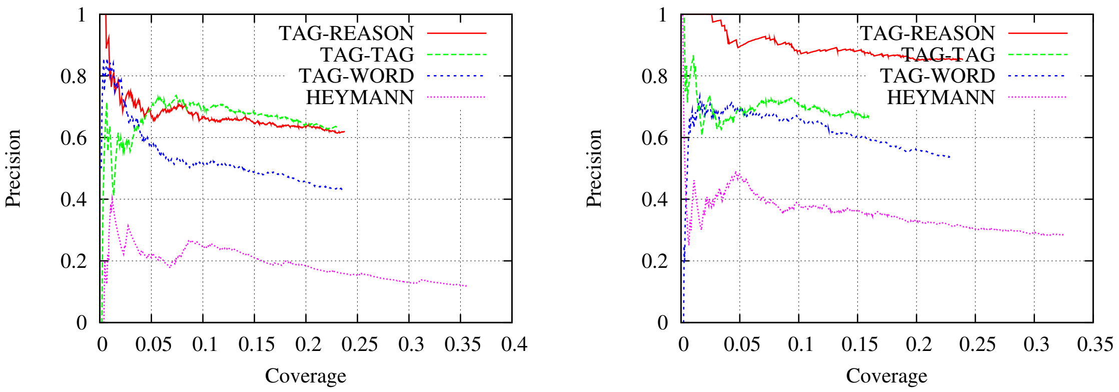
Besides TAG-TAG, TAG-WORD and TAG-REASON, we also include the method described in (Heymann and Garcia-Molina, 2006) as a baseline, denoted as HEYMANN. HEYMANN method was designed to find similar-to relation rather than subsumption relation. The similar-to relation is symmetric, while subsumption relation is more strict and asymmetric. In our experiments, we use the same evaluation process to evaluate TAG-TAG, TAG-WORD, TAG-REASON and HEYMANN, in which only subsumption relations will be marked as correct.

5.2.4 Results

For each method, we set the cut-off threshold n from 1 to 500, so as to plot the psrecision-coverage curves. The result is shown in Figure 4. The larger the area under the curve, the better the method's performance.

We have three observations from Figure 4. First, TAG-REASON has the best performance

[2] http://www.douban.com



(a) BLOG

(b) BOOK

Figure 4: The precision and coverage of TAG-TAG, TAG-WORD, TAG-REASON and HEYMANN methods. The larger the area under the curve, the better the result. The cut-off threshold $n \in [1, 500]$.

BLOG			BOOK		
Insufficient	Reversed	Irrelevant	Insufficient	Reversed	Irrelevant
childedu $\rightarrow_s$ father childedu $\rightarrow_s$ grandma emotion$\rightarrow_s$warm childedu$\rightarrow_s$child education$\rightarrow_s$child	stock $\rightarrow_s$ security stock $\rightarrow_s$ financial delicious$\rightarrow_s$taste delicious$\rightarrow_s$food earthquake$\rightarrow_s$disaster	travel$\rightarrow_s$building emotion $\rightarrow_s$time emotion$\rightarrow_s$original culture$\rightarrow_s$spring poem$\rightarrow_s$night	textbook $\rightarrow_s$ exam history $\rightarrow_s$ military piano$\rightarrow_s$scores history$\rightarrow_s$culture novel$\rightarrow_s$love	English $\rightarrow_s$ foreignlang biography$\rightarrow_s$people jpbuilding$\rightarrow_s$jpculture novel$\rightarrow_s$pureliterature ancientgreek$\rightarrow_s$greek	japan $\rightarrow_s$lightnovel building$\rightarrow_s$textbook sales$\rightarrow_s$O japan $\rightarrow_s$ shower photo$\rightarrow_s$umbrella
Total 52%	Total 14%	Total 34%	Total 37%	Total 48%	Total 15%

Table 2: Examples of mistakes and the percentage of each mistake type.

on both data sets: On the BOOK data set, TAG-REASON outperforms others by a marked margin; On the BLOG data set, TAG-REASON has higher precision when coverage is smaller (which means within top-ranked relations), and has comparable precision to TAG-TAG when coverage increases. Second, similarity-based clustering method (namely HEYMANN) performed worse than others, suggesting it may not be adequate for discovering subsumption relation. Third, while also using content information, TAG-WORD performs poorer than both TAG-REASON and TAG-TAG, which suggests that noise in the content would prevent TAG-WORD from getting the correct estimation of $p(t_a|t_b)$.

To summarize, by leveraging *relevant* content, TAG-REASON could discover better subsumption relations than just using tag-tag co-occurrences and similarity-based hierarchical clustering.

5.2.5 Mistakes in Discovered Relations

We also studied the type of mistakes in subsumption relation discovery. To our observation, a mistakenly discovered relation $t_a \rightarrow_s t_b$ falls into one of the following categories:

1. **insufficient** t_a relates with t_b, but using t_b does not implies the using of t_a in all cases.
2. **reversed** $t_b \rightarrow_s t_a$ is correct, while $t_a \rightarrow_s t_b$ is not.
3. **irrelevant** There is no obvious connection between t_a and t_b.

We collected all incorrect relations discovered by the TAG-REASON method. Then, the type of mistake for each relation is labeled manually. The result is shown in Table 2, along with selected examples of each type.

Table 2 shows different error patterns for BLOG and BOOK. In BLOG, most of the mistakes are of the type *insufficient*. Taking "`education` $\rightarrow_s$ `child`" for example, annotating a document as `child` does not imply that it is about child education, it may about food or clothes for a child. In BOOK, most of the mistakes are *reversed* mistakes, which is a result of the omitted-tag problem discussed in Section 3.2.1.

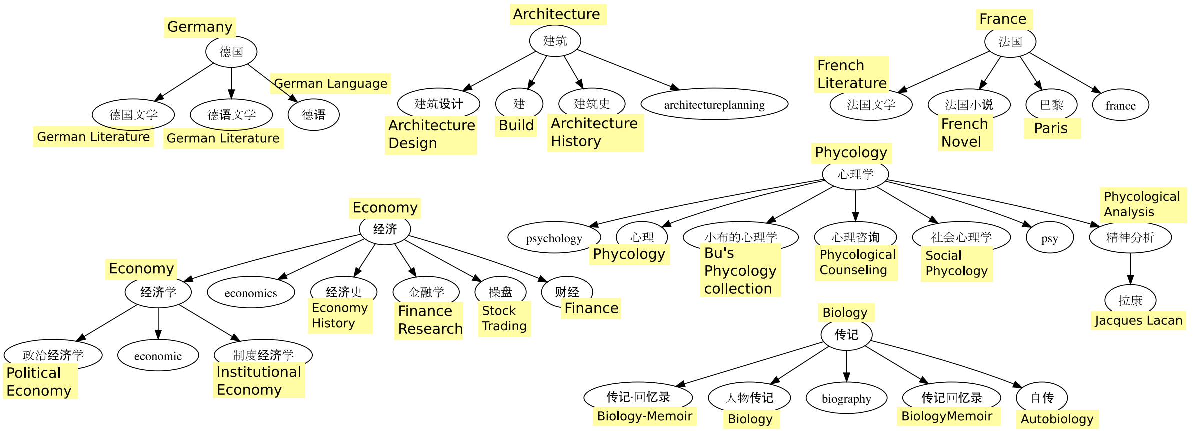


Figure 5: Part of the constructed Layered-DAG from the BOOK data set.

	BLOG		BOOK	
Method	Precision	Coverage	Precision	Coverage
TAG-TAG	-4.7%	$+7.9\%$	-7.4%	$+12.5\%$
TAG-WORD	0%	0%	-9.0%	$+2.2\%$
TAG-REASON	-3.6%	$+5.4\%$	-0.9%	$+5.4\%$

Table 3: The effects on precision and coverage by Layered-DAG construction

5.3 Layered-DAG Construction

Using the algorithm introduced in Section 4, we constructed Layered-DAGs from the discovered relations. Constructing Layered-DAG will remove certain relations, which will decrease the precision and increase the coverage. Table 3 shows the changes of precision and coverage brought by Layered-DAG construction. In most of the cases, the increasing of coverage is more than the decreasing of precision.

As a representative example, we show part of a constructed Layered-DAG from the BOOK data set in Figure 5, since the whole graph is too big to fit in the paper. All tags in Chinese are translated to English.

6 Conclusion and Future Work

In this paper, we explored the structure of social tags by discovering subsumption relations. First, we defined the subsumption relation $t_a \rightarrow_s t_b$ as t_a can be used to replace t_b without ambiguity. Then, we cast the subsumption relation identification problem to the estimation of $p(t_a|t_b)$. We proposed three methods, namely TAG-TAG, TAG-WORD and TAG-REASON, while the last two leverage the content of document to help estimation. We also proposed an greedy algorithm for constructing a Layered-DAG from the discovered relations, which helps minimizing redundancy.

We performed experiments on two real world data sets, and evaluated the discovered subsumption relations quantitatively by pooling. The results showed that the proposed methods outperform similarity-based hierarchical clusteing in finding subsumption relations. The TAG-REASON method, which uses only the relevant content to the tags, has the best performance. Empirical study showed that Layered-DAG construction works effectively as expected.

The results suggest two directions for future work: First, more ways for $p(t_a|t_b)$ estimation could be explored, for example, combining TAG-TAG and TAG-REASON; Second, external knowledge, such as the Wikipedia and the WordNet, could be exploited as background knowledge to improve the accuracy.

ACKNOWLEDGEMENTS

This work is supported by the National Science Foundation of China under Grant No. 60873174 and the National 863 High-Tech Program of China under Grant No. 2007AA01Z148. We also thank Douban Inc.(www.douban.com) for providing the DOUBAN data set, and Shoukun Wang, Guozhu Wen et al. of Douban Inc. for insightful discussion.

References

Angeletou, S., M. Sabou, L. Specia, and E. Motta. 2007. Bridging the gap between folksonomies and the semantic web: An experience report. In *Workshop: Bridging the Gap between Semantic Web and Web*, volume 2. Citeseer.

Begelman, Grigory, Keller, and F. Smadja. 2006. Automated tag clustering: Improving search and exploration in the tag space. In *Collaborative Web Tagging Workshop, 15 th International World Wide Web Conference*.

Brooks, Christopher H. and Nancy Montanez. 2006. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 625–632, New York, NY, USA. ACM.

Echarte, F., J. J. Astrain, A. Córdoba, and J. Villadangos. 2007. Ontology of folksonomy: A New modeling method. *Proceedings of Semantic Authoring, Annotation and Knowledge Markup (SAAKM)*.

Heymann, Paul and Hector Garcia-Molina. 2006. Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Stanford University, April.

Kim, Hak L., Simon Scerri, John G. Breslin, Stefan Decker, and Hong G. Kim. 2008. The state of the art in tag ontologies: a semantic model for tagging and folksonomies. In *DCMI '08: Proceedings of the 2008 International Conference on Dublin Core and Metadata Applications*, pages 128–137. Dublin Core Metadata Initiative.

Kome, Sam H. 2005. Hierarchical subject relationships in folksonomies. Master's thesis, University of North Carolina at Chapel Hill, November.

Mika, P. 2005. Ontologies are us: A unified model of social networks and semantics. *The Semantic Web–ISWC 2005*, pages 522–536.

Passant, Alexandre. 2007. Using ontologies to strengthen folksonomies and enrich information retrieval in weblogs. In *Proceedings of International Conference on Weblogs and Social Media*.

Sanderson, M. and B. Croft. 1999. Deriving concept hierarchies from text. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–213. ACM.

Schmitz, P. 2006. Inducing ontology from flickr tags. In *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, pages 210–214. Citeseer.

Schwarzkopf, E., D. Heckmann, and D. Dengler. 2007. In *Workshop on Data Mining for User Modeling, ICUM'07*, page 63. Citeseer.

Shepitsen, Andriy, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. 2008. Personalized recommendation in collaborative tagging systems using hierarchical clustering. In *Proceedings of ACM RecSys'08*.

Si, Xiance and Maosong Sun. 2010. Tag allocation model: Modeling noisy social annotations by reason finding. In *Proceedings of 2010 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*.

Specia, Lucia and Enrico Motta. 2007. Integrating folksonomies with the semantic web. pages 624–639.

Van Damme, C., M. Hepp, and K. Siorpaes. 2007. Folksontology: An integrated approach for turning folksonomies into ontologies. *Bridging the Gap between Semantic Web and Web*, 2:57–70.

Voorhees, E.M. and D.K. Harman. 2005. *TREC: Experiment and evaluation in information retrieval*. MIT Press.

Wu, Harris, Mohammad Zubair, and Kurt Maly. 2006a. Harvesting social knowledge from folksonomies. In *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 111–114, New York, NY, USA. ACM.

Wu, Xian, Lei Zhang, and Yong Yu. 2006b. Exploring social annotations for the semantic web. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 417–426, New York, NY, USA. ACM.

Ukwabelana - An open-source morphological Zulu corpus

Sebastian Spiegler
Intelligent Systems Group
University of Bristol
spiegler@cs.bris.ac.uk

Andrew van der Spuy
Linguistics Department
University of the Witwatersrand
andrew.vanderspuy@wits.ac.za

Peter A. Flach
Intelligent Systems Group
University of Bristol
peter.flach@bris.ac.uk

Abstract

Zulu is an indigenous language of South Africa, and one of the eleven official languages of that country. It is spoken by about 11 million speakers. Although it is similar in size to some Western languages, e.g. Swedish, it is considerably under-resourced. This paper presents a new open-source morphological corpus for Zulu named *Ukwabelana corpus*. We describe the agglutinating morphology of Zulu with its multiple prefixation and suffixation, and also introduce our labeling scheme. Further, the annotation process is described and all single resources are explained. These comprise a list of 10,000 labeled and 100,000 unlabeled word types, 3,000 part-of-speech (POS) tagged and 30,000 raw sentences as well as a morphological Zulu grammar, and a parsing algorithm which hypothesizes possible word roots and enumerates parses that conform to the Zulu grammar. We also provide a POS tagger which assigns the grammatical category to a morphologically analyzed word type. As it is hoped that the corpus and all resources will be of benefit to any person doing research on Zulu or on computer-aided analysis of languages, they will be made available in the public domain from http://www.cs.bris.ac.uk/Research/MachineLearning/Morphology/Resources/.

1 Introduction

Zulu (also known as isiZulu) is a Bantu language of South Africa, classified as S.30 in Guthrie's classification scheme (Guthrie, 1971). Since 1994, it has been recognized as one of the eleven official languages of South Africa. It has a written history of about 150 years: the first grammar was published by Grout (1859), and the first dictionary by Colenso (1905). There are about 11 million mother-tongue speakers, who constitute approximately 23% of South Africa's population, making Zulu the country's largest language.

Zulu is highly mutually intelligible with the Xhosa, Swati and Southern Ndebele languages, and with Ndebele of Zimbabwe (Lanham, 1960), to the extent that all of these can be considered dialects or varieties of a single language, Nguni. Despite its size, Zulu is considerably *under-resourced*, compared to Western languages with similar numbers of speakers, e.g. Swedish. There are only about four regular publications in Zulu, there are few published books, and the language is not used as a medium of instruction.

This of course is partly due to the short time-span of its written history, but the main reason, of course, is the apartheid history of South Africa: for most of the twentieth century resources were allocated to Afrikaans and English, the two former official languages, and relatively few resources to the indigenous Bantu languages. Since 1994, Zulu has had a much larger presence in the media, with several television programs being broadcast in Zulu every day. Yet much needs to be done in order to improve the resources available to Zulu speakers and students of Zulu.

The aim of the project reported in this paper was to establish a Zulu corpus, named the *Ukwabelana corpus*[1], consisting of morphologically labeled words (that is, word types) and part-of-speech (POS) tagged sentences. Along with the labeled corpus, unlabeled words and sentences, a morphological grammar, a semi-automatic mor-

[1] *Ukwabelana* means 'to share' in Zulu where the 'k' is pronounced voiced like a [g].

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1020–1028,
Beijing, August 2010

phological analyzer and a *POS tagger* for morphologically analyzed words will be provided.

The sources used for the corpus were limited to fictional works and the Zulu Bible. This means that there is not a wide variety of registers, and perhaps even of vocabulary items. This defect will have to be corrected in future work.

The *Ukwabelana corpus* can be used to develop and train automatic morphological analyzers, which in turn tag a large corpus of written Zulu, similar to the Brown corpus or the British National Corpus. Moreover, the list of POS tagged sentences is an essential step towards building an automatic syntactic tagger, which still does not exist for Zulu, and a tagged corpus of Zulu. Such a corpus would be beneficial to language researchers as it provides them with examples of actual usage, as opposed to elicited or invented examples, which may be artificial or unlikely to occur in real discourse. This would greatly improve the quality of Zulu dictionaries and grammars, most of which rely heavily on the work of Doke (1927) and Doke, Malcom and Sikakana (1958), with little in the way of innovation. Morphological tagging is also useful for practical computational applications like predictive text, spell-checking, grammar checking and machine translation; in the case of Zulu, where a large percentage of grammatical information is conveyed by prefixes and suffixes rather than by separate words, it is essential. For example, in English, the negative is expressed by means of a separate word *'not'*, but in Zulu the negative is constructed using a prefix-and-suffix combination on the verb, and this combination differs according to the mood of the verb (indicative, participial or subjunctive). The practical computational applications mentioned could have a very great impact on the use of Zulu as a written language, as spell-checking and grammar checking would benefit proofreaders, editors and writers. Machine translation could aid in increasing the number of texts available in Zulu, thus making it more of a literary language, and allowing it to become established as a language of education. The use of Zulu in public life could also increase. Currently, the tendency is to use English, as this is the language that reaches the widest audience. If high-quality automatic translation becomes available, this would no longer be necessary. As it is hoped that the *Ukwabelana corpus* will be of benefit to any person doing research on Zulu or on computer-aided analysis of languages, it will be made available as the first morphologically analysed corpus of Zulu in the public domain.

2 Related work

In this section, we will give an overview of linguistic research on Nguni languages, following the discussions in van der Spuy (2001), and thereafter a summary of computational approaches to the analysis of Zulu.

2.1 Linguistic research on Nguni languages

The five Nguni languages Zulu, Xhosa, South African Ndebele, Swati, and Zimbabwean Ndebele are highly mutually intelligible, and for this reason, works on any of the other Nguni languages are directly relevant to an analysis of Zulu.

There have been numerous studies of Nguni grammar, especially its morphology; in fact, the Nguni languages probably rival Swahili and Chewa for the title of most-studied Bantu language. The generative approach to morphological description (as developed by Aronoff (1976), Selkirk (1982), Lieber (1980), Lieber (1992)) has had very little influence on most of the work that has been done on Nguni morphology.

Usually, the descriptions have been atheoretical or structuralist. Doke's paradigmatic description of the morphology (Doke, 1927; Doke, 1935) has remained the basis for linguistic work in the Southern Bantu languages. Doke (1935) criticized previous writers on Bantu grammars for basing their classification, treatment and terminology on their own mother tongue or Latin. His intention was to create a grammatical structure for Bantu which did not conform to European or classical standards. Nevertheless, Doke himself could not shake off the European mindset: he treated the languages as if they had inflectional paradigms, with characteristics like subjunctive or indicative belonging to the whole word, rather than to identifiable affixes; in fact, he claimed (1950) that Bantu languages are "inflectional with [just] a tendency to agglutination", and assumed that the morphology was linear not hierarchical. Most subsequent linguistic studies and reference grammars of the Southern Bantu languages have been directed at refining or redefining Doke's categories from a paradigmatic perspective.

Important Nguni examples are Van Eeden (1956), Van Wyk (1958), Beuchat (1966), Wilkes (1971), Nkabinde (1975), Cope (1984), Davey (1984), Louw (1984), Ziervogel et al. (1985), Gauton (1990), Gauton (1994), Khumalo (1992), Poulos and Msimang (1998), Posthumus (1987), Posthumus (1988), Posthumus (1988) and Posthumus (2000). Among the very few generative morphological descriptions of Nguni are Lanham (1971), Mbadi (1988) and Du Plessis (1993). Lanham (1971) gives a transformational analysis of Zulu adjectival and relative forms. This analysis can be viewed as diachronic rather than synchronic. Mbadi (1988) applies Lieber (1980) and Selkirk's percolation theory (Selkirk, 1982) to a few Xhosa morphological forms. Du Plessis (1993) gives a hierarchical description of the morphology of the verb, but he assumes that derivation is syntactical rather than lexical.

In short, there has been no thorough-going generative analysis of the morphology which has treated the Nguni languages as agglutinative rather than inflectional.

2.2 Computational approaches to analyzing Zulu

In the last decade, various computational approaches for Zulu have been reported. Based on the *Xerox finite-state toolbox* by Beesley and Karttunen (2003), Pretorius and Bosch (2003) developed a prototype of a computational morphological analyzer for Zulu. Using a semi-automated process, a morphological lexicon and a rule-base were built incrementally. Later work (Pretorius and Bosch, 2007) dealt with overgeneration of the Zulu finite-state tool concerning locative formation from nouns and verbal extensions to verb roots. Pretorius and Bosch (2009) also used cross-linguistic similarities and dissimilarities of Zulu to bootstrap a morphological analyser for Xhosa. Joubert et al. (2004) followed a bootstrapping approach to morphological analysis. A simple framework uses morpheme lists, morphophonological and morphosyntactic rules which are learnt by consulting an *oracle*, in their case a linguistic expert who corrects analyses. The framework then revises its grammar so that the updated morpheme lists and rules do not contradict previously found analyses. Botha and Barnard (2005) compared two approaches for gathering Zulu text corpora from the World Wide Web. They drew the conclusion that using commercial search engines for finding Zulu websites outperforms webcrawlers even with a carefully selected starting point. They saw the reason for that in the fact that most documents on the internet are in one of the world's dominant languages. Bosch and Eiselen (2005) presented a spell checker for Zulu based on morphological analysis and regular expressions. It was shown that after a certain threshold for the lexicon size performance could only be improved by incrementally extending morphological rules. Experiments were performed for basic and complex Zulu verbs and nouns, and large numbers of words still were not recognized. Spiegler et al. (2008) performed experiments where they tested four machine learning algorithms for morphological analysis with different degrees of supervision. An unsupervised algorithm analyzed a raw word list, two semi-supervised algorithms were provided with word stems and subsequently segmented prefix and suffix sequences, and the supervised algorithm used a language model of analysed words which was applied to new words. They experimentally showed that there is a certain trade-off between the usage of labeled data and performance. They also reckoned that computational analysis improves if words of different grammatical categories are analysed separately since there exist homographic morphemes across different word categories.

3 Zulu morphology

Zulu is an agglutinative language, with a complex morphology. It presents an especial problem for computational analysis, because words usually incorporate both prefixes and suffixes, and there can be several of each. This makes it hard to identify the root by mechanical means, as the root could be the first, second, third, or even a later morpheme in a word. The complexities involved are exacerbated by the fact that a considerable number of affixes, especially prefixes, have allomorphic forms. This is largely brought about by the fact that Zulu has a prohibition against sequences of vowels, so that a prefix whose canonical form is *nga-* will have an allomorph *ng-* before roots that begin with vowels. Given a sequence *nga-*, then, it is possible that it constitutes an entire morpheme, or the beginning of a morpheme like the verb root *ngabaz-* 'to be uncertain', or a morpheme *ng-* followed by a vowel-commencing root like *and-* 'to increase'. Furthermore, many morphemes are homographs, so that the prefix *nga-* could represent either the potential mood morpheme or a form of the negative that occurs in subordinate clauses; and the sequence *ng-* could be the allomorph of either of these, or of a number of homographic morphemes *ngi-*, which represent the first person singular in various moods. Besides these phonologically conditioned allomorphs, there are also morphologically conditioned ones, for example the locative prefix *e-* has an allomorph *o-* that occurs in certain morphological circumstances. Certain morpheme sequences also exhibit syncretism, so that while most nouns take a sequence of prefixes known as the initial vowel and the noun prefix, as in *i-mi-zi* 'villages', nouns of certain classes, like class 5, syncretise these two prefixes, as in *i-gama* 'name', where the prefix *i-* represents both the initial vowel and the noun prefix.

Like all other Bantu languages, Zulu divides its nouns into a number of classes. The class is often identifiable from the noun prefix that is attached to the noun, and it governs the *agreement* of all words that modify the noun, as well as of predicates of which the noun is a subject. Object agreement may also be marked on the predicate. Two examples of this agreement are given below.

Example 1.

Leso si-tshudeni e-si-hle e-ngi-si-fundis-ile si-phas-e kahle.
that student who-AGR-good who-I-him-teach-PAST AGR-pass-PAST well.
'That good student whom I taught passed well.'

Example 2.

Lowo m-fundi o-mu-hle e-ngi-m-fundis-ile u-phas-e kahle.
that learner who-AGR-good who-I-him-teach-PAST AGR-pass-PAST well.
'That good learner whom I taught passed well.'

The differences in agreement morphology in the two sentences is brought about because the nouns *sitshudeni* and *mfundi* belong to different classes. Canonici (1996) argues that a noun should be assigned to a class by virtue of the agreement that it takes. In terms of this criterion, there are twelve noun classes in Zulu. These classes are numbered 1–7, 9, 10, 11, 14, 15. The numbering system was devised by Meinhof (1906), and reflects the historical affinities between Zulu and other Bantu languages: Zulu lacks classes 8, 12 and 13, which are found in other Bantu languages. In the labels used on the database, morphemes that command or show agreement have been labeled as <*xn*>, where *x* is a letter or sequence of letters, and *n* is a number: thus the morpheme *m-* in *mfundi* is labeled <*n1*>, as it marks the noun as belonging to noun class *1*. The morpheme *si-* in *engisifundisile* is marked <*o7*>, as it shows object agreement with a noun of class *7*.

Zulu *predicatives* may be either verbal or non-verbal – the latter are referred to in the literature as copulatives. Copulatives usually consist of a predicative prefix and a base, which may be a noun, an adjective, or a prepositional, locative or adverbial form. There may also be various tense, aspect and polarity markers. They translate the English verb 'be', plus its complement – Zulu has no direct equivalent of 'be'; the verb *-ba*, which has the closest meaning, is probably better translated as 'become'. Examples of copulative forms are *ubenguthisha* 'he was a teacher', *zimandla* 'they are strong', *basekhaya* 'they are at home'. Predicatives may occur in a variety of moods, tenses, aspects and polarities; these are usually distinguished by the affixes attached to the base form. Thus in *engasesendlini* '(s)he no longer being in the house', the initial prefix *e-* indicates third person singular, class 1, participial mood; the prefix *nga-* denotes negative; the first prefix *se-* denotes continuative aspect; the second prefix *se-* is the locative prefix; *n-* shows that the noun belongs to class 9; *dl-* is the noun root meaning 'house', an allomorph of the canonical form *-dlu*; and *-ini* is the locative suffix. Thus in typical agglutinative manner, each affix contributes a distinctive part of the meaning of the word as a whole. This characteristic of the language was exploited in the labeling system used for the morphological corpus: labels were designed so as to indicate the grammatical function of the morpheme. A person searching for past tense negative verbs, for example, could simply search for the combination of <*past* >, <*neg*> and <*vr*>. A complete list of morphemes, allomorphs and their labels is provided along with the corpus and other resources.

According to the Dokean grammatical tradition (Doke, 1927), Zulu has a large number of parts of speech. This is because what would be separate words in other languages are often prefixes in Zulu, and also because various subtypes of determiner are given individual names. The parts of speech recognised in the corpus are: noun, verb, adjective, pronoun, adverb, conjunction, prepositional, possessive, locative, demonstrative, presentative, quantitative, copulative and relative.

Adjective includes the traditional Dokean adjective (a closed class of roots which take noun prefixes as their agreement prefixes) and the predicative form of the Dokean relative, which is seen as an open class of adjectives (cf. van der Spuy (2006)). *Pronouns* are the personal pronouns, which may also (sometimes in allomorphic form) be used as agreement morphemes in quantifiers. Adverbs may be forms derived from adjectives by prefixing *ka-* to the root, or morphologically unanalysable forms like *phansi* 'in front, forward'. Ideophones have been included as adverbs. *Prepositionals* are words that incorporate the Dokean "adverbials" *na-* 'with', *nga-* 'by means of', *njenga-* 'like', *kuna-* 'more than', etc., which are better analysed as prepositions. The presentative is Doke's "locative demonstrative copulative" - the briefer name was suggested by van der Spuy (2001). *Copulatives* are all Doke's copulatives, excluding the adjectives mentioned above. *Relatives* are all predicative forms incorporating a relative prefix.

4 The labeling scheme

The labeling scheme has been based on the idea that each morpheme in a word should be labeled, even when words belong to a very restricted class. For example, the demonstratives could have been labeled as composite forms, but instead it is assumed that demonstratives contain between one and three morphemes, e.g. *le<d>si<d7>ya<po3>* 'a demonstrative of the third position referring to class 7' - i.e.. 'that one yonder, class 7'. It should be possible from this detailed labeling to build up an amalgam of the morphological structure of the word. The labels have been chosen to be both as brief as possible and as transparent as possible, though transparency was often sacrificed for brevity. Thus indicative subject prefixes are labeled <*i1-15*>, relative prefixes are labeled <*r*>, and noun prefixes are labeled <*n1-15*>; but negative subject prefixes are labeled <*g1-15*> and possessive agreement prefixes are labeled <*z1-15*>. Sometimes a single label was used for several different forms, when these are orthographically distinct, so for example <*asp*> (aspect) is used as a label for the following, among others: the continuative prefix *sa-* and its allomorph *se-*, the exclusive prefix *se-*, and the potential prefix *nga-* and its allomorph *ng-*. A person searching for forms containing the potential aspect would have to search for *'nga<asp> + ng<asp>'*. However, there should be no ambiguity, as the orthographic form would eliminate this. The detailed description of the scheme is provided by Spiegler et al. (2010).

5 Annotation process

The goal of this project was to build a reasonably sized corpus of morphologically annotated words of high quality which could be later used for developing and training automatic morphological analyzers. For this reason, we had gathered a list of the commonest Zulu word types, defined a partial grammar and parsed Zulu words with a logic algorithm which proposes possible parses based on the partial grammar. Compared to a completely manual approach, this framework provided possible annotations to choose from or the option to type in an annotation if none of the suggestions was the correct one. This semi-automatic process speeded up the labeling by an estimated factor of 3-4, compared to a purely manual approach. In Figure 1 we illustrate the annotation process and in the following subsections each step is detailed.

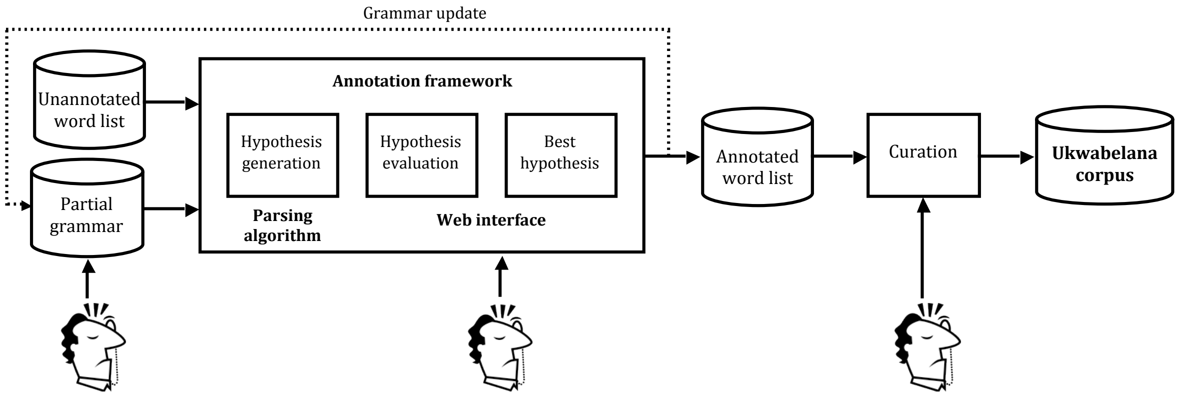


Figure 1: Process view of the annotation.

5.1 Unannotated word list

A list of unannotated Zulu words has been compiled from fictional works and the Zulu Bible. The original list comprises around 100,000 of the commonest Zulu word types. No information, morphological or syntactic, was given along with the words. We selected an initial subset of 10,000 words although our long-term goal is the complete analysis of the entire word list.

5.2 Partial grammar

Our choice for representing the morphological Zulu grammar was the formalism of *Definite Clause Grammars (DCGs)* used in the logic programming language *Prolog*. Although we defined our grammar as a simple context-free grammar, DCGs can also express context-sensitive grammars by associating variables as arguments to non-terminal symbols (Gazdar and Mellish, 1989). When defining our morphological grammar, we assumed that a linguistic expert could enumerate all or at least the most important morphological rules and morphemes of 'closed' morpheme categories, e.g. prefixes and suffixes of nouns and verbs. Morphemes of 'open' categories like noun and verb roots, however, would need to be hypothesized during the semi-automatic analysis and confirmed by the linguistic expert. Our final grammar comprised around 240 morphological rules and almost 300 entries in the morpheme dictionary. Since we did not only want to recognize admissible Zulu words but also obtain their morphological structure, we needed to extend our DCG by adding parse construction arguments as shown in the example below.

Example 3.

```
w((X)) --> n(X).
n((X,Y,Z)) --> iv(X),n2(Y),nr(Z).
iv(iv(a)) --> [a].
n2(n2(ba))--> [ba].
```

A possible parse for the word `abantu` 'people' could be `iv(a),n2(ba),*nr(ntu)` where '$\star$' marks the hypothesized noun root.

With our partial grammar we could not directly use the inbuilt Prolog parser since we had to account for missing dictionary entries: Zulu verb and noun roots. We therefore implemented an algorithm which would generate hypotheses for possible parses according to our grammar. The algorithm will be described in the next subsection.

5.3 Hypothesis generation

For the hypothesis generation we reverted to logic programming and *abductive reasoning*. Abduction is a method of reasoning which is used with incomplete information. It generates possible hypotheses (parses) for an observation (word) and a given theory (grammar). Depending on the implementation, abduction finds the best hypothesis by evaluating all possible explanations. Our abductive algorithm is an extension of the meta-interpreter designed by Flach (1994) which only enumerates possible parses based on the grammar. A linguistic expert would then choose the best hypothesis. The algorithm invokes rules *top-down* starting with the most general until it reaches the last level of syntactic variables. These variables are then matched against their dictionary entries from the left to the right of the word. A possible parse is found if either all syntactic variables can be matched to existing dictionary entries or if an unmatched variable is listed as *abducible*. Abducibles are predefined non-terminal symbols whose dictionary entry can be hypothesized. In our case, abducibles were noun and verb roots.

5.4 Evaluation and best hypothesis

Our annotation framework only enumerated allowable parses for a given word, therefore a linguistic expert needed to evaluate hypotheses. We provided a *web-interface* to the annotation framework, so that multiple users could participate in the annotation process. They would choose either a single or multiple correct parses. If none of the hypotheses were correct, the user would provide the correct analysis. Although our grammar was incomplete it still generated a substantial number of hypotheses per word. These were in no particular order and a result of the inherent ambiguity of Zulu morphology. We therefore experimented with different ways of improving the presentation of parses. The most promising approach was structural sorting. Parses were alphabetically re-ordered according to their morphemes and labels such that similar results were presented next to each other.

5.5 Grammar update

The grammar was defined in an iterative process and extended if the linguistic expert found morphemes of closed categories which had not been listed yet or certain patterns of incomplete or incorrect parses caused by either missing or inaccurate rules. The updated rules and dictionary were considered for newly parsed words.

5.6 Annotated word list and curation process

Although there had been great effort in improving the hypothesis generation of the parsing algorithm, a reasonable number of morphological analyses still had to be provided manually. During the curation process, we therefore had to deal with removing typos and standardizing morpheme labels provided by different experts. In order to guarantee a high quality of the morphological corpus, we also inspected single labels and analyses for their correctness. This was done by examining frequencies of labels and label combinations assuming that infrequent labels and combinations were likely to be incorrect and needed to be manually examined again. The finally curated corpus has an estimated error of 0.4 ± 0.5 incorrect single labels and 2.8 ± 2.1 incorrect complete analyses per 100 parses. Along with each word's analysis we wanted to provide part-of-speech (POS) tags. This was done by using a set of rules which determine the POS tag based on the morphological structure. We developed a prototype of a POS tagger which would assign the part-of-speech to a given morphological analysis based on a set of 34 rules. A summary of morphological analyses and words is given in Table 1. The rules are provided in Spiegler et al. (2010).

Category	**# Analyses**	**# Word types**
Verb	6965	4825
Noun	1437	1420
Relative	1042	988
Prepositional	969	951
Possessive	711	647
Copulative	558	545
Locative	380	379
Adverb	156	155
Modal	113	113
Demonstrative	63	61
Pronoun	38	31
Interjection	24	24
Presentative	15	15
Adjective	14	14
Conjunction	3	3
Total #	12488	10171

Table 1: Categories of labeled words.

5.7 POS tagging of sentences

In addition to the list of morphologically labeled words, we assigned parts-of-speech to a subset of 30,000 Zulu sentences. This task is straightforward if each word of a sentence only belongs to a single grammatical category. This was the case for 2595 sentences. For 431 sentences, however, we needed to disambiguate POS tags. We achieved this by analysing the left and right context of a word form and selecting the most probable part-of-speech from a given list of possible tags.

The overall error is estimated at 3.1±0.3 incorrect POS tags per 100 words for the 3,000 sen-

Dataset	# Sentences	# Word tokens	#Word types	# Words per sentence	Word length
Raw	29,424	288,106	87,154	9.79±6.74	7.49±2.91
Tagged	3,026	21,416	7,858	7.08±3.75	6.81±2.68

Table 2: Statistics of raw and POS-tagged sentences.

tences we tagged. The summary statistics for raw and tagged sentences are shown in Table 2.

6 The Ukwabelana corpus - a resource description

The *Ukwabelana corpus* is three-fold:

1. It contains 10,000 morphologically labeled words and 3,000 POS-tagged sentences.
2. The corpus also comprises around 100,000 common Zulu word types and 30,000 Zulu sentences compiled from fictional works and the Zulu Bible, from which the labeled words and sentences have been sampled.
3. Furthermore, all software and additional data used during the annotation process is provided: the partial grammar in DCG format, the abductive algorithm for parsing with incomplete information and a prototype for a POS tagger which assigns word categories to morphologically analyzed words.

We are making these resources publicly available from `http://www.cs.bris.ac.uk/Research/MachineLearning/Morphology/Resources/` so that they will be of benefit to any person doing research on Zulu or on computer-aided analysis of languages.

7 Conclusions and future work

In this paper, we have given an overview of the morphology of the language Zulu, which is spoken by 23% and understood by more than half of the South African population. As an indigenous language with a written history of 150 years which was only recognised as an official languages in 1994, it is considerably under-resourced. We have spent considerable effort to compile the first open-source corpus of labeled and unlabeled words as well as POS-tagged and untagged sentences to promote research on this Bantu language. We have described the annotation process and the tools for compiling this corpus. We see this work as a first step in an ongoing effort to ultimately label the entire word and sentence corpus.

Our future work includes further automation of the annotation process by extending the described abductive algorithm with a more sophisticated hypothesis evaluation and by combining syntactical and morphological information during the decision process. Our research interest also lies in the field of automatic grammar induction which will help to refine our partial grammar. Another aspect is interactive labeling where a linguistic expert directs the search of an online parsing algorithm by providing additional information. Apart from the benefits to language researchers, we foresee an application of the corpus by machine learners which can develop and train their algorithms for morphological analysis.

Acknowledgements

We would like to thank Etienne Barnard and the Human Language Technologies Research Group from the Meraka Institute for their support during this project. Furthermore, we want to acknowledge Johannes Magwaza, Bruno Golénia, Ksenia Shalonova and Roger Tucker. The research work was sponsored by EPSRC grant EP/E010857/1 *Learning the morphology of complex synthetic languages* and a grant from the NRF (S. Africa).

References

Aronoff. 1976. *Word Formation in Generative Grammar*. The MIT Press.

Beesley and Karttunen. 2003. *Finite State Morphology*. University of Chicago Press.

Beuchat. 1966. The Verb in Zulu. *African Studies*, 22:137–169.

Bosch and Eiselen. 2005. The Effectiveness of Morphological Rules for an isiZulu Spelling Checker. *S. African Journal of African Lang.*, 25:25–36.

Botha and Barnard. 2005. Two Approaches to Gathering Text Corpora from the World Wide Web. *16th Ann. Symp. of the Pattern Recog. Ass. of S. Africa*.

Canonici. 1996. *Zulu Grammatical Structure.* Zulu Lang. and Literature, University of Natal, Durban.

Colenso. 1905. *Zulu-English Dictionary*. Natal, Vause, Slatter & Co.

Cope. 1984. An Outline of Zulu Grammars. *African Studies*, 43(2):83–102.

Davey. 1984. Adjectives and Relatives in Zulu. *S. African Journal of African Lang.*, 4:125–138.

Doke. 1927. *Text Book of Zulu Grammar*. Witwatersrand University Press.

Doke. 1935. *Bantu Linguistic Terminology*. Longman, Green and Co, London.

Doke. 1954. *Handbook of African Lang.*, chapter The S.ern Bantu Lang. Oxford University Press.

Doke, Malcom and Sikakana. 1958. *Zulu-English vocabulary*. Witwatersrand Uni. Press.

Du Plessis. 1993. *Linguistica: Festschrift EB van Wyk*, chapter Inflection in Syntax, pp. 61–66. Van Schaik, Pretoria.

Flach. 1994. *Simply Logical*. John Wiley.

Gauton. 1990. Adjektiewe en Relatiewe in Zulu. Master's thesis, University of Pretoria.

Gauton. 1994. Towards the Recognition of a Word Class 'adjective' for Zulu. *S. African Journal of African Lang.*, 14:62–71.

Gazdar and Mellish. 1989. *Natural Language Processing in Prolog*. Addison-Wesley.

Grout. 1859. *The Isizulu: A Grammar Of The Zulu Lang.* Kessinger Publishing.

Guthrie. 1971. *Comparative Bantu: An Introduction to the Comparative Linguistics and Prehistory of the Bantu Lang.* Farnborough, Gregg International Publishers.

Joubert, Zimu, Davel, and Barnard. 2004. A Framework for Bootstrapping Morphological Decomposition. Tech. report, CSIR/University of Pretoria, S. Africa.

Khumalo. 1992. *African Linguistic Contributions*, chapter The morphology of the direct relative in Zulu. Via Afrika.

Lanham. 1960. *The Comparative Phonology of Nguni*. Ph.D. thesis, Witwatersrand Uni., Jo'burg, S. Africa.

Lanham. 1971. The Noun as Deep-Structure Source for Nguni Adjectives and Relatives. *African Studies*, 30:294–311.

Lieber. 1980. *On the Organization of the Lexicon*. Ph.D. thesis, Massachusetts Institute of Technology.

Lieber. 1992. *Deconstructing Morphology*. The University of Chicago Press.

Louw. 1984. Word Categories in Southern Bantu. *African Studies*, 43(2):231–239.

Mbadi. 1988. *Anthology of Articles on African Linguistics and Literature*, chapter The Percolation Theory in Xhosa Morphology. Lexicon, Jo'burg.

Meinhof. 1906. *Grundzüge einer Vergleichenden Grammatik der Bantusprachen*. Reimer, Berlin.

Nkabinde. 1975. *A Revision of the Word Categories in Zulu*. Ph.D. thesis, University of S. Africa.

Posthumus. 1987. Relevancy and Applicability of Terminology Concerning the Essential Verb Categories in African Lang. *Logos*, 7:185–212.

Posthumus. 1988. Identifying Copulatives in Zulu and S.ern Sotho. *S. African Journal of African Lang.*, 8:61–64.

Posthumus. 2000. The So-Called Adjective in Zulu. *S. African Journal of African Lang.*, 20:148–158.

Poulos and Msimang. 1998. *A Linguistic Analysis of Zulu*. Via Afrika.

Pretorius and Bosch. 2003. Finite-State Computational Morphology: An Analyzer Prototype For Zulu. *Machine Translation*, 18:195–216.

Pretorius and Bosch. 2007. Containing Overgeneration in Zulu Computational Morphology. *Proceedings of 3rd Lang. and Technology Conference*, pp. 54 – 58, Poznan.

Pretorius and Bosch. 2009. Exploiting Cross-Linguistic Similarities in Zulu and Xhosa Computational Morphology. *Workshop on Lang. Technologies for African Lang. (AfLaT)*, pp. 96–103.

Selkirk. 1982. *The Syntax of Words*. MIT Press.

Spiegler, Golenia, Shalonova, Flach, and Tucker. 2008. Learning the Morphology of Zulu with Different Degrees of Supervision. *IEEE Workshop on Spoken Lang. Tech.*

Spiegler, van der Spuy, Flach. 2010. Additional material for the Ukwabelana Zulu corpus. Tech. report, University of Bristol, U.K.

van der Spuy. 2001. *Grammatical Structure and Zulu Morphology*. Ph.D. thesis, University of the Witwatersrand, Jo'burg, S. Africa.

van der Spuy. 2006. Wordhood in Zulu. *S.ern African Linguistics and Applied Lang. Studies*, 24(3):311–329.

Van Eeden. 1956. *Zoeloe-Grammatika*. Pro Ecclesia, Stellenbosch.

Van Wyk. 1958. *Woordverdeling in Noord-Sotho en Zulu: 'n bydrae tot die vraagstuk van word-identifikasie in die Bantoetale*. Ph.D. thesis, University of Pretoria.

Wilkes. 1971. *Agtervoegsels van die werkwoord in Zulu*. Ph.D. thesis, Rand Afrikaans University.

Ziervogel, Louw, and Taljaard. 1985. *A Handbook of the Zulu Lang.* Van Schaik, Pretoria.

EMMA: A Novel *E*valuation *M*etric for *M*orphological *A*nalysis

Sebastian Spiegler
Intelligent Systems Group
University of Bristol
spiegler@cs.bris.ac.uk

Christian Monson
Center for Spoken Language Understanding
Oregon Health & Science University
monsonc@csee.ogi.edu

Abstract

We present a novel *E*valuation *M*etric for *M*orphological *A*nalysis (*EMMA*) that is both linguistically appealing and empirically sound. *EMMA* uses a graph-based assignment algorithm, optimized via integer linear programming, to match morphemes of predicted word analyses to the analyses of a morphologically rich answer key. This is necessary especially for unsupervised morphology analysis systems which do not have access to linguistically motivated morpheme labels. Across 3 languages, *EMMA* scores of 14 systems have a substantially greater positive correlation with mean average precision in an information retrieval (IR) task than do scores from the metric currently used by the Morpho Challenge (MC) competition series. We compute *EMMA* and MC metric scores for 93 separate system-language pairs from the 2007, 2008, and 2009 MC competitions, demonstrating that *EMMA* is not susceptible to two types of gaming that have plagued recent MC competitions: Ambiguity Hijacking and Shared Morpheme Padding. The *EMMA* evaluation script is publicly available from `http://www.cs.bris.ac.uk/Research/MachineLearning/Morphology/Resources/`.

1 Introduction

Words in natural language are constructed from smaller building blocks called *morphemes*. For example, the word *wives* breaks down into an underlying stem, *wife*, together with a *plural* suffix. Analyzing the morphological structure of words is known to benefit a variety of downstream natural language (NL) tasks such as speech recognition (Creutz, 2006; Arısoy et al., 2009), machine translation (Oflazer et al., 2007), and information retrieval (McNamee et al., 2008).

A variety of automatic systems can morphologically analyze words that have been removed from their surrounding context. These systems range from hand-built finite state approaches (Beesley and Karttunen, 2003) to recently proposed algorithms which learn morphological structure in an unsupervised fashion (Kurimo et al., 2007). Since unsupervised systems do not have access to linguistically motivated morpheme labels, they typically produce morphological analyses that are closely related to the written form. Such a system might decompose *wives* as *wiv -es*. Meanwhile, a hand-built system might propose *wife_N +Plural*, or even parse *wives* as a hierarchical feature structure. As morphological analysis systems produce such varied outputs, comparing decompositions from disparate systems is a challenge.

This paper describes *EMMA*, an *E*valuation *M*etric for *M*orphological *A*nalysis that quantitatively measures the quality of a set of morphological analyses in a linguistically adequate, empirically useful, and novel fashion. *EMMA* evaluates analyses that can be represented as a flat set of symbolic features, including hierarchical representations, which can be projected down to a linearized form (Roark and Sproat, 2007).

An automatic metric that discriminates between proposed morphological analyses should

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1029–1037,
Beijing, August 2010

fulfill certain computational and linguistic criteria. Computationally, the metric should:

1. *Correlate* with the performance of real-world NL processing tasks which embed the morphological analyses.
2. Be *Readily Computable*: The metric will only be useful if it is less time consuming and easier to compute than the larger NL task.
3. Be *Robust*: The metric should be difficult to game and should accurately reflect the distribution of predicted and true morphemes.
4. Be *Readily Interpretable*: When possible, the final numeric score should directly identify the strengths and weaknesses of the underlying morphological analysis system.

While accounting for these computational requirements, a morphology metric should still reward accurate models of linguistic structure. In particular, the metric should account for:

1. *Morphophonology*: Applying a morphological rule may alter the surface form of stem or affix. In the word *wives*, /waivz/, a rule of morphophonology voices the stem-final /f/ of *wife*, /waif/, when the plural suffix is added. A metric should penalize for not placing *wives* and *wife* as forms of the same lexeme.
2. *Allomorphy*: A metric should capture the successful grouping of allomorphs. The German plural has several surface allomorphs including *-en* in *Zeiten* (*times*), *-e* in *Hunde* (*dogs*), and *-s* in *Autos* (*cars*). A metric should reward a morphological analysis system that analyzes the different surface forms of the German plural as underlyingly identical.
3. *Syncretism*: In mirror fashion, a metric should reward analyses that distinguish between surface-identical syncretic morphemes: although *derives* and *derivations* both contain an *-s* morpheme, one marks *3rd person singular* and the other *plural*.
4. *Ambiguity*: Finally, a metric should account for legitimate morphological ambiguity. In Hebrew, the written word *MHGR* has three viable morphological segmentations: *M- H- GR*, "*from the foreigner*", *M- HGR*, "*from Hagar*", and the unsegmented form *MHGR*, meaning "*immigrant*" (Lavie et al., 2004). Absent disambiguating context, a morphological system should be rewarded for calling out all three analyses for *MHGR*.

Morphophonology, allomorphy, syncretism, and ambiguity are all common phenomena in the world's languages. The first three have all received much discussion in theoretical linguistics (Spencer and Zwicky, 2001), while morphological ambiguity has significant practical implications in NL processing, e.g. in machine translation of morphologically complex languages (Lavie et al., 2004; Oflazer et al., 2007).

In Section 2 we propose the metric *EMMA*, which has been specifically designed to evaluate morphological analyses according to our computational and linguistic criteria. Section 3 then describes and qualitatively critiques several well-used alternative metrics. Section 4 empirically compares *EMMA* against the qualitatively-strong metric used in the Morpho Challenge competition series (Kurimo et al., 2009). And we conclude in Section 5.

2 *EMMA*: An *E*valuation *M*etric for *M*orphological *A*nalysis

EMMA, the metric we propose for the evaluation of morphological analyses, like all the metrics that we consider in this paper, compares proposed morphological analyses against an answer key of definitively-analyzed words from a vocabulary. Since a set of proposed analyses is likely to use a different labeling scheme than the answer key, especially true of the output from unsupervised systems, *EMMA* does not perform a direct comparison among proposed and answer analyses. Instead, *EMMA* seeks a one-to-one relabeling of the proposed morphemes that renders them as similar as possible to the answer key. *EMMA*, then, measures the degree to which proposed analyses approximate an isomorphism of the answer key analyses. For exposition, we initially assume that, for each word, a single proposed analysis is scored against a single unambiguous answer analysis. We relax this restriction in Section 2.3, where EMMA scores multiple proposed analyses against a set of legitimately ambiguous morphological analyses.

To find the most appropriate one-to-one morpheme relabeling, *EMMA* turns to a standard algorithm from graph theory: optimal maximum matching in a bipartite graph. A *bipartite graph*, $G = \{X, Y; E\}$, consists of two disjoint sets of vertices, $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_m\}$, and a set of edges $e(x_i, y_j) \in E$ such that each edge has one end in X and the other end in Y. In *EMMA*, the set, A, of all unique morphemes in the answer key and the set, P, of all unique morphemes in the proposed analyses serve as the disjoint vertex sets of a bipartite graph.

A *matching* $M \subseteq E$ in a bipartite graph is defined as a set of edges $e(x_i, y_j)$ such that no x_i or y_j is repeated. A *maximum matching* is a matching where no M' with $|M'| > |M|$ exists. Furthermore, a weight $w(x_i, y_j) \in \Re$ may be assigned to each edge $e(x_i, y_j)$ of a bipartite graph. An *optimal assignment* is a maximum matching which also maximizes the sum of the weights of the edges of the matching

$$\sum_{e(x_i,y_j)\in M} w(x_i, y_j) \ .$$

EMMA weights the edge between a particular answer morpheme $a \in A$ and a proposed morpheme $p \in P$ as the number of words, w, in the vocabulary, V, where the answer analysis of w includes morpheme a while the proposed analysis includes p. *EMMA* constructs an optimal assignment maximum matching in this weighted bipartite morpheme graph. The edge weights ensure that the optimal matching will link the answer and proposed morphemes which globally occur in the analyses of the same words most often – restricting each answer morpheme to be represented by at most one proposed morpheme, and each proposed morpheme to represent at most one morpheme in the answer key. On the one hand, the restrictions thus imposed by bipartite matching penalize sets of proposed analyses that do not differentiate between surface-identical *syncretic morphemes*. On the other hand, the same one-to-one matching restrictions penalize proposed analyses that do not conflate *allomorphs* of the same underlying morpheme, whether those allomorphs are phonologically induced or not. Thus, *EMMA* meets our linguistic criteria from Section 1 of modeling syncretism, allomorphy, and morphophonology.

2.1 Maximum Matching by Integer Linear Programming

To construct the maximum matching optimal assignment of answer and proposed morphemes, *EMMA* uses standard integer linear programming techniques as implemented in *lpsolve* (Berkelaar et al., 2004). For the purpose of our integer program, we represent the weight of each potential edge of the optimal bipartite morpheme assignment in a count matrix $C = \{c_{ij}\}$ where c_{ij} is assigned the number of words $w \in V$ which share morpheme a_i in the answer key and p_j in the prediction. We then define a binary matrix $B = \{b_{ij}\}$ of the same dimensions as C. Each b_{ij} will be set to 1 if an edge exists from a_i to p_j in the optimal maximum matching, with $b_{ij} = 0$ otherwise. The integer linear program can then be defined as follows:

$$\operatorname*{argmax}_B \sum_{i,j} (C \cdot B)_{ij} \tag{1}$$

$$s.t. \sum_i b_{ij} \leq 1 \ , \quad \sum_j b_{ij} \quad \leq 1 \ , \quad b_{ij} \geq 0 \ ,$$

where $(C \cdot B)_{ij} = c_{ij} \cdot b_{ij}$ is the element-wise *Hadamard product*.

2.2 Performance Measures

Having settled on a maximum matching optimal assignment of proposed and answer morphemes, *EMMA* derives a final numeric score. Let w_k be the k^{th} word of V; and let A_k and P_k denote, respectively, the sets of morphemes in the answer key analysis of w_k and predicted analysis of w_k. Furthermore, let P_k^* denote the predicted morphemes for w_k where a morpheme p_j is replaced by a_i if $b_{ij} = 1$. Now that A_k and P_k^* contain morpheme labels that are directly comparable, we can define *precision* and *recall* scores for the proposed analysis of the word w_k. Precision is the fraction of correctly relabeled proposed morphemes from among all proposed morphemes of w_k; while *recall* is the number of correctly relabeled morphemes as a fraction of the answer key analysis of w_k. Precision and recall of the full vocabulary are the average word-level precision and recall:

$$precision = \frac{1}{|V|}\sum_{k}^{|V|}\frac{|A_k \bigcap P_k^*|}{|P_k^*|} \quad , \tag{2}$$

$$recall = \frac{1}{|V|}\sum_{k}^{|V|}\frac{|A_k \bigcap P_k^*|}{|A_k|} \quad . \tag{3}$$

Finally, *f-measure* is the harmonic mean of precision and recall:

$$f\text{-}measure = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad . \tag{4}$$

2.3 Morphological Ambiguity in *EMMA*

Thus far we have presented *EMMA* for the scenario where each word has a single morphological analysis. But, as we saw in Section 1 with the Hebrew word *MHGR*, natural language permits surface forms to have multiple legitimate morphological analyses. When a word is truly ambiguous, *EMMA* expects an answer key to contain a set of analyses for that word. Similarly, we permit sets of proposed alternative analyses. To extend *EMMA* with the ability to evaluate alternative analyses we first generalize the optimal maximum matching of morphemes from Section 2.1. We then define a new integer linear program to match answer and proposed *alternative analyses*. Finally, we adjust the performance measures of Section 2.2 to account for alternatives.

2.3.1 Ambiguity and Morpheme Matching

Let $A_{k,r}$ denote the r^{th} alternative answer analysis of the k^{th} word with $1 \leq r \leq m_k$, and let $P_{k,s}$ denote the s^{th} alternative prediction with $1 \leq s \leq n_k$, where m_k is the number of alternative analyses in the answer key and n_k the number of alternative predictions for w_k. We redefine $A_k = \bigcup_r^{m_k} A_{k,r}$ and $P_k = \bigcup_s^{n_k} P_{k,s}$ as the set of all answer or, respectively, predicted morphemes of w_k across all analysis alternatives. Instead of incrementing each c_{ij} entry in the count matrix C by a full count, we now add $\frac{1}{m_k \cdot n_k}$ to c_{ij} for all pairs $(a_i, p_j) \in A_k \times P_k$. This corresponds to counting each combination of an answer key and predicted morpheme normalized by the number of possible pairings between proposed and answer analysis alternatives. When both the answer and proposed analyses consist of just a single alternative, c_{ij} remains unchanged. Generalized morpheme matching still employs the linear program defined in Equation 1.

2.3.2 Matching of Alternative Analyses

After performing a one-to-one morpheme relabelling that accounts for ambiguity, we need to extend *EMMA* with the ability to evaluate alternative analyses. We again turn to optimal maximum matching in a bipartite graph: Where earlier we matched proposed and answer morphemes, now we match full proposed and answer analysis alternatives, maximizing the total number of correctly predicted morphemes across all alternatives. Generalizing on the notation of the unambiguous case, let $P_{k,s}^*$ denote the s^{th} alternative predicted analysis of the k^{th} word where predicted morphemes have been replaced by their assigned answer key morphemes. We introduce a new count matrix $C' = \{c'_{r,s}\}$, where $c'_{r,s}$ is the count of common morphemes of the r^{th} answer key alternative and s^{th} predicted alternative. Based on Equation 1, we calculate the binary matrix $B' = \{b'_{r,s}\}$ which contains the optimal assignment of the alternative answer key and predicted analyses for w_k.

2.3.3 Ambiguity and Performance Scores

We now adjust *EMMA*'s numeric performance measures to account for sets of ambiguous analysis alternatives. *Precision* becomes

$$\frac{1}{|V|}\sum_{k}^{|V|}\frac{1}{n_k}\sum_{r}^{m_k}\sum_{s}^{n_k} b'_{r,s}\frac{|A_{k,r} \bigcap P_{k,s}^*|}{|P_{k,s}^*|} \quad , \tag{5}$$

the ratio of correctly predicted morphemes across all predicted alternatives normalised by the number of predicted alternatives, n_k, and the vocabulary size, $|V|$. The factor $b'_{r,s}$ guarantees that scores are only averaged over pairs of proposed and answer analysis alternatives that have been assigned, that is, where $b'_{r,s} = 1$. *Recall* is measured similarly with

$$\frac{1}{|V|}\sum_{k}^{|V|}\frac{1}{m_k}\sum_{r}^{m_k}\sum_{s}^{n_k} b'_{r,s}\frac{|A_{k,r} \bigcap P_{k,s}^*|}{|A_{k,r}|} \quad . \tag{6}$$

Here, we normalize by m_k, the number of alternative analyses for the k^{th} word that are listed in the answer key. The normalisation factors $\frac{1}{m_k}$ and $\frac{1}{n_k}$ ensure that predicting too few or many alternative analyses is penalised.

3 Other Morphology Metrics

Having presented the *EMMA* metric for evaluating the quality of a set of morphological analyses, we take a step back and examine other metrics that have been proposed. Morphology analysis metrics can be categorized as either: 1. Directly comparing proposed analyses against an answer key, or 2. Indirectly comparing proposed and answer analyses by measuring the strength of an isomorphic-like relationship between the proposed and answer morphemes. The proposed *EMMA* metric belongs to the second category of isomorphism-based metrics.

3.1 Metrics of Direct Inspection

By Segmentation Point. Perhaps the most readily accessible automatic evaluation metric is a direct comparison of the morpheme boundary positions in proposed and answer analyses. As early as 1974, Hafer and Weiss used the direct boundary metric. Although intuitively simple, the segmentation point method implicitly assumes that it is possible to arrive at a valid morphological analysis by merely dividing the characters of a word into letter sequences that can be reconcatenated to form the original word. But, by definition, concatenation cannot describe non-contanative processes like morphophonology and allomorphy. Nor does simple segmentation adequately differentiate between surface-identical syncretic morphemes. Despite these drawbacks, precision and recall of segmentation points is still used in current morphological analysis research (Poon et al. (2009), Snyder and Barzilay (2008), Kurimo et al. (2006)).

Against Full Analyses. To confront the reality of non-concatenative morphological processes, an answer key can hold full morphological analyses (as opposed to merely segmented surface forms). But while a hand-built (Beesley and Karttunen, 2003) or supervised (Wicentowski , 2002) morphology analysis system can directly model the annotation standards of a particular morphological answer key, the label given to specific morphemes is ultimately an arbitrary choice that an unsupervised morphology induction system has no way to discover.

By Hand. On the surface, scoring proposed analyses by hand appears to provide a way to evaluate the output of an unsupervised morphology analysis system. Hand evaluation, however, does not meet our criteria from Section 1 for a robust and readily computable metric. It is time consuming and, as Goldsmith (2001) explains, leaves difficult decisions of what constitutes a morpheme to on-the-fly subjective opinion.

3.2 Metrics of Isomorphic Analysis

Recognizing the drawbacks of direct evaluation, Schone and Jurafsky (2001), Snover et al. (2002), and Kurimo et al. (2007) propose related measures of morphological analysis quality that are based on the idea of an isomorphism. For reasons that will be clear momentarily, we refer to the Schone and Jurafsky, Snover et al., and Kurimo et al. metrics as *soft* isomorphic measures. As discussed in Section 2, metrics of isomorphism measure similarities between the distribution of proposed morphemes and the distribution of answer morphemes, where proposed and answer morphemes may be disjoint symbol sets.

Unlike the *EMMA* metric proposed in Section 2, the soft metrics of isomorphism do not seek to explicitly link proposed morphemes to answer morphemes. Instead, their metrics group sets or pairs of words which share, in *either* the proposed analyses or in the answer analyses, a stem (Schone and Jurafsky, 2001; Snover, 2002), a suffix (Snover et al., 2002), or any arbitrary morpheme (Kurimo et al., 2007). The soft metrics subsequently note whether these same sets or pairs of words share any morpheme in the answer key or, respectively, in the proposed analyses. By foregoing a hard morpheme assignment, the soft metrics do not adequately punish sets of proposed and answer morphemes which fail to model syncretism and/or allomorphy. For example, proposed analyses that annotate *3rd person singular* and *plural* with a single undifferentiated +*s* morpheme will receive recall credit for both nouns and verbs.

3.3 The Morpho Challenge Metric

The Morpho Challenge (MC) competition series for unsupervised morphology analysis algorithms (Kurimo et al., 2009) has used a soft metric of isomorphism in its most recent three years of competition: 2007, 2008, and 2009. According to Kurimo et al. (2009) the Morpho Challenge (MC) measure samples *random word pairs* which share at least one common morpheme. Precision is calculated by generating random word pairs from the set of proposed analyses and then comparing the analyses of the word pairs in the answer key. The fraction of found and expected common morphemes is normalised by the number of words which are evaluated. *Recall* is defined in mirror fashion. The MC metric also normalizes precision and recall scores across sets of alternative analyses for each word in the proposal and answer key. To our knowledge the MC metric is the first isomorphism-based metric to attempt to account for *morphological ambiguity*. As we show in Section 4, however, MC's handling of ambiguity is easily gamed.

The MC metric does meet our criterion of being *readily computable* and, as we will show in the experimental section, the metric also *correlates* to a certain extent with performance on a higher-level natural language processing task. The downside of the MC metric, however, is *robustness*. In addition to MC's crude handling of ambiguity and its over-counting of allomorphs and syncretic morphemes, the random pair sampling method that MC uses is not independent of the set of analyses being evaluated. If two algorithms predict different morpheme distributions, the sampling method will find different numbers of word pairs. We substantiate our claim that the MC metric lacks robustness in Section 4 where we empirically compare it to the *EMMA* metric.

4 Experimental Evaluation

To experimentally evaluate our newly proposed *EMMA* metric, and to quantitatively compare the *EMMA* and MC metrics, we have evaluated results of 93 system-language pairs from Morpho Challenge 2007, 2008, and 2009.[1] The evaluation comprised three algorithms by Bernhard (2007) and Bernhard (2009), one algorithm by Can and Manandhar (2009), the MC baseline algorithm *Morfessor* by Creutz (2006), *UNGRADE* by Golenia et al. (2009), two algorithms by Lavallee and Langlais (2009), one algorithm by Lignos et al. (2009), five *ParaMor* versions by Monson et al. (2008) and Monson et al. (2009), three *Promodes* versions by Spiegler et al. (2009) and one algorithm by Tchoukalov et al. (2009). We ran these algorithms over six data sets available from the MC competition: Arabic (vowelized and non-vowelized), English, Finnish, German, and Turkish. We then scored the system outputs using both *EMMA* and the MC metric against an answer key provided by MC. In Sections 2 and 3.3 we have already commented on the *linguistic characteristics* of both metrics. In this section, we concentrate on their *computational performance*.

Both the *EMMA* and MC metrics are *readily computable*: Both are freely available[2] and they each take less than two minutes to run on the average desktop machines we have used. In terms of *interpretability*, *EMMA* not only returns the performance as precision, recall and f-measure as MC does, but also provides predicted analyses where mapped morphemes are replaced by answer key morphemes. This information is helpful when judging results qualitatively since it exposes tangible algorithmic characteristics. In Table 1 we present the algorithms with the highest MC and *EMMA* scores for each language. For all languages, the *EMMA* and MC metrics place different algorithms highest. One reason for the significantly different rankings that the two metrics provide may be the *sampling of random pairs* that MC uses. Depending on the distribution of predicted morphemes across words, the number of random pairs, which is used for calculating the precision, may vary. For instance, on vowelized Arabic, *Promodes 1* is evaluated over a sample of 100 pairs where MC selected just 47 pairs for *ParaMor Mimic*.

[1] Detailed results can be found in Spiegler (2010).

[2] *EMMA* may be downloaded from `http://www.cs.bris.ac.uk/Research/MachineLearning/Morphology/Resources/`

Language	Algorithm and year of participation in MC		MC evaluation metric			*EMMA* evaluation metric		
			Pr.	Re.	F1	Pr.	Re.	F1
Arabic (nv)	Promodes 2	2009	0.7789	0.3980	**0.5268**	0.5356	0.2444	0.3356
	Ungrade	2009	0.7971	0.1603	0.2670	0.7017	0.2490	**0.3675**
Arabic (vw)	Promodes 2	2009	0.5946	0.6017	**0.5982**	0.4051	0.3199	0.3575
	Promodes 1	2009	0.7381	0.3477	0.4727	0.5588	0.3281	**0.4135**
English	Bernhard 1	2007	0.7850	0.5763	**0.6647**	0.8029	0.7460	0.7734
	Lignos	2009	0.7446	0.4716	0.5775	0.9146	0.6747	**0.7766**
Finnish	ParaMorPlusMorfessor	2008	0.5928	0.5675	**0.5798**	0.2271	0.3428	0.2732
	Lavallee rali-cof	2009	0.6731	0.3563	0.4659	0.5061	0.4065	**0.4509**
German	ParaMorPlusMorfessor	2008	0.5562	0.6077	**0.5808**	0.3633	0.4948	0.4190
	Morfessor	2009	0.6528	0.3818	0.4818	0.7311	0.5556	**0.6314**
Turkish	ParaMorPlusMorfessor	2008	0.6779	0.5732	**0.6212**	0.3476	0.4315	0.3851
	Morfessor	2009	0.7894	0.3330	0.4684	0.5901	0.3703	**0.4550**

Table 1: Best performing algorithms with MC and *EMMA* evaluation metric.

Algorithm and year of participation in MC		MC evaluation metric			*EMMA* evaluation metric		
		Pr.	Re.	F1	Pr.	Re.	F1
Morfessor	2009	0.8143	0.2788	0.4154	0.4751	0.3472	0.4012
ParaMor	2008	0.4111	0.4337	0.4221	0.4322	0.3770	0.4027
ParaMorPlusMorfessor	2008	0.5928	0.5675	**0.5798**	0.2271	0.3428	**0.2732**
Paramor Morfessor Union	2009	0.4374	0.5676	0.4941	0.3878	0.4530	0.4178

Table 3: Gaming MC with ambiguity hijacking on Finnish.

Looking at any particular algorithm-language pair, the *EMMA* and MC scores differ considerably and respective raw scores are not directly comparable. More interesting is the extent to which both metrics *correlate* with real NL tasks. Table 2 lists the Spearman rank correlation coefficient for algorithms from MC 2009 on English, Finnish and German comparing rankings of f-measure results returned by either MC or *EMMA* against rankings using the mean average precision (MAP) of an information retrieval (IR) task.[3] All MAP scores are taken from Kurimo et al. (2009). Although both metrics positively correlate with the IR results; *EMMA*'s correlation is clearly stronger across all three languages.

To test the *robustness* of the *EMMA* and MC metrics, we performed two experiments where we intentionally attempt to game the metrics – *ambiguity hijacking* and *shared morpheme padding*. In both experiments, the MC metric showed vulnerability. Ambiguity hijacking results for Finnish appear in Table 3, other languages perform similarly. Using both metrics, we scored the Finnish analyses that were proposed by a) the *Morfessor* algorithm alone, b) *ParaMor* alone, and c) two ways of combining ParaMor and Morfessor: *ParaMorPlusMorfessor* simply lists the ParaMor and Morfessor analyses as alternatives – as if each word were ambiguous between a ParaMor and a Morfessor analysis; *ParaMorMorfessorUnion*, on the other hand, combines the morpheme boundary predictions of *ParaMor* and *Morfessor* into a single analysis. The *ParaMorPlusMorfessor* system games the ambiguity mechanism of the MC metric, achieving an f-measure higher than that of any of the three other algorithms. *EMMA*, however, correctly discovers that the analyses proposed by *ParaMorPlusMorfessor* lie farther from an isomorphism to the the answer key than do the unified analyses of *ParaMorMorfessorUnion*.

In Table 4 we show a second way of gaming the MC metric – *shared morpheme padding*. We add the same unique bogus morpheme to each proposed analysis of every word for all systems.

[3] Detailed results can be found in Spiegler (2010).

Language	MC evaluation			EMMA evaluation		
	Precision	Recall	F-measure	Precision	Recall	F-measure
Arabic (nv)	0.91±0.02	**10.83 ± 8.33**	**7.20±5.10**	0.91±0.05	1.30±0.07	1.20±0.05
Arabic (vw)	0.85±0.04	**11.17±8.81**	**7.13±5.23**	0.89±0.07	1.21±0.06	1.12±0.05
English	**0.36±0.08**	**2.02±0.66**	0.63±0.10	0.73±0.15	1.05±0.08	0.86±0.12
Finnish	0.57±0.08	**3.07±2.47**	1.19±0.68	0.87±0.19	1.12±0.10	0.99±0.14
German	**0.43±0.08**	**2.90±1.45**	0.84±0.16	0.80±0.17	1.09±0.08	0.94±0.11
Turkish	0.58±0.09	**2.95±1.65**	1.19±0.37	0.85±0.08	1.07±0.04	0.97±0.05

Table 4: Gaming MC with shared morpheme padding: Average and standard deviations of the ratio of padded to original scores.

Padding analyses with a shared morpheme significantly increases the recall scores of the MC metric. We summarize our experimental results by calculating, for each language-algorithm pair, the ratio of the score for the padded analyses as compared to that of the original, unpadded analyses. Table 4 reports average and standard deviation of the ratios across all systems for each language. In Arabic (nv. and vw.), the recall increases by 10.83 and 11.17 times, which leads to an inflation of f-measure by 7.20 and 7.13 times – this is a direct result of the soft nature of the MC isomorphism. In contrast, *EMMA*'s recall scores increase much less than MC's do, and *EMMA*'s precision scores decrease proportionately. A small change to the set of proposed analyses does not lead to a huge difference in f-measure – characteristic of a more *robust* metric.

5 Conclusion

This paper has proposed, *EMMA*, a novel evaluation metric for the assessment of the quality of a set of morphological analyses. *EMMA*'s:

1. Coverage of the major morphological phenomena,
2. Correlation with performance on natural language processing tasks, and
3. Computational robustness

all recommend the the metric as a strong and useful measure – particularly when evaluating unsupervised morphology analysis systems which, lacking access to labeled training data, are uninformed of the labeling standard used in the answer key.

	Correlation with IR	
	IR vs. MC	IR vs. *EMMA*
English	0.466	**0.608**
Finnish	0.681	**0.759**
German	0.379	**0.637**

Table 2: Spearman rank correlation coefficient of metrics vs. Information Retrieval (IR).

Acknowledgements

We would like to acknowledge various fruitful discussions with Aram Harrow, Alex Popa, Tilo Burghardt and Peter Flach. The work was partially sponsored by EPSRC grant EP/E010857/1 *Learning the morphology of complex synthetic languages*, as well as by NSF Grant #IIS-0811745 and DOD/NGIA grant #HM1582-08-1-0038.

References

Arısoy, Ebru, Doğan Can, Sıddıka Parlak, Haşim Sak, and Murat Saraçlar. 2009. Turkish Broadcast News Transcription and Retrieval. *IEEE Trans. on Audio, Speech and Lang. Proc.*

Beesley, Kenneth R. and Lauri Karttunen. 2003. *Finite State Morphology*. University of Chicago Press.

Berkelaar, Michel, Kjell Eikland, and Peter Notebaert. 2004. Open source (mixed-integer) linear programming system, version 5.1.0.0. `http://lpsolve.sourceforge.net/`.

Bernhard, Delphine. 2007. Simple morpheme labelling in unsupervised morpheme analysis. *Working Notes, CLEF 2007 Workshop*.

Bernhard, Delphine. 2009. Morphonet: Exploring the use of community structure for unsupervised morpheme analysis. *Working Notes, CLEF 2009 Workshop*.

Can, Burcu and Suresh Manandhar. 2009. Unsupervised learning of morphology by using syntactic categories. *Working Notes, CLEF 2009 Workshop*.

Creutz, Mathias. 2006. *Induction of the Morphology of Natural Language: Unsupervised Morpheme Segmentation with Application to Automatic Speech Recognition*. Ph.D. thesis, Helsinki University of Technology, Espoo, Finland.

Goldsmith, John. 2001. Unsupervised learning of the morphology of a natural language. *Comp. Ling.*, 27.

Golénia, Bruno, Sebastian Spiegler, and Peter Flach. 2009. Ungrade: unsupervised graph decomposition. *Working Notes, CLEF 2009 Workshop*.

Hafer, M. A. and S. F. Weiss. 1974. Word segmentation by letter successor varieties. *Inf. Storage and Retrieval*, 10.

Kurimo, Mikko, Mathias Creutz, Matti Varjokallio, Ebru Arisoy, Murat Saraclar. 2006. Unsupervised segmentation of words into morphemes - Morpho Challenge 2005. *Interspeech*.

Kurimo, Mikko, Mathias Creutz, and Ville Turunen. 2007. Overview of morpho challenge in CLEF 2007. *Working Notes, CLEF 2007 Workshop*.

Kurimo, Mikko and Ville Turunen. 2008. Unsupervised Morpheme Analysis Evaluation by IR experiments – Morpho Challenge 2008. *Working Notes, CLEF 2008 Workshop*.

Kurimo, Mikko, Sami Virpioja, and Ville T. Turunen. 2009. Overview and results of morpho challenge 2009. *Working Notes, CLEF 2009 Workshop*.

Lavallee, Jean-Francois and Philippe Langlais. 2009. Morphological Acquisition by Formal Analogy. *Working Notes, CLEF 2009 Workshop*.

Lavie, Alon, Erik Peterson, Katharina Probst, Shuly Wintner, Yaniv Eytani. 2004. Rapid Prototyping of a Transfer-based Hebrew-to-English Machine Translation System. *Proc. of TMI-2004*.

Lignos, Constantine, Erwin Chan, Mitchell P. Marcus, and Charles Yang. 2009. A rule-based unsupervised morphology learning framework. *Working Notes, CLEF 2009 Workshop*.

McNamee, Paul, Charles Nicholas, and James Mayfield. 2008. Don't Have a Stemmer? Be Un+concern+ed *Proc. of the 31st Anual International ACM SIGIR Conference 20-24 July 2008*.

Monson, Christian, Jaime Carbonell, Alon Lavie, and Lori Levin. 2008. Paramor and morpho challenge 2008. *Working Notes, CLEF 2008 Workshop*.

Monson, Christian, Kristy Hollingshead, and Brian Roark. 2009. Probabilistic paramor. *Working Notes, CLEF 2009 Workshop*.

Oflazer, Kemal, and İlknur Durgar El-Kahlout. 2007. Different Representational Units in English-to-Turkish Statistical Machine Translation. *Proc. of Statistical Machine Translation Workshop at ACL 2007*.

Poon, Hoifung, Colin Cherry and Kristina Toutanova 2009. Unsupervised Morphological Segmentation with Log-Linear Models. *Proc. of ACL*.

Roark, Brian and Richard Sproat. 2007. *Computational Approaches to Morphology and Syntax*. Oxford Univ. Press.

Schone, Patrick and Daniel Jurafsky. 2001. Knowlege-free induction of inflectional morphologies. *Proc. of NAACL-2001*.

Snover, Matthew G., Gaja E. Jarosz and Michael R. Brent. 2002. Unsupervised Learning of Morphology Using a Novel Directed Search Algorithm: Taking the First Step. *Proc. of the ACL-02 SIGPHON Workshop*.

Snyder, Benjamin and Regina Barzilay. 2008. Unsupervised Multilingual Learning for Morphological Segmentation. *Proc. of ACL-08: HLT*.

Spencer, Andrew and Arnold M. Zwicky, editors. 2001. *The Handbook of Morphology*. Wiley-Blackwell.

Spiegler, Sebastian, Bruno Golénia, and Peter A. Flach. 2009. Promodes: A probabilistic generative model for word decomposition. *Working Notes, CLEF 2009 Workshop*.

Spiegler, Sebastian. 2010. *EMMA: A Novel Metric for Morphological Analysis - Experimental Results in Detail.* Computer Science Department, University of Bristol, U.K.

Tchoukalov, Tzvetan, Christian Monson, and Brian Roark. 2009. Multiple sequence alignment for morphology induction. *Working Notes, CLEF 2009 Workshop*.

Wicentowski, Richard 2002. *Modeling and Learning Multilingual Inflectional Morphology in a Minimally Supervised Framework.* Ph.D. thesis, The Johns Hopkins University, Baltimore, Maryland, U.S.A.

Modeling Socio-Cultural Phenomena in Discourse

Tomek Strzalkowski[1,2], **George Aaron Broadwell**[1], **Jennifer Stromer-Galley**[1],
Samira Shaikh[1], **Sarah Taylor**[3] **and Nick Webb**[1]
[1]ILS Institute, University at Albany, SUNY
[2]IPI, Polish Academy of Sciences
[3]Lockheed Martin Corporation
tomek@albany.edu

Abstract

In this paper, we describe a novel approach to computational modeling and understanding of social and cultural phenomena in multi-party dialogues. We developed a two-tier approach in which we first detect and classify certain social language uses, including topic control, disagreement, and involvement, that serve as first order models from which presence the higher level social constructs such as leadership, may be inferred.

1. Introduction

We investigate the language dynamics in small group interactions across various settings. Our focus in this paper is on English online chat conversations; however, the models we are developing are more universal and applicable to other conversational situations: informal face-to-face interactions, formal meetings, moderated discussions, as well as interactions conducted in languages other than English, e.g., Urdu and Mandarin.

Multi-party online conversations are particularly interesting because they become a pervasive form of communication within virtual communities, ubiquitous across all age groups. In particular, a great amount of communication online occurs in virtual chat-rooms, typically conducted using a highly informal text dialect. At the same time, the reduced-cue environment of online interaction necessitates more explicit linguistic devices to convey social and cultural nuances than is typical in face-to-face or even voice conversations.

Our objective is to develop computational models of how certain social phenomena such as leadership, power, and conflict are signaled and reflected in language through the choice of lexical, syntactic, semantic and conversational forms by discourse participants. In this paper we report the results of an initial phase of our work during which we constructed a prototype system called DSARMD-1 (Detecting Social Actions and Roles in Multiparty Dialogue). Given a representative segment of multiparty task-oriented dialogue, DSARMD-1 automatically classifies all discourse participants by the degree to which they deploy selected *social language uses*, such as topic control, task control, involvement, and disagreement. These are the mid-level social phenomena, which are deployed by discourse participants in order to achieve or assert higher-level *social constructs*, including leadership. In this work we adopted a two-tier empirical approach where social language uses are modeled through *observable* linguistic features that can be automatically extracted from dialogue. The high-level social constructs are then inferred from a combination of language uses attributed to each discourse participant; for example, a high degree of influence and a high degree of involvement by the same person may indicate a leadership role. In this paper we limit our discussion to the first tier only: how to effectively model and classify social language uses in multi-party dialogue.

2. Related Research

Issues related to linguistic manifestation of social phenomena have not been systematically researched before in computational linguistics; indeed, most of the effort thus far was directed towards the communicative dimension of discourse. While the Speech Acts theory (Austin, 1962; Searle, 1969) provides a generalized framework for multiple levels of discourse analysis (locution, illocution and perlocution), most current approaches to dialogue focus on information content and structural components (Blaylock, 2002; Carberry & Lambert, 1999; Stolcke, et al., 2000) in dialogue; few take into account the effects that speech acts may have upon the social

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1038–1046,
Beijing, August 2010

roles of discourse participants. Also relevant is research on modeling sequences of dialogue acts – to predict the next one (Samuel et al. 1998; Ji & Bilmes, 2006 inter alia) – or to map them onto subsequences or "dialogue games" (Carlson 1983; Levin et al., 1998), which are attempts to formalize participants' roles in conversation (e.g., Linell, 1990; Poesio & Mikheev, 1998; Field et al., 2008).

There is a body of literature in anthropology, linguistics, sociology, and communication on the relationship between language and power, as well as other social phenomena, e.g., conflict, leadership; however, existing approaches typically look at language use in situations where the social relationships are known, rather than using language predictively. For example, conversational analysis (Sacks et al., 1974) is concerned with the structure of interaction: turn-taking, when interruptions occur, how repairs are signaled, but not what they reveal about the speakers. Research in anthropology and communication has concentrated on how certain social norms and behaviors may be reflected in language (e.g., Scollon and Scollon, 2001; Agar, 1994) with few systematic studies attempting to explore the reverse, i.e., what the linguistic phenomena tell us about social norms and behaviors.

3. Data & Annotation

Our initial focus has been on on-line chat dialogues. While chat data is plentiful on-line, its adaptation for research purposes presents a number of challenges that include users' privacy issues on the one hand, and their complete anonymity on the other. Furthermore, most data that may be obtained from public chat-rooms is of limited value for the type of modeling tasks we are interested in due to its high-level of noise, lack of focus, and rapidly shifting, chaotic nature, which makes any longitudinal studies virtually impossible. To derive complex models of conversational behavior, we need the interaction to be reasonably focused on a task and/or social objectives within a group.

Few data collections exist covering multiparty dialogue, and even fewer with on-line chat. Moreover, the few collections that exist were built primarily for the purpose of training dialogue act tagging and similar linguistic phenomena; few if any of these corpora are suitable for deriving pragmatic models of conversation, including socio-linguistic phenomena. Existing resources include a multi-person meeting corpus ICSI-MRDA and the AMI Meeting Corpus (Carletta, 2007), which contains 100 hours of meetings captured using synchronized recording devices. Still, all of these resources look at spoken language rather than on-line chat. There is a parallel interest in the online chat environment, although the development of useful resources has progressed less. Some corpora exist such as the NPS Internet chat corpus (Forsyth and Martell, 2007), which has been hand-anonymized and labeled with part-of-speech tags and dialogue act labels. The StrikeCom corpus (Twitchell et al., 2007) consists of 32 multi-person chat dialogues between players of a strategic game, where in 50% of the dialogues one participant has been asked to behave 'deceptively'.

It is thus more typical that those interested in the study of Internet chat compile their own corpus on an as needed basis, e.g., Wu et al. (2002), Khan et al. (2002), Kim et al. (2007).

Driven by the need to obtain a suitable dataset we designed a series of experiments in which recruited subjects were invited to participate in a series of on-line chat sessions in a specially designed secure chat-room. The experiments were carefully designed around topics, tasks, and games for the participants to engage in so that appropriate types of behavior, e.g., disagreement, power play, persuasion, etc. may emerge spontaneously. These experiments and the resulting corpus have been described elsewhere (Shaikh et al., 2010b), and we refer the reader to this source. Ultimately a corpus of 50 hours of English chat dialogue was collected comprising more than 20,000 turns and 120,000 words. In addition we also assembled a corpus of 20 hours of Urdu chat.

A subset of English language dataset has been annotated at four levels: communication links, dialogue acts, local topics and meso-topics (which are essentially the most persistent local topics). Although full details of these annotations are impossible to explain within the scope of this article, we briefly describe them below. Annotated datasets were used to develop and train automatic modules that detect and classify social uses of language in discourse. It is important to note that the annotation has been developed to support the objectives of our project and does not necessarily conform to other similar annotation systems used in the past.

- *Communicative links*. In a multi-party dialogue an utterance may be directed towards a specific participant, a subgroup of participants or to everyone.
- *Dialogue Acts*. We developed a hierarchy of 15 dialogue acts for annotating the functional aspect of the utterance in discussion. The tagset we adopted is based on DAMSL (Allen & Core, 1997) and SWBD (Jurafsky et al., 1997), but compressed to 15 tags tuned significantly towards dialogue pragmatics and away from more surface characteristics of utterances (Shaikh et al., 2010a).
- *Local topics*. Local topics are defined as nouns or noun phrases introduced into discourse that are subsequently mentioned again via repetition, synonym, or pronoun.
- *Topic reference polarity*. Some topics, which we call *meso-topics*, persist through a number of turns in conversation. A selection of meso-topics is closely associated with the task in which the discourse participants are engaged. Meso-topics can be distinguished from the local topics because the speakers often make polarized statements about them.

4. Socio-linguistic Phenomena

We are interested in modeling the social phenomena of Leadership and Power in discourse. These high-level phenomena (or Social Roles, *SR*) will be detected and attributed to discourse participants based on their deployment of selected *Language Uses* (LU) in multi-party dialogue. Language Uses are mid-level socio-linguistic devices that link linguistic components deployed in discourse (from lexical to pragmatic) to social constructs obtaining for and between the participants. The language uses that we are currently studying are *Agenda Control*, *Disagreement*, and *Involvement* (Broadwell et al., 2010).

Our research so far is focused on the analysis of English-language synchronous chat, and we are looking for correlations between various metrics that can be used to detect LU in multiparty dialogue. We expect that some of these correlations may be culturally specific or language-specific, as we move into the analysis of Urdu and Mandarin discourse in the next phase of this project.

4.1 Agenda Control in Dialogue

Agenda Control is defined as efforts by a member or members of the group to advance the group's task or goal. This is a complex LU that we will model along two dimensions: (1) *Topic Control* and (2) *Task Control*. Topic Control refers to attempts by any discourse participants to impose the topic of conversation. Task Control, on the other hand, is an effort by some members of the group to define the group's project or goal and/or steer the group towards that goal. We believe that both behaviors can be detected using scalar measures per participant based on certain linguistic features of their utterances.

For example, one hypothesis is that topic control is indicated by the rate of *local topic introductions* (LTI) per participant (Givon, 1983). Local topics may be defined quite simply as noun phrases introduced into discourse, which are subsequently mentioned again via repetition, synonym, pronoun, or other form of co-reference. Thus, one measure of topic control is the number of local topics introduced by each participant as percentage of all local topics in a discourse.

Using an LTI index we can construct assertions about topic control in a discourse. For example, suppose the following information is discovered about the speaker LE in a multi-party discussion *dialogue-1*[1] where 90 local topics are identified:

1. LE introduces 23/90 (25.6%) of local topics in this dialogue.
2. The mean rate of local topic introductions is this dialogue is 14.29%, and standard deviation is 8.01.
3. LE is in the top quintile of participants for introducing new local topics

We can now claim the following, with a degree of confidence (to be determined):

$$TopicControl_{LTI}\,(LE,\ 5,\ dialogue\text{-}1)$$

We read this as follows: *speaker LE exerts the highest degree of topic control in dialogue-1.* Of course, LTI is just one source of evidence and we developed other metrics to complement it. We mention three of them here:

[1] Dialogue-1 refers to an actual dataset of 90-minute chat among 7 participants, covering approximately 700 turns. The task is to select a candidate for a job given a set of resumes.

- *SMT Index*. This is a measure of topic control suggested in (Givon, 1983) and it is based on subsequent mentions of already introduced local topics. Speakers who introduce topics that are discussed at length by the group tend to control the topic of the discussion. The *subsequent mentions of local topics* (SMT) index calculates the percentage of second and subsequent references to the local topics, by repetition, synonym, or pronoun, relative to the speakers who introduced them.
- *Cite Score*. This index measures the extent to which other participants discuss topics introduced by that speaker. The difference between SMT and CiteScore is that the latter reflect to what degree a speaker's efforts to control the topic are assented to by other participants in a conversation.
- *TL Index* (TL). This index stipulates that more influential speakers take longer turns than those who are less influential. The TL index is defined as the average number of words per turn for each speaker. Turn length also reflects the extent to which other participants are willing to 'yield the floor' in conversation.

Like LTI, all the above indices are mapped into a degree of topic control, based on quintiles in normal distribution (Table 1).

	LTI	**SMT**	**CS**	**TL**	AVG
LE	5	5	5	5	5.00
JR	4	4	4	3	3.75
KI	4	3	3	1	2.75
KN	3	5	4	4	4.00
KA	2	2	2	4	2.50
CS	2	2	2	2	2.00
JY	1	1	1	2	1.25

Table 1: Topic Control distribution in dialogue-1. Each row represents a speaker in the group (LE, JR, etc.). Columns show indices used, with degrees per speaker on 5-point scale based on quintiles in normal distribution, and the average value.

Ideally, all the above indices (and others yet to be defined) should predict the same outcome, i.e., for each dialogue participant they should assign the same degree of topic control, relative to other speakers. This is not always the case, and where the indices divert in their predictions, our level of confidence in the generated claims decreases. We are currently working on how these different metrics correlate to each other and how they should be weighted to maximize accuracy of making Topic Control claims. Nonetheless, we can already output a Topic Control map (shown in Table 1) that captures a sense of internal social dynamics within the group.

The other aspect of Agenda Control phenomenon is Task Control. It is defined as an effort to determine the group's goal and/or steer the group towards that goal. Unlike Topic Control, which is imposed by influencing the subject of conversation, Task Control is gained by directing other participants to perform certain tasks or accept certain opinions. Consequently, Task Control is detected by observing the usage of certain dialogue acts, including Action-Directive, Agree-Accept, Disagree-Reject, and related categories. Here again, we define several indices that allow us to compute a degree of Task Control in dialogue for each participant:

- *Directive Index* (DI). The participant who directs others is attempting to control the course of the task that the group is performing. We count the number of directives, i.e., utterances classified as Action-Directive, made by each participant as a percentage of all directives in discourse.
- *Directed Topic Shift Index* (DTSI). When a participant who controls the task offers a directive on the task, then the topic of conversation shifts. In order to detect this condition, we calculate the ratio of coincidence of directive dialogue acts by each participant with topic shifts following them.
- *Process Management index* (PMI). Another measure of Task Control is the proportion of turns each participant has that explicitly address the problem solving process. This includes utterances that involve coordinating the activities of the participants, planning the order of activities, etc. These fall into the category of Task (or Process) Management in most DA tagging systems.
- *Process Management Success Index* (PMSI). This index measures the degree of success by each speaker at controlling the task. A credit is given to the speaker whose suggested curse of action is supported by other speakers for each response that supports the suggestion. Conversely, a credit is taken away for each response that rejects or

qualifies the suggestion. PMSI is computed as distribution of task management credits among the participants over all dialogue utterances classified as Task/Process Management. [2]

As an example, let's consider the following information computed for the PMI index over *dialogue-1*:

1. *Dialogue-1* contains 246 utterances classified as Task/Process Management rather than doing the task.
2. Speaker KI makes 65 of these utterances for a PMI of 26.4%.
3. Mean PMI for participants is 14.3%; 80th percentile is >21.2%. PMI for KI is in the top quintile for all participants.

Based on this evidence we may claim (with yet to be determined confidence) that:

TaskControl$_{PMI}$(*KI, 5, dialogue-1*)

This may be read as follows: *speaker KI exerts the highest degree of Task Control in dialogue-1.* We note that Task Control and Topic Control do not coincide in this discourse, at least based on the PMI index. Other index values for Task Control may be computed and tabulated in a way similar to LTI in Table 1. We omit these here due to space limitations.

4.2 Disagreement in Dialogue

Disagreement is another language use that correlates with speaker's power and leadership. There are two ways in which disagreement is realized: *expressive disagreement* and *topical disagreement* (Stromer-Galley, 2007; Price, 2002). Both can be detected using scalar measures applied to subsets of participants, typically any two participants. In addition, we can also measure for each participant the rate with which he or she generates disagreement (with any and all other speakers). *Expressive Disagreement* is normally understood at the level of dialogue acts, i.e., when discourse participants make explicit utterances of disagreement, disapproval, or rejection in response to a prior speaker's utterance. Here is an example (KI and KA are two speakers in a multiparty dialogue in which participants discuss candidates for a youth counselor job):

KA: *CARLA... women are always better with kids*
KI: *That's not true!*
KI: *Men can be good with kids too*

While such exchanges are vivid examples of expressive disagreement, we are interested in more sustained phenomenon where two speakers repeatedly disagree, thus revealing a social relationship between them. Therefore, one measure of Expressive Disagreement that we consider is the number of Disagree-Reject dialogue acts between any two speakers as a percentage of all utterances exchanged between these two speakers. This becomes a basis for the *Disagree-Reject Index* (DRX). In dialogue-1 we have:

1. Speakers KI and KA have 47 turns between them. Among these there are 8 turns classified as Disagree-Reject, for the DRX of 15.7%.
2. The mean DRX for speakers who make any Disagree-Reject utterances is 9.5%. The pair of speakers KI-KA is in the top quintile (>13.6%).

Based on this evidence we can conclude the following:

ExpDisagreement$_{DRX}$ (*KI,KA, 5, dialogue-1*)

which may be read as follows: *speakers KI and KA have the highest level of expressive disagreement in dialogue-1.* This measure is complemented by a *Cumulative Disagreement Index* (CDX), which is computed for each speaker as a percentage of all Disagree-Reject utterances in the discourse that are made by this speaker. Unlike DRX, which is computed for pairs of speakers, the CDX values are assigned to each group participant and indicate the degree of disagreement that each person generates.

While Expressive Disagreement is based on the use of more overt linguistic devices, *Topical Disagreement* is defined as a difference in referential *valence* in utterances (statements, opinions, questions, etc.) made on a topic. Referential valence of an utterance is determined by the type of statement made about the topic in question, which can be positive (+), negative (–), or neutral (0). A positive statement is one in favor of (*express advocacy*) or in support of (*supporting information*) the topic being discussed. A negative statement is one that is against or negative on

[2] The exact structure of the credit function is still being determined experimentally. For example, more credit may be given to first supporting response and less for subsequent responses; more credit may be given for unprompted suggestions than for those that were responding to questions from others.

the topic being discussed. A neutral statement is one that does not indicate the speaker's position on the topic. Here is an example of opposing polarity statements about the same topic in discourse:

Sp-1: *I like that he mentions "Volunteerism and Leadership"*
Sp-2: *but if they're looking for someone who is experienced then I'd cross him off*

Detecting topical disagreement in discourse is more complicated because its strength may vary from one topic in a conversation to the next. A reasonable approach is thus to measure the degree of disagreement between two speakers on one topic first, and then extrapolate over the entire discourse. Accordingly, our measure of topical disagreement is valuation differential between any two speakers as expressed in their utterances about a topic. Here, the topic (or an "issue") is understood more narrowly than the local topic defined in the previous section (as used in Topic Control, for example), and may be assumed to cover only the most persistent local topics, i.e., topics with the largest number of references in dialogue, or what we call the *meso-topics*. For example, in a discussion of job applicants, each of the applicants becomes a meso-topic, and there may be additional meso-topics present, such as qualifications required, etc.

The resulting *Topical Disagreement Metric* (TDM) captures the degree to which any two speakers advocate the opposite sides of a meso-topic. TDM is computed as an average of *P*-valuation differential for one speaker (advocating *for* a meso-topic) and (–*P*)-valuation differential for the other speaker (advocating *against* the meso-topic).

Using TDM we can construct claims related to disagreement in a given multiparty dialogue of sufficient duration (exactly what constitutes a sufficient duration is still being researched). Below is an example based on a 90-minute chat dialogue-1 about several job candidates for a youth counselor. The discussion involved 7 participants, including KI and KA. Topical disagreement is measured on 5 points scale (corresponding to quintiles in normal distribution):

$TpDisAgree_{TDM}$(*KI,KA,"Carla",4,dialogue-1*)

This may be read as follows: speakers KI and KA *topically disagree to degree 4* on topic [job candidate] "Carla" in dialogue-1. In order to calculate this we compute the value of TDM index between these two speakers. We find that KA makes 30% of all positive utterances made by anyone about Carla (40), while KI makes 45% of all negative utterances against Carla. This places these two speakers in the top quintiles in the "for Carla" polarity distribution and "against Carla" distribution, respectively. Taking into account any opposing polarity statements made by KA against Carla and any statements made by KI for Carla, we calculate the level of topical disagreement between KA and KI to be 4 on the 1-5 scale.

TDM allows us to compute topical disagreement between any two speakers in a discourse, which may also be represented in a 2-dimensional table revealing another interesting aspect of internal group dynamics.

4.3 Involvement in Dialogue

The third type of social language use that we discuss in this paper is Involvement. Involvement is defined as a degree of engagement or participation in the discussion of a group. It is an important element of leadership, although its importance is expected to differ between cultures; in Western cultures, high involvement and influence (topic control) often correlates with group leadership.

In order to measure Involvement we designed several indices based on turn characteristics for each speaker. Four of the indices are briefly explained below:

- The *NP index* (NPI) is a measure of gross informational content contributed by each speaker in discourse. *NPI* counts the ratio of third-person nouns and pronouns used by a speaker to the total number of nouns and pronouns in the discourse.
- The *Turn index* (*TI*) is a measure of *interactional frequency*; it counts the ratio of turns per participant to the total number of turns in the discourse.
- The *Topic Chain Index* (*TCI*) counts the degree to which participants discuss of the most persistent topics. In order to calculate TCI values, we define a *topic chains* for all local topics. We compute frequency of mentions of these longest topics for each participant.
- The *Allotopicality Index* (*ATP*) counts the number of mentions of local topics that were introduced by other participants. An

ATP value is the proportion of a speaker's allotopical mentions, i.e., excluding "self-citations", to all allotopical mentions in a discourse.

As an example, we may consider the following situation in *dialogue-1*:

1. *Dialogue-1* contains 796 third person nouns and pronouns, excluding mentions of participants' names.
2. Speaker JR uses 180 nouns and pronouns for an NPI of 22.6%.
3. The median NPI is 14.3%; JR are in the upper quintile of participants (> 19.9%).

From the above evidence we can draw the following claim:

$Involvement_{NPI}(JR, 5, dialogue\text{-}1)$

This may be read as: *speaker JR is the most involved participant in dialogue-1.*

As with other language uses, multiple indices for Involvement can be combined into a 2-dimensional map capturing the group internal dynamics.

5. Implementation & Evaluation

We developed a prototype automated DSARMD system that comprises a series of modules that create automated annotation of the source dialogue for all the language elements discussed above, including communicative links, dialogue acts, local/meso topics, and polarity. Automatically annotated dialogue is then used to generate language use degree claims. In order to evaluate accuracy of the automated process we conducted a preliminary evaluation comparing the LU claims generated from automatically annotated data to the claims generated from manually coded dialogues. Below we briefly describe the methodology and metrics used.

Each language use is asserted per a participant in a discourse (or per each pair of participants, e.g., for Disagreement) on a 5-point "strength" scale. This can be represented as an ordered sequence $LU_X(d_1, d_2, ... d_n)$, where LU is the language use being asserted, *X* is the index used, d_i is the degree of LU attributed to speaker *i*. This assignment is therefore a 5-way classification of all discourse participants and its correctness is measured by dividing the number of correct assignments by the total number of elements to be classified, which gives the micro-averaged precision. The accuracy metric is computed with several variants as follows:

1. *Strict mapping*: each complete match is counted as 1; all mismatches are counted as 0. For example, the outputs LU_X (5,4,3,2,1) and LU_X (4,5,3,1,1) produce two exact matches (for the third and the last speaker) for a precision of 0.4.
2. *Weighted mapping*: since each degree value d_i in $LU_X(d_1, d_2, ... d_n)$ represents a quintile in normal distribution, we consider the position of the value within the quintile. If two mismatched values are less than ½ quintile apart we assign a partial credit (currently 0.5).
3. *Highest – Rest*: we measure accuracy with which the highest LU degree (but not necessarily the same degree) is assigned to the right speaker vs. any other score. This results in binary classification of scores. The sequences in (1) produce 0.6 match score.
4. *High – Low*: An alternative binary classification where scores 5 and 4 are considered High, while the remaining scores are considered Low. Under this metric, the sequences in (1) match with 100% precision.

The process of automatic assignment of language uses derived from automatically processed dialogues was evaluated against the control set of assignments based on human-annotated data. In order to obtain a reliable "ground truth", each test dialogue was annotated by at least three human coders (linguistics and communication graduate students, trained). Since human annotation was done at the linguistic component level, a strict inter-annotator agreement was not required; instead, we were interested whether in each case a comparable statistical distribution of the corresponding LU index was obtained. Annotations that produced index distributions dissimilar from the majority were eliminated.

Automated dialogue processing involved the following modules:

- *Local topics detection* identifies first mentions by tracking occurrences of noun phrases. Subsequent mentions are identified using fairly simple pronoun resolution (based mostly on lexical features), with Wordnet used to identify synonyms, etc.
- *Meso-topics* are identified as longest-chain local topics. Their *polarity* is assessed at the utterance level by noting presence of positive or negative cue words and phrases.
- *Dialogue acts* are tagged based on presence

of certain cue phrases derived from a training corpus (Webb et al., 2008).

- *Communicative links* are mapped by computing inter-utterance similarity based on n-gram overlap.

Preliminary evaluation results are shown in Tables 3-5 with average performance over 3 chat sessions (approx 4.5 hours) involving three groups of speakers and different tasks (job candidates, political issues). Topic Control and Involvement tables show average accuracy per index. For example, the LTI index, computed over automatically extracted local topics, produces Topic Control assignments with the average precision of 80% when compared to assignments derived from human-annotated data using the strict accuracy metric. However, automated prediction of Involvement based on NPI index is far less reliable, although we can still pick the most involved speaker with 67% accuracy. We omit the indices based on turn length (TL) and turn count (TI) because their values are trivially computed. At this time we do not combine indices into a single LU prediction. Additional experiments are needed to determine how much each of these indices contributes to LU prediction.

Topic Control	**LTI**	**SMT**	**CS**
Strict	0.80	0.40	0.40
Weighted	0.90	0.53	0.53
Highest-Rest	0.90	0.67	0.67
High-Low	1.00	0.84	0.90

Table 3: Topic Control LU assignment performance averages of selected indices over a subset of data covering three dialogues with combined duration of 4.5 hours with total of 19 participants (7, 5, 7 per session).

Involvement	**NPI**	**TCI**	**ATP**
Strict	0.31	0.42	0.39
Weighted	0.46	0.49	0.42
Highest-Rest	0.67	0.77	0.68
High-Low	0.58	0.74	0.48

Table 4: Involvement LU assignment performance averages for selected indices over the same subset of data as in Table 3.

Topical Disagreement performance is shown in Table 5. We calculated precision and recall of assigning a correct degree of disagreement to each pair of speakers who are members of a group. Precision and recall averages are then computed over all meso-topics identified in the test dataset, which consists of three separate 90-minute dialogues involving 7, 5 and 7 speakers, respectively. Our calculation includes the cases where different sets of meso-topics were identified by the system and by the human coder. A strict mapping of levels of disagreement between speakers is hard to compute accurately; however, finding the speakers who disagree the most, or the least, is significantly more robust.

Topical Disagreement	**Prec.**	**Recall**
Strict	0.33	0.32
Weighted	0.54	0.54
Highest-Rest	0.89	0.85
High-Low	0.77	0.73

Table 5: Topical Disagreement LU assignment performance averages over 13 meso-topics discussed in three dialogues with combined duration of 4.5 hours with total of 19 participants (7, 5, and 7 per session).

6. Conclusion

In this paper we presented a preliminary design for modeling certain types of social phenomena in multi-party on-line dialogues. Initial, limited-scale evaluation indicates that the model can be effectively automated. Much work lies ahead, including large scale evaluation, testing index stability and resilience to NL component level error. Current performance of the system is based on only preliminary versions of linguistic modules (topic extraction, polarity assignments, etc.) which perform at only 70-80% accuracy, so these need to be improved as well. Research on Urdu and Chinese dialogues is just starting.

Acknowledgements

This research was funded by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through the U.S. Army Research Lab. All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of IARPA, the ODNI or the U.S. Government.

References

Agar, Michael. 1994. Language Shock, Understanding the Culture of Conversation. Quill, William Morrow, New York.

Allen, J. M. Core. 1997. Draft of DAMSL: Dialog Act Markup in Several Layers. www.cs. rochester.edu/research/cisd/resources/damsl/

Anderson, A., et al. 1991. The HCRC Map Task Corpus. Language and Speech 34(4), 351--366.

Austin, J. L. 1962. *How to do Things with Words*. Clarendon Press, Oxford.

Bird, Steven, et al. 2009. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly Media.

Blaylock, Nate. 2002. Managing Communicative Intentions in Dialogue Using a Collaborative Problem-Solving Model. Technical Report 774, University of Rochester, CS Dept.

Broadwell, G. A et al. (2010). Social Phenomena and Language Use. ILS Technical report.

Carberry, Sandra and Lynn Lambert. 1999. A Process Model for Recognizing Communicative Acts and Modeling Negotiation Dialogue. *Computational Linguistics*, 25(1), pp. 1-53.

Carletta, J. (2007). Unleashing the killer corpus: experiences in creating the multi-everything AMI Meeting Corpus. Language Resources and Evaluation Journal 41(2): 181-190

Carlson, Lauri. 1983. *Dialogue Games: An Approach to Discourse Analysis*. D. Reidel.

Eric N. Forsyth and Craig H. Martell. 2007. Lexical and Discourse Analysis of Online Chat Dialog. First IEEE International Conference on Semantic Computing (ICSC 2007), pp. 19-26.

Field, D., et al. 2008. Automatic Induction of Dialogue Structure from the Companions Dialogue Corpus, 4th Int. Workshop on Human-Computer Conversation, Bellagio.

Givon, Talmy. 1983. Topic continuity in discourse: A quantitative cross-language study. Amsterdam: John Benjamins.

Ivanovic, Edward. 2005. Dialogue Act Tagging for Instant Messaging Chat Sessions. In Proceedings of the ACL Student Research Workshop. 79–84. Ann Arbor, Michigan.

Ji, Gang Jeff Bilmes. 2006. Backoff Model Training using Partially Observed Data: Application to Dialog Act Tagging. HLT-NAACL

Jurafsky, Dan, Elizabeth Shriberg, and Debra Biasca. 1997. Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual. http://stripe.colorado.edu/~jurafsky/manual.august1.html

Jurafsky, D., et al. 1997. Automatic detection of discourse structure for speech recognition and understanding. IEEE Workshop on Speech Recognition and Understanding, Santa Barbara.

Khan, Faisal M., et al. 2002. Mining Chat-room Conversations for Social and Semantic Interactions. Computer Science and Engineering, Lehigh University.

Kim, Jihie., et al. 2007. An Intelligent Discussion-Bot for Guiding Student Interactions in Threaded Discussions. AAAI Spring Symposium on Interaction Challenges for Intelligent Assistants

Levin, L., et al. (1998). A discourse coding scheme for conversational Spanish. International Conference on Speech and Language Processing.

Levin, L., et al. (2003). Domain specific speech acts for spoken language translation. 4th SIGdial Workshop on Discourse and Dialogue.

Linell, Per. 1990. The power of dialogue dynamics. In Ivana Markov´a and Klaus Foppa, editors, *The Dynamics of Dialogue*. Harvester, 147–177.

Poesio, Massimo and Andrei Mikheev. 1998. The predictive power of game structure in dialogue act recognition. International Conference on Speech and Language Processing *(ICSLP-98)*.

Price, V., Capella, J. N., & Nir, L. (2002). Does disagreement contribute to more deliberative opinion? Political Communication, 19, 95-112.

Sacks, H. and Schegloff, E., Jefferson, G. 1974. A simplest systematic for the organization of turn-taking for conversation. In: *Language* 50(4), 696-735.

Samuel, K. et al. 1998. Dialogue Act Tagging with Transformation-Based Learning. 36th Annual Meeting of the ACL.

Scollon, Ron and Suzanne W. Scollon. 2001. *Intercultural Communication, A Discourse Approach*. Blackwell Publishing, Second Edition.

Searle, J. R. 1969. *Speech Acts*. Cambridge University Press, London-New York.

Shaikh, S. et al. 2010. DSARMD Annotation Guidelines, V. 2.5. ILS Technical Report.

Shaikh S. et al. 2010. MPC: A Multi-Party Chat Corpus for Modeling Social Phenomena in Discourse, Proc. LREC-2010, Malta.

Stolcke, Andreas et al. 2000. Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. *Computational Linguistics*, 26(3).

Stromer-Galley, J. 2007. Measuring deliberation's content: A coding scheme. Journal of Public Deliberation, 3(1).

Tianhao Wu, et al. 2002. Posting Act Tagging Using Transformation-Based Learning. Foundations of Data Mining and Discovery, IEEE International Conference on Data Mining

Twitchell, Douglas P., Jay F. Nunamaker Jr., and Judee K. Burgoon. 2004. Using Speech Act Profiling for Deception Detection. Intelligence and Security Informatics, LNCS, Vol. 3073

Webb, N., T. Liu, M. Hepple and Y. Wilks. 2008. Cross-Domain Dialogue Act Tagging. 6th International Conference on Language Resources and Evaluation (LREC-2008), Marrakech.

Discriminative Induction of Sub-Tree Alignment using Limited Labeled Data

Jun Sun[1,2] **Min Zhang**[1] **Chew Lim Tan**[2]
[1]Institute for Infocomm Research [2]School of Computing, National University of Singapore
sunjun@comp.nus.edu.sg mzhang@i2r.a-star.edu.sg tancl@comp.nus.edu.sg

Abstract

We employ Maximum Entropy model to conduct sub-tree alignment between bilingual phrasal structure trees. Various lexical and structural knowledge is explored to measure the syntactic similarity across Chinese-English bilingual tree pairs. In the experiment, we evaluate the sub-tree alignment using both gold standard tree bank and the automatically parsed corpus with manually annotated sub-tree alignment. Compared with a heuristic similarity based method, the proposed method significantly improves the performance with only limited sub-tree aligned data. To examine its effectiveness for multilingual applications, we further attempt different approaches to apply the sub-tree alignment in both phrase and syntax based SMT systems. We then compare the performance with that of the widely used word alignment. Experimental results on benchmark data show that sub-tree alignment benefits both systems by relaxing the constraint of the word alignment.

1 Introduction

Recent research in Statistical Machine Translation (SMT) tends to incorporate more linguistically grammatical information into the translation model known as linguistically motivated syntax-based models. To develop such models, the phrasal structure parse tree is usually adopted as the representation of bilingual sentence pairs either on the source side (Huang et al., 2006; Liu et al., 2006) or on the target side (Galley et al., 2006; Marcu et al., 2006), or even on both sides (Graehl and Knight, 2004; Zhang et al., 2007). Most of the above models either construct a pipeline to transform from/to tree structure, or synchronously generate two trees in parallel (i.e., synchronous parsing). Both cases require syntactically rich translational equivalences to handle non-local reordering. However, most current works obtain the syntactic translational equivalences by initially conducting alignment on the word level. To employ word alignment as a hard constraint for rule extraction has difficulty in capturing such non-local phenomena and will fully propagate the word alignment error to the later stage of rule extraction.

Alternatively, some initial attempts have been made to directly conduct syntactic structure alignment. As mentioned in Tinsley et al. (2007), the early work usually constructs the structure alignment by hand, which is time-consuming. Recent research tries to automatically align the bilingual syntactic sub-trees. However, most of these works suffer from the following problems. Firstly, the alignment is conducted based on heuristic rules, which may lose extensibility and generality in spite of accommodating some common cases (Groves et al., 2004). Secondly, various similarity computation methods are used based merely on lexical translation probabilities (Tinsley et al., 2007; Imamura, 2001) regardless of structural features. We believe the structure information is an important issue to capture the non-local structural divergence of languages by modeling beyond the plain text.

To address the above issues, we present a statistical framework based on Maximum Entropy (MaxEnt) model. Specifically, we consider sub-tree alignment as a binary classification problem and use Maximum Entropy model to classify each instance as *aligned* or *unaligned*. Then, we perform a greedy search within the reduced search space to conduct sub-tree alignment links based on the alignment probabilities obtained from the classifier.

Unlike the previous approaches that can only measure the structural divergence via lexical features, our approach can incorporate both lexical and structural features. Additionally, instead of explicitly describing the instances of sub-tree pairs as factorized sub-structures, we frame most of our features as score based feature functions, which helps solve the problem using limited sub-tree alignment annotated data. To train the model and evaluate the alignment performance, we adopt

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1047–1055,
Beijing, August 2010

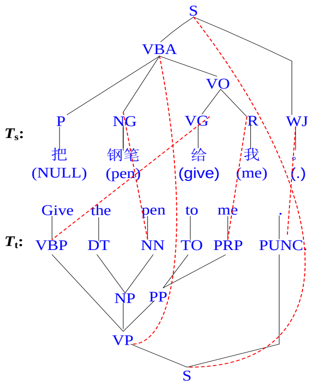


Figure 1: Sub-tree alignment as referred to Node alignment

HIT Chinese-English parallel tree bank for gold standard evaluation. To explore its effectiveness in SMT systems, we also manually annotate sub-tree alignment on automatically parsed tree pairs and perform the noisy data evaluation. Experimental results show that by only using limited sub-tree aligned data of both corpora, the proposed approach significantly outperforms the baseline method (Tinsley et al., 2007). The proposed features are very effective in modeling the bilingual structural similarity. We further apply the sub-tree alignment to relax the constraint of word alignment for both phrase and syntax based SMT systems and gain an improvement in BLEU.

2 Problem definition

A sub-tree alignment process pairs up the sub-trees across bilingual parse trees, whose lexical leaf nodes covered are translational equivalent, i.e., sharing the same semantics. Grammatically, the task conducts links between syntactic constituents with the maximum tree structures generated over their word sequences in bilingual tree pairs.

In general, sub-tree alignment can also be interpreted as conducting multiple links across internal nodes between sentence-aligned tree pairs as shown in Fig. 1. The aligned sub-tree pairs usually maintain a non-isomorphic relation with each other especially for higher layers. We adapt the same criteria as Tinsley et al. (2007) in our study:

(i) a node can only be linked once;

(ii) descendants of a source linked node may only link to descendants of its target linked counterpart;

(iii) ancestors of a source linked node may only link to ancestors of its target linked counterpart.

where the term "node" refers to root of a sub-tree, which can be used to represent the sub-tree.

3 Model

We solve the problem as binary classification and employ MaxEnt model with a greedy search.

Given a bilingual tree pair S^I and T^J, $S^I = \{s_1, \dots, s_i, \dots, s_I\}$ is the source tree consisting of I sub-trees，where I is also the number of nodes in the source tree S^I. $T^J = \{t_1, \dots, t_j, \dots, t_J\}$ is the target tree consisting of J sub-trees, where J is also the number of nodes in the target tree T^J.

For each sub-tree pair (s_i, t_j) in the given bilingual parse trees (S^I, T^J), the sub-tree alignment probability is given by:

$$Pr(a|s_i, t_{j,}\theta) = \frac{\exp[\sum_{m=1}^{M} \lambda_m h_m(a, s_i, t_j, \theta)]}{\sum_{a'} \exp[\sum_{m=1}^{M} \lambda_m h_m(a', s_i, t_j, \theta)]} \quad (1)$$

where

$$a = \begin{cases} 1 & \text{if } (s_i, t_j) \text{ is aligned} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Feature functions are defined in a quadruple (a, s_i, t_j, θ). θ is an additional variable to incorporate new dependencies other than the sub-tree pairs. For each feature function $h_m(a, s_i, t_j, \theta)$, a weight λ_m is applied to tailor the distribution.

After classifying the candidate sub-tree pairs as *aligned* or *unaligned*, we perform a greedy search within the reduced search space to conduct *sure* links based on the conditional probability $Pr(a|s_i, t_{j,}\theta)$ obtained from the classifier. The alignment probability is independently normalized for each sub-tree pair and hence suitable as a searching metric.

The greedy search algorithm can be described as an automaton. A state in the search space is a partial alignment with respect to the given bilingual tree pair. A transition is to add one more link of node pairs to the current state. The initial state has no link. The terminal state is a state where no more links can be added according to the definition in Section 2. We use greedy search to generate the best-links at the early stage. There are cases that the correctly-aligned tree pairs have very few links, while we have a bunch of candidates with lower alignment probabilities. However, the sum of the lower probabilities is larger than that of the correct links', since the number of correct links is much fewer. This makes the alignment results biased to be with more links. The greedy search helps avoid this asymmetric problem.

4 Feature Functions

In this section, we introduce a variety of feature functions to capture the semantically equivalent counterparts and structural divergence across languages. For the semantic equivalence, we define lexical and word alignment feature functions. Since those feature functions are directional, we describe most of these functions as conditional feature functions based on the conditional lexical probabilities. We also introduce the tree structural features to deal with the structural divergence of bilingual parse trees. Inspired by Burkett and Klein (2008), we introduce the feature functions in an internal-external manner based on the fact that the feature scores for an aligned sub-tree pair tend to be high inside both sub-trees, while they tend to be low inside one sub-tree and outside the other.

4.1 Internal Lexical Features

We use this feature to measure the degree of semantic equivalence of the sub-tree pair. According to the definition of sub-tree alignment in Section 2, the word sequence covered by the sub-tree pair should be translational equivalence. Therefore, the lexicons within the two corresponding sub-spans should be highly related in semantics. We define the internal lexical features as follows:

$$\phi(s_i|t_j) = \left(\prod_{v\in in(t_j)} \sum_{u\in in(s_i)} P(u|v)\right)^{\frac{1}{|in(t_j)|}}$$

$$\phi(t_j|s_i) = \left(\prod_{u\in in(s_i)} \sum_{v\in in(t_j)} P(v|u)\right)^{\frac{1}{|in(s_i)|}}$$

where $P(v|u)$ refers to the lexical translation probability from the source word u to the target word v within the sub-tree spans, while $P(u|v)$ refers to that from target to source; $in(s_i)$ refers to the word set for the internal span of the source sub-tree s_i, while $in(t_j)$ refers to that of the target sub-tree t_j.

4.2 Internal-External Lexical Features

Intuitively, lexical translation probabilities tend to be high within the translational equivalence, while low within the non-equivalent counterparts. According to this, we define the internal-external lexical feature functions as follows:

$$\varphi(s_i|t_j) = \frac{\sum_{v\in in(t_j)} \max_{u\,\in out(s_i)}\left\{\left(P(u|v)\cdot P(v|u)\right)^{\frac{1}{2}}\right\}}{|in(t_j)|}$$

$$\varphi(t_j|\,s_i) = \frac{\sum_{u\in in(s_i)} \max_{v\,\in out(t_j)}\left\{\left(P(u|v)\cdot P(v|u)\right)^{\frac{1}{2}}\right\}}{|in(s_i)|}$$

where $out(s_i)$ refers to the word set for the external span of the source sub-tree s_i, while $out(t_j)$ refers to that of the target sub-tree t_j. We choose a representation different from the internal lexical feature scores, since for cases with small inner span and large outer span, the sum of internal-external scores may be overestimated. As a result, we change the *sum* operation into *max*, which is easy to be normalized.

4.3 Internal Word Alignment Features

Although the word alignment information within bilingual sentence pairs is to some extent not reliable, the links of word alignment account much for the co-occurrence of the aligned terms. We define the internal word alignment features as follows:

$$\rho(s_i,t_j) = \frac{\sum_{v\in in(t_j)} \sum_{u\in in(s_i)} \delta(u,v)\cdot\left(P(u|v)\cdot P(v|u)\right)^{\frac{1}{2}}}{\left(|in(s_i)|\cdot|in(t_j)|\right)^{\frac{1}{2}}}$$

where

$$\delta(u,v) = \begin{cases} 1 & \text{if } (u,v) \text{ is aligned} \\ 0 & \text{otherwise} \end{cases}$$

The binary function $\delta(u,v)$ is introduced to trigger the computation only when a word aligned link exists for the two words (u,v) within the sub-tree span.

4.4 Internal-External Word Alignment Features

Similar to lexical features, we also introduce internal-external word alignment features as follows:

$$\sigma(s_i|t_j) = \frac{\sum_{v\in in(t_j)} \sum_{u\in out(s_i)} \delta(u,v)\cdot\left(P(u|v)\cdot P(v|u)\right)^{\frac{1}{2}}}{\left(|out(s_i)|\cdot|in(t_j)|\right)^{\frac{1}{2}}}$$

$$\sigma(t_j|s_i) = \frac{\sum_{v\in out(t_j)} \sum_{u\in in(s_i)} \delta(u,v)\cdot\left(P(u|v)\cdot P(v|u)\right)^{\frac{1}{2}}}{\left(|in(s_i)|\cdot|out(t_j)|\right)^{\frac{1}{2}}}$$

where

$$\delta(u,v) = \begin{cases} 1 & \text{if } (u,v) \text{ is aligned} \\ 0 & \text{otherwise} \end{cases}$$

4.5 Tree Structural Features

In addition to the lexical correspondence, we also capture the structural divergence by introducing the tree structural features as follows:

Span difference: Translational equivalent sub-tree pairs tend to share similar length of spans. Thus the model will penalize the candidate sub-tree pairs with largely different length of spans.

$$\xi(s_i,t_j) = \left|\frac{|in(s_i)|}{\max_{1\le k\le I}(|in(s_k)|)} - \frac{|in(t_j)|}{\max_{1\le h\le J}(|in(t_h)|)}\right|$$

Number of Descendants: Similarly, the number of the root's descendants of the aligned sub-trees should also correspond.

$$\tau(s_i,t_j) = \left|\frac{|R(s_i)|}{\max_{1\le k\le I}(|R(s_k)|)} - \frac{|R(t_j)|}{\max_{1\le h\le J}(|R(t_h)|)}\right|$$

where $R(.)$ refers to the descendant set of the root to an individual sub-tree.

Tree Depth difference: Intuitively, translational equivalent sub-tree pairs tend to have similar depth from the root node of the parse tree. We can further allow the model to penalize the candidate sub-tree pairs with different distance from the root node.

$$\omega(s_i, t_j) = \left|\frac{Depth(s_i)}{Height(S^I)} - \frac{Depth(t_j)}{Height(T^J)}\right|$$

4.6 Binary Grammatical Features

In the previous sections, we design some score based feature functions to describe syntactic tree structural similarities, rather than directly using the substructures. This is because for limited annotated tree alignment data, features like tokens and grammar rules are rather sparse. In spite of this, we still have a closed set of grammatical tags which can be covered by a small amount of data. Therefore, we use the combination of root grammar tags of the sub-tree pairs as binary features.

5 Training

We train the sub-tree alignment model in two steps:

Firstly, we learn the various feature functions. On one hand, GIZA++ is offline trained on a large amount of bilingual sentences to compute the lexical and word alignment features. On the other hand, the tree structural features, similar to word and phrase penalty features in phrase based SMT models, are computed online for both training and testing.

Secondly, we train the MaxEnt model in Eq. 1, using the training corpus which consists of the bilingual parse tree pairs with manually annotated sub-tree alignment. We apply the widely used GIS (Generalized Iterative Scaling) algorithm (Darroch and Ratcliff, 1972) to optimize λ_1^M. In practice, we modify Och's implementation YASMET.

Since we consider each sub-tree pair as an individual instance, it is easy to see that the negative samples heavily overwhelm the positive ones. For GIS training, such a skewed distribution easily drives the parameters to facilitate the negative instances. We address this problem by giving more weight to the positive training instances.

6 Experiments on Sub-Tree Alignments

We utilize two different corpora to evaluate the proposed sub-tree alignment method and its capability to plug in the related applications respectively. One is HIT English Chinese parallel tree bank with both tree structure and sub-tree alignment manually annotated. The other is the automatically parsed bilingual tree pairs (allowing minor parsing errors) with manually annotated sub-tree alignment. The latter benefits MT task, since most linguistically motivated syntax SMT systems require a held-out automatic parser to achieve rule induction.

6.1 Data preparation

For the gold standard corpus based experiment, we use HIT[1] Chinese-English parallel tree bank, which is collected from English learning text books in China as well as example sentences in dictionaries. It consists of 16131 gold standard parse tree pairs with manually annotated sub-tree alignments. The annotation strictly preserves the semantic equivalence, i.e., it only conducts *sure* links in the internal node level, while ignoring *possible* links adopted in word alignment. In contrast, in the POS level, n-to-n links are allowed in annotation. In order to be consistent with the definition in Section 2, we delete those n-to-n links in POS level. The word segmentation, tokenization and parse-tree in the corpus are manually constructed or checked. The Chinese parse tree in HIT tree bank adopts a different annotation criterion from the Penn TreeBank annotation, which is designed by the HIT research team. The new criterion can better facilitate the description of some rare structural phenomena in Chinese. The English parse tree still uses Penn TreeBank annotation. The statistics of HIT corpus is shown in Table 1.

	Chinese	English
# of Sentence pair	16131	
Avg. Sentence Length	13.06	13.00
Avg. # of sub-tree	21.60	23.74
Avg. # of alignment	11.71	

Table 1. Statistics for HIT gold standard Tree bank

Since the induction of sub-tree alignment is designed to benefit the machine translation modeling, it is preferable to conduct the sub-tree alignment experiment on the corpus for MT evaluation. However, most syntax based SMT systems use an automatic parser to facilitate training and decoding, which introduces parsing errors. Additionally, the gold standard HIT corpus is not applicable for MT

[1] HIT corpus is designed and constructed by HIT mitlab. http://mitlab.hit.edu.cn/index.php/resources.html . We licensed the corpus from them for research usage.

experiment due to problems of domain divergence, annotation discrepancy (Chinese parse tree adopts a different grammar from Penn Treebank annotations) and degree of tolerance for parsing errors.

Due to the above issues, we annotate a new data set to apply the sub-tree alignment in machine translation. We randomly select 300 bilingual sentence pairs from the Chinese-English FBIS corpus with the length ≤ 30 in both the source and target sides. The selected plain sentence pairs are further parsed by Stanford parser (Klein and Manning, 2003) on both the English and Chinese sides. We manually annotate the sub-tree alignment for the automatically parsed tree pairs according to the definition in Section 2. To be fully consistent with the definition, we strictly preserve the semantic equivalence for the aligned sub-trees to keep a high precision. In other words, we do not conduct any doubtful links. The corpus is further divided into 200 aligned tree pairs for training and 100 for testing. Some initial statistic of the automatically parsed corpus is shown in Table 2.

		Chinese	English
Train	# of Sentence pair	200	
	Avg. Sentence Length	17	20.84
	Avg. # of sub-tree	28.87	34.54
	Avg. # of alignment	17.07	
Test	# of Sentence pair	100	
	Avg. Sentence Length	16.84	20.75
	Avg. # of sub-tree	29.18	34.1
	Avg. # of alignment	17.75	

Table 2. FBIS selected Corpus Statistics

6.2 Baseline approach

We implement the work in Tinsley et al. (2007) as our baseline methodology.

Given a tree pair $< S^I, T^J >$, the baseline approach first takes all the links between the sub-tree pairs as alignment hypotheses, i.e., the Cartesian product of the two sub-tree sets:

$$\{s_1, \ldots, s_i, \ldots, s_I\} \times \{t_1, \ldots, t_j, \ldots, t_J\}$$

By using the lexical translation probabilities, each hypothesis is assigned an alignment score. All hypotheses with zero score are pruned out. Then the algorithm iteratively selects the link of the sub-tree pairs with the maximum score as a *sure* link, and blocks all hypotheses that contradict with this link and itself, until no non-blocked hypotheses remain.

The baseline system uses many heuristics in searching the optimal solutions with alternative score functions. Heuristic *skip1* skips the tied hypotheses with the same score, until it finds the highest-scoring hypothesis with no competitors of the same score. Heuristic *skip2* deals with the same problem. Initially, it skips over the tied hypotheses. When a hypothesis sub-tree pair (s_i, t_j) without any competitor of the same score is found, where neither s_i nor t_j has been skipped over, the hypothesis is chosen as a *sure* link. Heuristic *span1* postpones the selection of the hypotheses on the POS level. Since the highest-scoring hypotheses tend to appear on the leaf nodes, it may introduce ambiguity when conducting the alignment for a POS node whose child word appears twice in a sentence.

The baseline method proposes two score functions based on the lexical translation probability. They also compute the score function by splitting the tree into the internal and external components.

Tinsley et al. (2007) adopt the lexical translation probabilities dumped by GIZA++ (Och and Ney, 2003) to compute the span based scores for each pair of sub-trees. Although all of their heuristics combinations are re-implemented in our study, we only present the best result among them with the highest Recall and F-value as our baseline, denoted as *skip2_s1_span1*[2].

6.3 Experimental settings

• To examine the effectiveness of the proposed features, we

(1) learn the word alignment using the combination of the 14k of HIT tree bank and FBIS (240k) corpus for both our approach and the baseline method, and divide the remaining HIT corpus as 1k for training and 1k for testing.

(2) learn the word alignment on the entire FBIS training corpus (240k) for both our approach and the baseline method. We then train and test on FBIS corpus of 200 and 100 respectively as stated in Table 2.

• In our task, annotating large amount of sub-tree alignment corpus is time consuming and more difficult compared with the tasks like sequence labeling. One of the important issues we are concerned about is whether we can achieve an acceptable performance with limited training data. We

(3) adopt the entire FBIS data (240k) to learn the word alignment and various amount of HIT gold standard corpus to train the MaxEnt model. Then we test the alignment performance on the same HIT test set (1k) as (1).

[2] s1 denotes score function 1 in Tinsley et al. (2007)

Features	Precision	Recall	F-value
In Lexical	50.96	48.11	49.49
+ InOut Lexical	55.26	53.84	54.54
+ In word align	56.16	60.59	58.29
+ InOut word align	55.80	62.25	58.85
+ Tree Structure	57.64	63.11	60.25
+ Binary Feature	73.14	85.11	78.67
Baseline [Tinsley 2007]	64.14	66.99	65.53

Table 3. Sub-tree alignment of different feature combination for HIT gold standard test set

Features	Precision	Recall	F-value
In Lexical	63.53	54.87	58.88
+ InOut Lexical	66.00	63.66	64.81
+ In word align	70.89	75.88	73.30
+ InOut word align	72.05	80.16	75.89
+ Tree Structure	72.03	80.95	76.23
+ Binary Feature	76.08	85.29	80.42
Baseline [Tinsley 2007]	70.48	78.70	74.36

Table 4. Sub-tree alignment of different feature combination for FBIS test set

- We further test the robustness of our method under different amount of data to learn the lexical and word alignment feature functions. We gradually change the amount of FBIS corpus to train the word alignment. Then we

(4) use the same training (1k) and testing data (1k) with (1);

(5) use FBIS corpus 200 to train MaxEnt model and 100 for testing similar to (2).

6.4 Experimental results

We use Precision, Recall and F-score to measure the alignment performance and obtain the results as follows:

- In Table 3 and 4 for **Exp (1)** and **(2)** respectively, we show that by incrementally adding new features in a certain order, the F-value consistently increases and both outperform the baseline method. From both tables, we find that the **Binary features**, with the combination of root grammar tags of the sub-tree pairs, significantly improve the alignment performance. We also try the different combinations of the parent, child or even siblings to the root nodes. However, all these derivative configurations decrease the performance. We attribute the ineffectiveness to data sparseness. Further exploration suggests that the binary feature in HIT gold standard corpus exhibits a substantially larger improvement against other features than FBIS corpus (Table 3 against Table 4). The reason could be that the grammar tags in the gold standard corpus are accurate, while FBIS corpus suffers from parsing errors. Apart from that, the lexical/word-alignment features in Table 3 do not perform well, since the word alignment is trained mainly on the cross domain FBIS corpus. This is also an important reason why there is a large gap in performance between Table 3 and 4, where the automatic parsed FBIS corpus performs better than HIT gold standard tree bank in all configurations as well as the baseline.
- In Fig. 2(a) for **Exp (3)**, we examine performance under different amount of training data from 1k to 15k. The results change very little with over the amount of 1k. Even with only 0.25k training data, we are able to gain a result close to the best performance. This suggests that by utilizing only a small amount of sub-tree aligned corpus, we can still achieve a satisfactory alignment result. The benefits come from the usage of the score based feature functions by avoiding using substructures as binary features, which suffers from the data sparseness problem.
- In Fig. 2(b-e) for **Exp (4&5)**, we find that increasing the amount of corpus to train GIZA++ does not improve much for the proposed method on both HIT gold standard corpus (Fig. 2: b, c) and the automatic parsed data (Fig. 2: d, e). This is due to the various kinds of features utilized by the MaxEnt model, which does not bet on the lexical and word alignment feature too much. As for the baseline method, we can only detect a relatively large improvement in the initial increment of corpus, while later additions do not help. This result suggests that the baseline method is relatively less extensible since it works completely on the lexical similarities which can be only learned from the word alignment corpus.

7 Experiments on Machine Translation

In addition to the alignment evaluation, we conduct MT evaluation as well. We explore the effectiveness of sub-tree alignment for both phrase and linguistically motivated syntax based systems.

7.1 Experimental configuration

In the experiments, we train the translation model on FBIS corpus (7.2M (Chinese) + 9.2M (English) words in 240,000 sentence pairs) and train a 4-gram language model on the Xinhua portion of the English Gigaword corpus (181M words) using the SRILM Toolkits (Stolcke, 2002). We use these

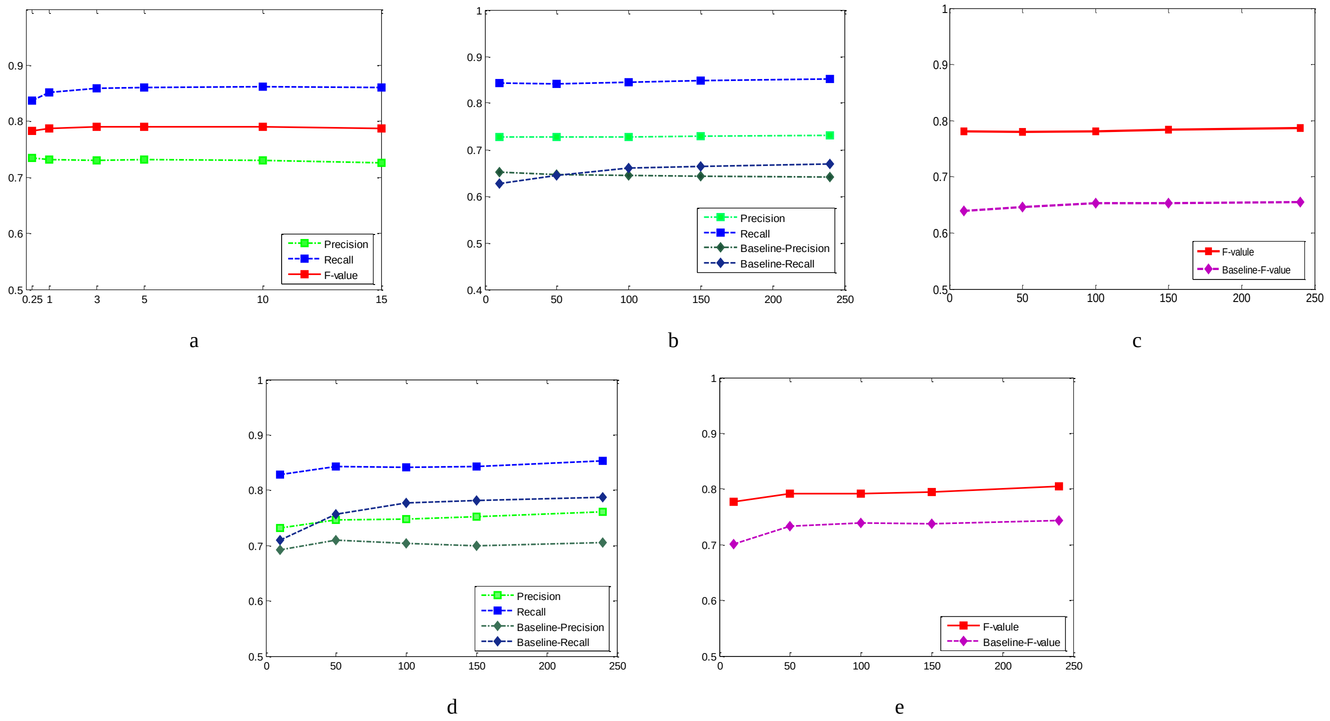


Figure 2: a. Precision/Recall/F-score for various amount of training data (k).
b~e. Various amount of data to train word alignment
b. Precision/Recall for HIT test set. c. F-score for HIT test set.
d. Precision/Recall for FBIS test set. e. F-score for FBIS test set.

sentences with less than 50 characters from the NIST MT-2002 test set as the development set (to speed up tuning for syntax based system) and the NIST MT-2005 test set as our test set. We use the Stanford parser (Klein and Manning, 2003) to parse bilingual sentences on the training set and Chinese sentences on the development and test set. The evaluation metric is case-sensitive BLEU-4.

For the phrase based system, we use Moses (Koehn et al, 2007) with its default settings. For the syntax based system, since sub-tree alignment can directly benefit Tree-2-Tree based systems, we apply the sub-tree alignment in an SMT system based on Synchronous Tree Substitution Grammar (STSG) (Zhang et al., 2007). The STSG based decoder uses a pair of *elementary tree* as a basic translation unit. Recent research on tree based systems shows that relaxing the restriction from tree structure to tree sequence structure (Synchronous Tree Sequence Substitution Grammar: STSSG) significantly improves the translation performance (Zhang et al., 2008). We implement the STSG/STSSG based model in Pisces decoder with the same features and settings in Sun et al. (2009). The STSSG based decoder translates each span iteratively in a bottom up manner which guarantees that when translating a source span, any of its sub-spans has already been translated. The STSG based experiment can be easily achieved by restricting the translation rule set in the STSSG decoder to be elementary tree pairs only.

For the alignment setting of the baselines, we use the word alignment trained on the entire FBIS(240k) corpus by GIZA++ with heuristic grow-diag-final for Moses and the syntax systems and perform rule extraction constrained on the word alignment. As for the experiments adopting sub-tree alignment, we use the above word alignment to learn lexical/word alignment features, and train the sub-tree alignment model with FBIS training data (200).

7.2 Experimental results

Utilizing the syntactic rules only has been argued to be ineffective (Koehn et al., 2003). Therefore, instead of using the sub-tree aligned rules only, we try to improve the word alignment constrained rule set by sub-tree alignment as shown in Table 5.

Firstly, we try to *Directly Concatenate* (DirC) the sub-tree alignment constraint rule set[3] to the original syntax/phrase rule set based on word alignment. Then we re-train the MT model based

[3] For syntax based system, it's just the sub-tree pairs deducted from the sub-tree alignment; for phrase based system, it's the phrases with context equivalent to the aligned sub-tree pairs.

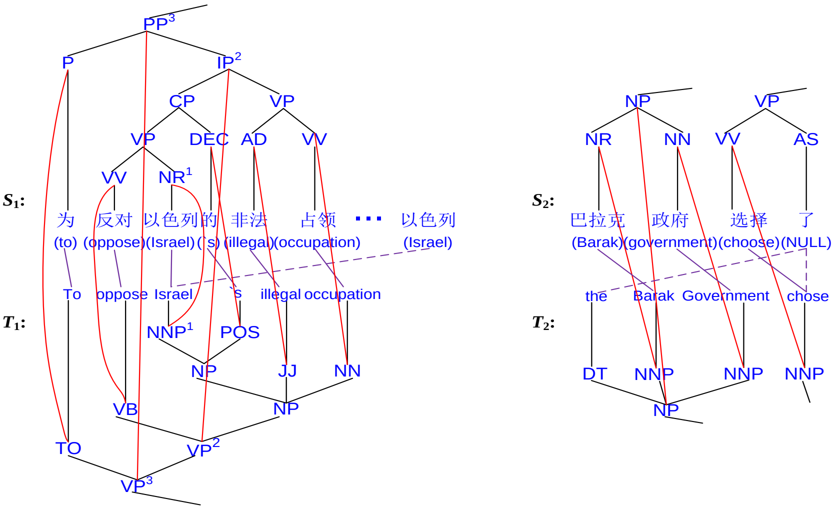

S1:
T1:
S2:
T2:
PP3
P
IP2
CP
VP
DEC
AD
VV
NR1
为 反对 以色列 的 非法 占领 ··· 以色列
(to) (oppose) (Israel) ('s) (illegal) (occupation) (Israel)
To oppose Israel 's illegal occupation
NNP1
POS
NP
JJ
NN
VB
TO
VP2
VP3
NP
NR
NN
VP
VV
AS
巴拉克 政府 选择 了
(Barak) (government) (choose) (NULL)
the Barak Government chose
DT
NNP
NNP
NNP
NP

Figure 3: Comparison between Sub-tree alignment results and Word alignment results

on the obtained rule set. Tinsley et al. (2009) attempts different duplication of sub-tree alignment constraint rule set to append to the original phrase rule set and reports positive results. However, as shown in Table 5, we only achieve very minor improvement (in STSSG based model the score even drops) by direct introducing the new rules.

Secondly, we propose a new approach to utilize sub-tree alignment by modifying the rule extraction process. We allow the bilingual phrases which are consistent with *Either* Word alignment *or* Sub-tree alignment (EWoS) instead of to be consistent with word alignment only. The results in Table 5 show that EWoS achieves consistently better performance than the baseline and DirC method. We also find that sub-tree alignment benefits the STSSG based model less compared with other systems. This is probably due to the fact that the STSSG based system relies much on the tree sequence rules.

To benefit intuitive understanding, we provide two alignment snippets in the MT training corpus in Fig. 3, where the red lines across the non-terminal nodes are the sub-tree aligned links conducted by our model, while the purple lines across the terminal nodes are the word alignment links trained by GIZA++. In the first example, the word *Israel* is wrongly aligned to two "以色列"s by GIZA++, where the wrong link is denoted by the dash line. This is common, since in a compound sentence in English, the entities appeared more than once are often replaced by pronouns at its later appearances. Therefore, the syntactic rules constraint by NR^1-NNP^1, IP^2-VP^2 and PP^3-VP^3 respectively cannot be extracted for syntax systems; while for phrase systems, context around the first "以色列" cannot be fully explored. In the second example, the empty word "了" is wrongly aligned, which usually occurs in Chinese-English word alignment. As shown in Fig. 3, both cases can be resolved by sub-tree alignment conducted by our model, indicating that sub-tree alignment is a decent supplement to the word alignment rule set.

System	Rules	BLEU
Moses	BP*	23.86
	DirC	24.12
	EWoS	24.45
Syntax STSG	STSG	24.71
	DirC	24.91
	EWoS	25.21
Syntax STSSG	STSSG	25.92
	DirC	25.88
	EWoS	26.12

Table 5. MT evaluation on various systems
BP* denotes bilingual phrases.
BP, STSG, STSSG are baseline rule sets using word alignment to constrain rule extraction.

8 Conclusion

In this paper, we propose a framework for bilingual sub-tree alignment using Maximum Entropy model. We explore various lexical and structural features to improve the alignment performance. We also manually annotated the automatic parsed tree pairs for both alignment evaluation and MT experiment. Experimental results show that our alignment framework significantly outperforms the baseline method and the proposed features are very effective to capture the bilingual structural similarity. Additionally, we find that our approach can perform well using only a small amount of sub-tree aligned training corpus. Further experiment shows that our approach benefits both phrase and syntax based MT systems.

References

David Burkett and Dan Klein. 2008. *Two languages are better than one (for syntactic parsing)*. In Proceedings of EMNLP-08. 877-886.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang and Ignacio Thayer. 2006. *Scalable Inference and training of context-rich syntactic translation models*. In Proceedings of COLING-ACL-06. 961-968.

Jonathan Graehl and Kevin Knight. 2004. *Training tree transducers*. In Proceedings of HLT-NAACL-2004. 105-112.

Declan Groves, Mary Hearne, and Andy Way. 2004. *Robust sub-sentential alignment of phrase-structure trees*. In Proceedings of COLING-04, pages 1072-1078.

Liang Huang, Kevin Knight and Aravind Joshi. 2006. *Statistical syntax-directed translation with extended domain of Locality*. In Proceedings of AMTA-06.

Kenji Imamura. 2001. *Hierarchical Phrase Alignment Harmonized with Parsing*. In Proceedings of NLPRS. 377-384.

Dan Klein and Christopher D. Manning. 2003. *Accurate Unlexicalized Parsing*. In Proceedings of ACL-03. 423-430.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin and Evan Herbst. 2007. *Moses: Open Source Toolkit for Statistical Machine Translation*. In Proceedings of ACL-07. 177-180.

Philipp Koehn, Franz Josef Och and Daniel Marcu. 2003. *Statistical Phrase-based Translation*. In Proceedings of HLT-NAACL-2003. 48-54.

Yang Liu, Qun Liu and Shouxun Lin. 2006. *Tree-to-String alignment template for statistical machine translation*. In Proceedings of ACL-06, 609-616.

Daniel Marcu, Wei Wang, Abdessamad Echihabi and Kevin Knight. 2006. *SPMT: statistical machine translation with syntactified target language phrases*. In Proceedings of EMNLP-06. 44-52.

Franz Josef Och and Hermann Ney. 2003. *A systematic comparison of various statistical alignment models*. Computational Linguistics, 29(1):19-51, March.

Andreas Stolcke. 2002. *SRILM - an extensible language modeling toolkit*. In Proceedings of ICSLP-02. 901-904.

Jun Sun, Min Zhang and Chew Lim Tan. 2009. *A non-contiguous Tree Sequence Alignment-based Model for Statistical Machine Translation*. In Proceedings of ACL-IJCNLP-09. 914-922.

John Tinsley, Ventsislav Zhechev, Mary Hearne, and Andy Way. 2007. *Robust language pair-independent sub-tree alignment*. In Proceedings of Machine Translation Summit-XI-07.

John Tinsley, Mary Hearne, and Andy Way. 2009. *Parallel treebanks in phrase-based statistical machine translation*. In Proceedings of CICLING-09.

Min Zhang, Hongfei Jiang, AiTi Aw, Jun Sun, Sheng Li and Chew Lim Tan. 2007. *A tree-to-tree alignment-based model for statistical machine translation*. In Proceedings of MT Summit-XI -07. 535-542.

Min Zhang, Hongfei Jiang, AiTi Aw, Haizhou Li, Chew Lim Tan and Sheng Li. 2008. *A tree sequence alignment-based tree-to-tree translation model*. In Proceedings of ACL-08. 559-567.

Investigating the cross-linguistic potential of VerbNet -style classification

Lin Sun and Anna Korhonen
Computer Laboratory
University of Cambridge
ls418,alk23@cl.cam.ac.uk

Thierry Poibeau
LaTTiCe, UMR8094
CNRS & ENS
thierry.poibeau@ens.fr

Cédric Messiant
LIPN, UMR7030
CNRS & U. Paris 13
cedric.messiant@lipn.fr

Abstract

Verb classes which integrate a wide range of linguistic properties (Levin, 1993) have proved useful for natural language processing (NLP) applications. However, the real-world use of these classes has been limited because for most languages, no resources similar to VerbNet (Kipper-Schuler, 2005) are available. We apply a verb clustering approach developed for English to French – a language for which no such experiment has been conducted yet. Our investigation shows that not only the general methodology but also the best performing features are transferable between the languages, making it possible to learn useful VerbNet style classes for French automatically without language-specific tuning.

1 Introduction

A number of verb classifications have been built to support natural language processing (NLP) tasks (Grishman et al., 1994; Miller, 1995; Baker et al., 1998; Palmer et al., 2005; Kipper-Schuler, 2005; Hovy et al., 2006). These include both syntactic and semantic classifications, as well as ones which integrate aspects of both. Classifications which integrate a wide range of linguistic properties can be particularly useful for NLP applications suffering from data sparseness. One such classification is VerbNet (Kipper-Schuler, 2005). Building on the taxonomy of Levin (1993), VerbNet groups verbs (e.g. *deliver, post, dispatch*) into classes (e.g. SEND) on the basis of their shared meaning components and syntactic behaviour, identified in terms of meaning preserving diathesis alternations. Such classes can be identified across the entire lexicon, and they may also apply across languages, since their meaning components are said to be cross-linguistically applicable (Jackendoff, 1990).

Offering a powerful tool for generalization, abstraction and prediction, VerbNet classes have been used to support many important NLP tasks, including e.g. computational lexicography, parsing, word sense disambiguation, semantic role labeling, information extraction, question-answering, and machine translation (Swier and Stevenson, 2004; Dang, 2004; Shi and Mihalcea, 2005; Abend et al., 2008). However, to date their exploitation has been limited because for most languages, no Levin style classification is available.

Since manual classification is costly (Kipper et al., 2008) automatic approaches have been proposed recently which could be used to learn novel classifications in a cost-effective manner (Joanis et al., 2008; Li and Brew, 2008; Ó Séaghdha and Copestake, 2008; Vlachos et al., 2009; Sun and Korhonen, 2009). However, most work on Levin type classification has focussed on English. Large-scale research on other languages such as German (Schulte im Walde, 2006) and Japanese (Suzuki and Fukumoto, 2009) has focussed on semantic classification. Although the two classification systems have shared properties, studies comparing the overlap between VerbNet and WordNet (Miller, 1995) have reported that the mapping is only partial and many to many due to fine-grained nature of classes based on synonymy (Shi and Mihalcea, 2005; Abend et al., 2008).

Only few studies have been conducted on Levin style classification for languages other than English. In their experiment involving 59 verbs and three classes, Merlo et al. (2002) applied a supervised approach developed for English to Italian, obtaining high accuracy (86.3%). In another experiment with 60 verbs and three classes,

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1056–1064,
Beijing, August 2010

they showed that features extracted from Chinese translations of English verbs can improve English classification. These results are promising, but those from a later experiment by Ferrer (2004) are not. Ferrer applied a clustering approach developed for English to Spanish, and evaluated it against the manual classification of Vázquez et al. (2000), constructed using criteria similar (but not identical) to Levin's. This experiment involving 514 verbs and 31 classes produced results only slightly better than the random baseline.

In this paper, we investigate the cross-linguistic potential of Levin style classification further. In past years, verb classification techniques – in particular unsupervised ones – have improved considerably, making investigations for a new language more feasible. We take a recent verb clustering approach developed for English (Sun and Korhonen, 2009) and apply it to French – a major language for which no such experiment has been conducted yet. Basic NLP resources (corpora, taggers, parsers and subcategorization acquisition systems) are now sufficiently developed for this language for the application of a state-of-the-art verb clustering approach to be realistic.

Our investigation reveals similarities between the English and French classifications, supporting the linguistic hypothesis (Jackendoff, 1990) and the earlier result of Merlo et al. (2002) that Levin classes have a strong cross-linguistic basis. Not only the general methodology but also best performing features are transferable between the languages, making it possible to learn useful classes for French automatically without language-specific tuning.

2 French Gold Standard

The development of an automatic verb classification approach requires at least an initial gold standard. Some syntactic (Gross, 1975) and semantic (Vossen, 1998) verb classifications exist for French, along with ones which integrate aspects of both (Saint-Dizier, 1998). Since none of these resources offer classes similar to Levins', we followed the idea of Merlo et al. (2002) and translated a number of Levin classes from English to French. As our aim was to to investigate the cross-linguistic applicability of classes, we took an English gold standard which has been used to evaluate several recent clustering works – that of Sun et al. (2008). This resource includes 17 fine-grained Levin classes. Each class has 12 member verbs whose predominant sense in English (according to WordNet) belongs to that class.

Member verbs were first translated to French. Where several relevant translations were identified, each of them was considered. For each candidate verb, subcategorization frames (SCFs) were identified and diathesis alternations were considered using the criteria of Levin (1993): alternations must result in the same or extended verb sense. Only verbs sharing diathesis alternations were kept in the class.

For example, the gold standard class 31.1 AMUSE includes the following English verbs: *stimulate, threaten, shock, confuse, upset, overwhelm, scare, disappoint, delight, exhaust, intimidate* and *frighten*. Relevant French translations were identified for all of them: *abattre, accabler, briser, déprimer, consterner, anéantir, épuiser, exténuer, écraser, ennuyer, éreinter, inonder*. The majority of these verbs take similar SCFs and diathesis alternations, e.g. *Cette affaire écrase Marie (de chagrin), Marie est écrasée par le chagrin, Le chagrin écrase Marie*. However, *stimuler* (*stimulate*) and *menacer* (*threaten*) do not, and they were therefore removed.

40% of translations were discarded from classes because they did not share the same alernations. The final version of the gold standard (shown in table 1) includes 171 verbs in 16 classes. Each class is named according to the original Levin class. The smallest class (30.3) includes 7 verbs and the largest (37.3) 16. The average number of verbs per class is 10.7.

3 Verb Clustering

We performed an experiment where we

- took a French corpus and a SCF lexicon automatically extracted from that corpus,
- extracted from these resources a range of features (lexical, syntactic and semantic) – a representative sample of those employed in recent English experiments,

Class No	Class	Verbs
9.1	PUT	accrocher, déposer, mettre, placer, répartir, réintégrer, empiler, emporter, enfermer, insérer, installer
10.1	REMOVE	ôter, enlever, retirer, supprimer, retrancher, débarrasser, soustraire, décompter, éliminer
11.1	SEND	envoyer, lancer, transmettre, adresser, porter, expédier, transporter, jeter, renvoyer, livrer
13.5.1	GET	acheter, prendre, saisir, réserver, conserver, garder, préserver, maintenir, retenir, louer, affréter
18.1	HIT	cogner, heurter, battre, frapper, fouetter, taper, rosser, brutaliser, éreinter, maltraiter, corriger,
22.2	AMALGAMATE	incorporer, associer, réunir, mélanger, mêler, unir, assembler, combiner, lier, fusionner
29.2	CHARACTERIZE	appréhender, concevoir, considérer, décrire, définir, dépeindre, désigner, envisager, identifier, montrer, percevoir, représenter, ressentir
30.3	PEER	regarder, écouter, examiner, considérer, voir, scruter, dévisager
31.1	AMUSE	abattre, accabler, briser, déprimer, consterner, anéantir, épuiser, exténuer, écraser, ennuyer, éreinter, inonder,
36.1	CORRESPOND	coopérer, participer, collaborer, concourir, contribuer, prendre part, s’associer, travaille
37.3	MANNER OF SPEAKING	râler, gronder, crier, ronchonner, grogner, bougonner, maugréer, rouspéter, grommeler, larmoyer, gémir, geindre, hurler, gueuler, brailler, chuchoter
37.7	SAY	dire, révéler, déclarer, signaler, indiquer, montrer, annoncer, répondre, affirmer, certifier, répliquer
43.1	LIGHT EMISSION	briller, étinceler, flamboyer, luire, resplendir, pétiller, rutiler, rayonner., scintiller
45.4	CHANGE OF STATE	mélanger, fusionner, consolider, renforcer, fortifier, adoucir, polir, atténuer, tempérer, pétrir, façonner, former
47.3	MODES OF BEING	trembler, frémir, osciller, vaciller, vibrer, tressaillir, frissonner, palpiter, grésiller, trembloter, palpiter
51.3.2	RUN	voyager, aller, se promener, errer, circuler, se déplacer, courir, bouger, naviguer, passer

Table 1: A Levin style gold standard for French

- clustered the features using a method which has proved promising in both English and German experiments: spectral clustering,
- evaluated the clusters both quantitatively (using the gold standard) and qualitatively,
- and compared the performance to that recently obtained for English in order to gain a better understanding of the cross-linguistic and language-specific properties of verb classification

This work is described in the subsequent sections.

3.1 Data: the LexSchem Lexicon

We extracted the features for clustering from LexSchem (Messiant et al., 2008). This large subcategorization lexicon provides SCF frequency information for 3,297 French verbs. It was acquired fully automatically from Le Monde newspaper corpus (200M words from years 1991-2000) using ASSCI – a recent subcategorization acquisition system for French (Messiant, 2008). Systems similar to ASSCI have been used in recent verb classification works e.g. (Schulte im Walde, 2006; Li and Brew, 2008; Sun and Korhonen, 2009). Like these other systems, ASSCI takes raw corpus data as input. The data is first tagged and lemmatized using the Tree-Tagger and then parsed using Syntex (Bourigault et al., 2005). Syntex is a shallow parser which employs a combination of statistics and heuristics to identify grammatical relations (GRs) in sentences. ASSCI considers GRs where the target verbs occur and constructs SCFs from nominal, prepositional and adjectival phrases, and infinitival and subordinate clauses. When a verb has no dependency, its SCF is considered as intransitive. ASSCI assumes no predefined list of SCFs but almost any combination of permitted constructions can appear as a candidate SCF. The number of automatically generated SCF types in LexSchem is 336.

Many candidate SCFs are noisy due to processing errors and the difficulty of argument-adjunct distinction. Most SCF systems assume that true arguments occur in argument positions more frequently than adjuncts. Many systems also integrate filters for removing noise from system output. When LexSchem was evaluated after filtering its F-measure was 69 – which is similar to that of other current SCF systems (Messiant et al., 2008) We used the unfiltered version of the lexicon because English experiments have shown that information about adjuncts can help verb clustering (Sun et al., 2008).

4 Features

Lexical entries in LexSchem provide a variety of material for verb clustering. Using this material, we constructed a range of features for experimentation. The first three include basic information about SCFs:

F1: SCFs and their relative frequencies with individual verbs. SCFs abstract over particles and prepositions.

F2: F1, with SCFs parameterized for the tense (the POS tag) of the verb.

F3: F2, with SCFs parameterized for prepositions (PP).

The following six features include information about the lexical context (co-occurrences) of verbs. We adopt the best method of Li and Brew (2008) where collocations (COs) are extracted from the window of words immediately preceding and following a lemmatized verb. Stop words are removed prior to extraction.

F4, F6, F8: COs are extracted from the window of 4, 6 and 8 words, respectively. The relative word position is ignored.

F5, F7, F9: F4, F6 and F8 with the relative word position recorded.

The next four features include information about lexical preferences (LP) of verbs in argument head positions of specific GRs associated with the verb:

F10: LP(PREP): the type and frequency of prepositions in the preposition (PREP) relation.

F11: LP(SUBJ): the type and frequency of nouns in the subject (SUBJ) relation.

F12: LP(IOBJ): the type and frequency of nouns in the object (OBJ) and indirect object (IOBJ) relation.

F13: LP(ALL): the combination of F10-F13.

The final two features refine SCF features with LPs and semantic information about verb selectional preferences (SP):

F14-F16: F1-F3 parameterized for LPs.

F17: F3 refined with SPs.

We adopt a fully unsupervised approach to SP acquisition using the method of Sun and Korhonen (2009), with the difference that we determine the optimal number of SP clusters automatically following Zelnik-Manor and Perona (2004). The method is introduced in the following section. The approach involves (i) taking the GRs (SUBJ, OBJ, IOBJ) associated with verbs, (ii) extracting all the argument heads in these GRs, and (iii) clustering the resulting N most frequent argument heads into M classes. The empirically determined N 200 was used. The method produced 40 SP clusters.

5 Clustering Methods

Spectral clustering (SPEC) has proved promising in previous verb clustering experiments (Brew and Schulte im Walde, 2002; Sun and Korhonen, 2009) and other similar NLP tasks involving high dimensional feature space (Chen et al., 2006). Following Sun and Korhonen (2009) we used the MNCut spectral clustering (Meila and Shi, 2001) which has a wide applicability and a clear probabilistic interpretation (von Luxburg, 2007; Verma and Meila, 2005). However, we extended the method to determine the optimal number of clusters automatically using the technique proposed by (Zelnik-Manor and Perona, 2004).

Clustering groups a given set of verbs $V = \{v_n\}_{n=1}^{N}$ into a disjoint partition of K classes. SPEC takes a similarity matrix as input. All our features can be viewed as probabilistic distributions because the combination of different features is performed via parameterization. Thus we use the Jensen-Shannon divergence (JSD) to construct the similarity matrix. The JSD between two feature vectors v and v' is $d_{jsd}(v, v') = \frac{1}{2}D(v||m) + \frac{1}{2}D(v'||m)$ where D is the Kullback-Leibler divergence, and m is the average of the v and v'.

The similarity matrix W is constructed where $W_{ij} = \exp(-d_{jsd}(v, v'))$. In SPEC, the similarities W_{ij} are viewed as the connection weight ij of a graph G over V. The similarity matrix W is thus the adjacency matrix for G. The degree of a vertex i is $d_i = \sum_{j=1}^{N} w_{ij}$. A cut between two partitions A and A' is defined to be $\text{Cut}(A, A') = \sum_{m \in A, n \in A'} W_{mn}$.

The similarity matrix W is normalized into a stochastic matrix P.

$$P = D^{-1}W \qquad (1)$$

The degree matrix D is a diagonal matrix where $D_{ii} = d_i$.

It was shown by Meila and Shi (2001) that if P has the K leading eigenvectors that are piecewise constant[1] with respect to a partition I^* and their eigenvalues are not zero, then I^* minimizes the multiway normalized cut(MNCut):

$$\text{MNCut}(I) = K - \sum_{k=1}^{K} \frac{\text{Cut}(I_k, I_k)}{\text{Cut}(I_k, I)}$$

P_{mn} can be interpreted as the transition probability between vertices m, n. The criterion can thus be expressed as $\text{MNCut}(I) = \sum_{k=1}^{K}(1 - P(I_k \rightarrow I_k|I_k))$ (Meila, 2001), which is the sum of transition probabilities across different clusters. This criterion finds the partition where the random walks are most likely to happen within the same cluster. In practice, the leading eigenvectors of P are not piecewise constant. But we can extract the partition by finding the approximately equal elements in the eigenvectors using a clustering algorithm like K-Means.

As the value of K is not known beforehand, we use Zelnik-Manor and Perona (2004)'s method to estimate it. This method finds the optimal value by minimizing a cost function based on the eigenvector structure of W.

Like Brew and Schulte im Walde (2002), we compare SPEC against a K-Means baseline. We used the Matlab implementation with euclidean distance as the distance measure.

6 Experimental Evaluation

6.1 Data and Pre-processing

The SCF-based features (F1-F3 and F14-F17) were extracted directly from LexSchem. The CO (F4-F9) and LP features (F10-F13) were extracted from the raw and parsed corpus sentences, respectively, which were used for creating the lexicon. Features that only appeared once were removed. Feature vectors were normalized by the sum of the feature values before clustering. Since our clustering algorithms have an element of randomness, we repeated clustering multiple times. We report the results that minimize the distortion (the distance to cluster centroid).

6.2 Evaluation Measures

We employ the same measures for evaluation as previously employed e.g. by Ó Séaghdha and Copestake (2008) and Sun and Korhonen (2009).

The first measure is modified purity (mPUR) – a global measure which evaluates the mean precision of clusters. Each cluster is associated with its prevalent class. The number of verbs in a cluster K that take this class is denoted by $n_{prevalent}$(K). Verbs that do not take it are considered as errors. Clusters where $n_{prevalent}$(K) = 1 are disregarded as not to introduce a bias towards singletons:

$$\text{mPUR} = \frac{\sum_{n_{prevalent(k_i)}>2} n_{prevalent(k_i)}}{\text{number of verbs}}$$

The second measure is weighted class accuracy (ACC): the proportion of members of dominant clusters DOM-CLUST$_i$ within all classes c_i.

$$\text{ACC} = \frac{\sum_{i=1}^{C} \text{verbs in DOM-CLUST}_i}{\text{number of verbs}}$$

mPUR and ACC can be seen as a measure of precision(P) and recall(R) respectively. We calculate F measure as the harmonic mean of P and R:

$$\text{F} = \frac{2 \cdot \text{mPUR} \cdot \text{ACC}}{\text{mPUR} + \text{ACC}}$$

The random baseline (BL) is calculated as follows: $\text{BL} = 1/\text{number of classes}$

7 Evaluation

7.1 Quantitative Evaluation

In our first experiment, we evaluated 116 verbs – those which appeared in LexSchem the minimum

[1] The eigenvector v is piecewise constant with respect to I if $v(i) = v(j) \forall i, j \in I_k$ and $k \in 1, 2...K$

of 150 times. We did this because English experiments had shown that due to the Zipfian nature of SCF distributions, 150 corpus occurrences are typically needed to obtain a sufficient number of frames for clustering (Sun et al., 2008).

Table 2 shows F-measure results for all the features. The 4th column of the table shows, for comparison, the results of Sun and Korhonen (2009) obtained for English when they used the same features as us, clustered them using SPEC, and evaluated them against the English version of our gold standard, also using F-measure[2].

As expected, SPEC (the 2nd column) outperforms K-Means (the 3rd column). Looking at the basic SCF features F1-F3, we can see that they perform significantly better than the BL method. F3 performs the best among the three features both in French (50.6 F) and in English (63.3 F). We therefore use F3 as the SCF feature in F14-F17 (the same was done for English).

In French, most CO features (F4-F9) outperform SCF features. The best result is obtained with F7: 55.1 F. This is clearly better than the best SCF result 50.6 (F3). This result is interesting since SCFs correspond better than COs with features used in manual Levin classification. Also, SCFs perform considerably better than COs in the English experiment (we only have the result for F4 available, but it is considerably lower than the result for F3). However, earlier English studies have reported contradictory results (e.g. Li and Brew (2008) showed that CO performs better than SCF in supervised verb classification), indicating that the role of CO features in verb classification requires further investigation.

Looking at the LP features, F13 produces the best F (52.7) for French which is slightly better than the best SCF result for the language. Also in English, F13 performs the best in this feature group and yields a higher result than the best SCF-based feature F3.

Parameterizing the best SCF feature F3 with LPs (F14-16) and SPs (F17) yields better performance in French. F15 and F17 have the F of 54.5 and 54.6, respectively. These results are so close to the result of the best CO feature F7 (55.1 – which is the highest result in this experiment) that the differences are not statistically significant. In English, the results of F14-F17 are similarly good; however, only F17 beats the already high performance of F13.

On the basis of this experiment, it is difficult to tell whether shallow CO features or more sophisticated SCF-based features are better for French. In the English experiment sophisticated features performed better (the SCF-SP feature was the best). However, the English experiment employed a much larger dataset. These more sophisticated features may suffer from data sparseness in our French experiment since although we required the minimum of 150 occurrences per verb, verb clustering performance tends to improve when more data is available, and given the fine-grained nature of LexShem SCFs it is likely that more data is required for optimal performance.

We therefore performed another experiment with French on the full set of 147 verbs, using SPEC, where we investigated the effect of instance filtering on the performance of the best features from each feature group: F3, F7, F13 and F17. The results shown in Table 3 reveal that the performance of the features remains fairly similar until the instance threshold of 1000. When 2000 occurrences per verb are used, the differences become clearer, until at the threshold of 4000, it is obvious that the most sophisticated SCF-SP feature F17 is by far the best feature for French (65.4 F) and the SCF feature F3 the second best (60.5 F). The CO-feature F7 and the LP feature F13 are not nearly as good (53.4 and 51.0 F).

Although the results at different thresholds are not comparable due to the different number of verbs and classes (see columns 2-3), the results for features at the same threshold are. Those results suggest that when 2000 or more occurrences per verb are used, most features perform like they performed for English in the experiment of Sun and Korhonen (2009), with CO being the least informative[3] and SCF-SP being the most informa-

[2] Note that the results for the two languages are not mutually comparable due to differences in test sets, data sizes, and feature extraction systems (see Section 8 for discussion). The results for English are included so that we can compare the relative performance of individual features in the two languages in question.

[3] However, it is worth noting that CO is not a useless feature. As table 3 shows, when 150 or fewer occurrences are

		SPEC	K	Eng.
BL		6.7	6.7	6.7
F1	SCF	42.4	39.3	57.8
F2	SCF(POS)	45.9	40.3	46.7
F3	SCF(PP)	**50.6**	36.9	63.3
F4	CO(4)	50.3	38.2	40.9
F5	CO(4+loc)	48.8	26.3	-
F6	CO(6)	52.7	29.2	-
F7	CO(6+loc)	**55.1**	33.8	-
F8	CO(8)	54.2	36.4	-
F9	CO(8+loc)	54.6	37.2	-
F10	LP(PREP)	35.5	32.8	49.0
F11	LP(SUBJ)	33.7	23.6	-
F12	LP(OBJ)	50.1	33.3	-
F13	LP(ALL)	**52.7**	40.1	74.6
F14	SCF+LP(SUBJ)	50.3	40.1	71.7
F15	SCF+LP(OBJ)	**54.5**	35.6	74.0
F16	SCF+LP(SUBJ+OBJ)	53.4	36.2	73.0
F17	SCF+SP	54.6	39.8	80.4

Table 2: Results for all the features for French (SPEC and K-means) and English (SPEC)

THR	Verbs	Cls	F3	F7	F13	F17
0	147	15	43.7	57.5	43.3	50.1
50	137	15	47.9	56.1	44.8	49.1
100	125	15	49.2	54.3	44.8	49.5
150	116	15	50.6	55.1	52.7	54.6
200	110	15	54.9	52.9	49.7	52.5
400	96	15	52.7	52.9	43.9	53.2
1000	71	15	51.4	54.0	44.8	54.5
2000	59	12	52.3	45.9	42.7	53.5
3000	51	12	55.7	49.0	46.8	59.2
4000	43	10	60.5	53.4	51.0	**65.4**

Table 3: The effect of verb frequency

tive feature. The only exception is the LP feature which performed better than CO in English.

7.2 Qualitative Evaluation

We conducted qualitative analysis of the clusters for French: those created using SPEC with F17 and F3. Verbs in the gold standard classes 29.2, 36.1, 37.3, 37.7 and 47.3 (Table 1) performed particularly well, with the majority of member verbs found in the same cluster. These verbs are ideal for clustering because they have distinctive syntactic-semantic characteristics. For example, verbs in 29.2 CHARACTERIZE class (e.g. *concevoir, considérer, dépeindre*) not only have a very specific meaning but they also take high frequency SCFs involving the preposition *comme* (Eng. *as*) which is not typical to many other classes. Interestingly, Levin classes 29.2, 36.1, 37.3, and 37.7 were among the best performing classes also in the supervised verb classification experiment of Sun et al. (2008) because these classes have distinctive characteristics also in English.

The benefit of sophisticated features which integrate also semantic (SP) information (F17) is particularly evident for classes with non-distinctive syntactic characteristics. For example, the intransitive verbs in 43.1 LIGHT EMISSION class (e.g. *briller, étinceler, flamboyer*) are difficult to cluster based on syntax only, but semantic features work because the verbs pose strong SPs on their subjects (entities capable of light emission). In the experiment of Sun et al. (2008), 43.1 was the worst performing class, possibly because no semantic features were used in the experiment.

The most frequent source of error is syntactic idiosyncracy. This is particularly evident for classes 10.1 REMOVE and 45.4 CHANGE OF STATE. Although verbs in these classes can take similar SCFs and alternations, only some of them are frequent in data. For example, the SCF *ôter X à Y* is frequent for verbs in 10.1, but not *ôter X de Y*. Although class 10.1 did not suffer from this problem in the English experiment of Sun et al. (2008), class 45.4 did. Class 45.4 performs particularly bad in French also because its member verbs are low in frequency.

Some errors are due to polysemy, caused partly by the fact that the French version of the gold standard was not controlled for this factor. Some verbs have their predominant senses in classes which are missing in the gold standard, e.g. the most frequent sense of *retenir* is *memorize*, not *keep* as in the gold standard class 13.5.1. GET.

Finally, some errors are not true errors but demonstrate the capability of clustering to learn novel information. For example, the CHANGE OF STATE class 45.4 includes many antonyms (e.g. *weaken* vs. *strenghten*). Clustering (using F17) separates these antonyms, so that verbs *adoucir, atténuer* and *tempérer* appear in one cluster and *consolider* and *renforcer* in another. Although these verbs share the same alternations, their SPs are different. The opposite effect can be observed when clustering maps together classes

available for a verb, CO outperforms all the other features in French, compensating for data sparseness.

which are semantically and syntactically related (e.g. 36.1 CORRESPOND and 37.7 SPEAK). Such classes are distinct in Levin and VerbNet, although should ideally be related. Cases such as these show the potential of clustering in discovering novel valuable information in data.

8 Discussion and Conclusion

When sufficient corpus data is available, there is a strong correlation between the types of features which perform the best in English and French. When the best features are used, many individual Levin classes have similar performance in the two languages. Due to differences in data sets direct comparison of performance figures for English and French is not possible. When considering the general level of performance, our best performance for French (65.4 F) is lower than the best performance for English in the experiment of Sun and Korhonen (2009). However, it does compare favourably to the performance of other state-of-the-art (even supervised) English systems (Joanis et al., 2008; Li and Brew, 2008; Ó Séaghdha and Copestake, 2008; Vlachos et al., 2009). This is impressive considering that we experimented with a fully unsupervised approach originally developed for another language.

When aiming to improve performance further, employing larger data is critical. Most recent experiments on English have employed bigger data sets, and unlike us, some of them have only considered the predominant senses of medium-high frequency verbs. As seen in section 7.1, such differences in data can have significant impact on performance. However, parser and feature extraction performance can also play a big role in overall accuracy, and should therefore be investigated further (Sun and Korhonen, 2009). The relatively low performance of basic LP features in French suggests that at least some of the current errors are due to parsing. Future research should investigate the source of error at different stages of processing. In addition, it would be interesting to investigate whether language-specific tuning (e.g. using language specific features such as auxiliary classes) can further improve performance on French.

Earlier works most closely related to ours are those of Merlo et al. (2002) and Ferrer (2004). Our results contrast with those of Ferrer who showed that a clustering approach does not transfer well from English to Spanish. However, she used basic SCF and named entity features only, and a clustering algorithm less suitable for high dimensional data. Like us, Merlo et al. (2002) created a gold standard by translating Levin classes to another language (Italian). They also applied a method developed for English to Italian, and reported good overall performance using features developed for English. Although the experiment was small (focussing on three classes and a few features only) and involved supervised classification, the results agree with ours.

These experiments support the linguistic hypothesis that Levin style classification can be cross-linguistically applicable. A clustering technique such as the one presented here could be used as a tool for investigating whether classifications are similar across a wider range of more diverse languages. From the NLP perspective, the fact that an unsupervised technique developed for one language can be applied to another language without the need for substantial tuning means that automatic techniques could be used to hypothesise useful Levin style classes for further languages. This, in turn, could facilitate the creation of multilingual VerbNets in the future.

9 Acknowledgement

Our work was funded by the Royal Society University Research Fellowship (AK), the Dorothy Hodgkin Postgraduate Award (LS), the EPSRC grants EP/F030061/1 and EP/G051070/1 (UK) and the EU FP7 project 'PANACEA'.

References

Omri Abend, Roi Reichart, and Ari Rappoport. A supervised algorithm for verb disambiguation into VerbNet classes. In *Proc. of COLING*, pages 9–16, 2008.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet Project. In *COLING-ACL*, pages 86–90, 1998.

Didier Bourigault, Marie-Paule Jacques, Cécile Fabre, Cécile Frérot, and Sylwia Ozdowska. Syntex, analyseur syntaxique de corpus. In *Actes des*

12èmes journées sur le Traitement Automatique des Langues Naturelles, 2005.

Chris Brew and Sabine Schulte im Walde. Spectral clustering for German verbs. In *Proc. of EMNLP*, pages 117–124, 2002.

Jinxiu Chen, Dong-Hong Ji, Chew Lim Tan, and Zheng-Yu Niu. Unsupervised relation disambiguation using spectral clustering. In *Proc. of COLING/ACL*, pages 89–96, 2006.

Hoa Trang Dang. *Investigations into the Role of Lexical Semantics in Word Sense Disambiguation*. PhD thesis, CIS, University of Pennsylvania, 2004.

Eva Esteve Ferrer. Towards a semantic classification of Spanish verbs based on subcategorisation information. In *Proc. of ACL Student Research Workshop*, 2004.

Ralph Grishman, Catherine Macleod, and Adam Meyers. Comlex syntax: building a computational lexicon. In *Proc. of COLING*, pages 268–272, 1994.

Maurice Gross. *Méthodes en syntaxe*. Hermann, Paris, 1975.

Eduard Hovy, Mitch Marcus, Martha Palmer, L. Ramshaw, and R. Weischedel. Ontonotes: The 90% solution. In *HLT/NAACL*, 2006.

Ray Jackendoff. *Semantic Structures*. The MIT Press, Cambridge, MA, 1990.

Eric Joanis, Suzanne Stevenson, and David James. A general feature space for automatic verb classification. *Nat. Lang. Eng.*, 14(3):337–367, 2008.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. A large-scale classification of English verbs. *Language Resources and Evaluation*, 42:21–40, 2008.

Karin Kipper-Schuler. *VerbNet: A broad-coverage, comprehensive verb lexicon*. University of Pennsylvania, PA, 2005.

Beth. Levin. English verb classes and alternations: A preliminary investigation. *Chicago, IL*, 1993.

Jianguo Li and Chris Brew. Which Are the Best Features for Automatic Verb Classification. In *Proc. of ACL*, pages 434–442, 2008.

Marina. Meila. The multicut lemma. Technical report, University of Washington, 2001.

Marina Meila and Jianbo Shi. A random walks view of spectral segmentation. In *AISTATS*, 2001.

Paola Merlo, Suzanne Stevenson, Vivian Tsang, and Gianluca Allaria. A multilingual paradigm for automatic verb classification. In *Proc. of ACL*, 2002.

Cédric Messiant. ASSCI : A subcategorization frames acquisition system for French. In *Proc. of ACL Student Research Workshop*, pages 55–60, 2008.

Cédric Messiant, Thierry Poibeau, and Anna Korhonen. LexSchem: a Large Subcategorization Lexicon for French Verbs. In *Proc. of LREC*, 2008.

George A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 1995.

Diarmuid Ó Séaghdha and Ann Copestake. Semantic classification with distributional kernels. In *Proc. of COLING*, pages 649–656, 2008.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 3(1):71–106, 2005.

Patrick Saint-Dizier. Verb Semantic Classes Based on 'alternations' and WordNet-like criteria . In P. Saint-Dizier, editor, *Predicative Forms in Natural language and lexical Knowledge Bases* , pages 247–279. Kluwer Academic, 1998.

Sabine Schulte im Walde. Experiments on the Automatic Induction of German Semantic Verb Classes. *Computational Linguistics*, 2006.

Lei Shi and Rada Mihalcea. Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing. In *Proc. of CICLing*, pages 100–111, 2005.

Lin Sun and Anna Korhonen. Improving verb clustering with automatically acquired selectional preferences. In *Proc. of EMNLP*, pages 638–647, 2009.

Lin Sun, Anna Korhonen, and Yuval Krymolowski. Verb class discovery from rich syntactic data. *LNCS*, 4919:16, 2008.

Yoshimi Suzuki and Fumiyo Fukumoto. Classifying Japanese Polysemous Verbs based on Fuzzy C-means Clustering. In *Proc. of TextGraphs-4*, pages 32–40, 2009.

Robert Swier and Suzanne Stevenson. Unsupervised semantic role labelling. In *Proc. of EMNLP*, 2004.

Gloria Vázquez, Ana Fernández, Irene Castellón, and M. Antonia Martí. Clasificación verbal: Alternancias de diátesis. In *Quaderns de Sintagma*. Universitat de Lleida, 2000.

Deepak Verma and Marina Meila. A comparison of spectral clustering algorithms. Technical report, Department of CSE University of Washington Seattle, 2005.

Andreas Vlachos, Anna Korhonen, and Zoubin Ghahramani. Unsupervised and Constrained Dirichlet Process Mixture Models for Verb Clustering. In *Proc. of the Workshop on on GEMS*, pages 74–82, 2009.

Ulrike von Luxburg. A tutorial on spectral clustering. *STAT COMPUT*, 17:395 – 416, 2007.

Piek Vossen. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, Dordrecht, 1998.

Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. *NIPS*, 17(1601-1608):16, 2004.

Semi-supervised dependency parsing using generalized tri-training

Anders Søgaard and Christian Rishøj
Center for Language Technology
University of Copenhagen
{soegaard|crjensen}@hum.ku.dk

Abstract

Martins et al. (2008) presented what to the best of our knowledge still ranks as the best overall result on the CONLL-X Shared Task datasets. The paper shows how triads of stacked dependency parsers described in Martins et al. (2008) can label unlabeled data for each other in a way similar to co-training and produce end parsers that are significantly better than any of the stacked input parsers. We evaluate our system on five datasets from the CONLL-X Shared Task and obtain 10–20% error reductions, incl. the best reported results on four of them. We compare our approach to other semi-supervised learning algorithms.

1 Introduction

Semi-supervised learning of structured variables is a difficult problem that has received considerable attention recently, but most results have been negative (Abney, 2008). This paper uses stacked learning (Wolpert, 1992) to reduce structured variables, i.e. dependency graphs, to multinomial variables, i.e. attachment and labeling decisions, which are easier to manage in semi-supervised learning scenarios, and which can later be combined into dependency trees using parsing algorithms for arc-factored dependency parsing. Our approach thus combines ensemble-based methods and semi-supervised learning.

Ensemble-based methods such as stacked learning are used to reduce the instability of classifiers, to average out their errors and to combine the strengths of diverse learning algorithms. Ensemble-based methods have attracted a lot of attention in dependency parsing recently (Sagae and Lavie, 2006; Hall et al., 2007; Nivre and McDonald, 2008; Martins et al., 2008; Fishel and Nivre, 2009; Surdeanu and Manning, 2010). Nivre and McDonald (2008) were first to introduce stacking in the context of dependency parsing.

Semi-supervised learning is typically motivated by data sparseness. For many classification tasks in natural language processing, labeled data can be in short supply but unlabeled data is more readily available. Semi-supervised methods exploit unlabeled data in addition to labeled data to improve performance on classification tasks. If the predictions of a learner l on unlabeled data are used to improve a learner l' in semi-supervised learning, the robustness of learning will depend on the stability of l. Combining ensemble-based and semi-supervised methods may thus lead to more robust semi-supervised learning.

Ensemble-based and semi-supervised methods are some of the areas that receive most attention in machine learning today, but relatively little attention has been given to *combining* these methods (Zhou, 2009). Semi-supervised learning algorithms can be categorized with respect to the number of views, i.e. the number of feature sets, and the number of learners used to inform each other (Hady and Schwenker, 2008). Self-training and expectation maximization are perhaps the best known semi-supervised learning algorithms (Abney, 2008). They are both single-view and single-learner algorithms. Since there is thus only a single perspective on data,

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1065–1073,
Beijing, August 2010

selecting unlabeled data points with predictions is a difficult task. There is an imminent danger that the learner amplifies its previous mistakes, and while several techniques such as balancing and throttling have been developed to avoid such caveats, using single-view and single-learner algorithms often requires both caution and experience with the modeling task at hand.

Algorithms with multiple views on data are known to be more robust. This insight led to the development of co-training (Blum and Mitchell, 1998), a two-view method where views inform each other, but it also paved the way for the integration of ensemble-based and semi-supervised methods, i.e. for methods with multiple learners. It was mentioned that relatively little work has been devoted to this topic, but there are notable exceptions:

Bennett et al. (2003) generalized boosting to semi-supervised learning in a seminal paper, where the idea of iterative or recursive ensembles was also introduced. Li and Zhou (2005) introduce *tri-training*, a form of co-training that trains an ensemble of three learners on labeled data and runs them on unlabeled data. If two learners agree on their labeling of a data point, the data point is added to the labeled data of the third learner with the prediction of the first two. Didaci and Roli (2006) extend self-training and co-training to multiple learners. Li and Zhou (2007) generalize tri-training to larger ensembles of random trees. The technique is also known as co-forests. Hady and Schwenker (2008) generalize existing ensemble-based methods for semi-supervised learning scenarios; in particular they embed ensembles in a form of co-training that is shown to maintain the diversity of the ensemble over time. Milidiu and Duarte (2009) generalize boosting at start to semi-supervised learning.

This paper applies a generalization of tri-training to two classification problems, attachment and labeling. The attachment classifier's weights are used for arc-factored dependency parsing, and the labeling classifier's weights are then used to label the dependency tree delivered by the parser.

Semi-supervised dependency parsing has attracted a lot of attention recently (Koo et al., 2008; Wang et al., 2008; Suzuki et al., 2009), but there has, to the best of our knowledge, been no previous attempts to apply tri-training or related combinations of ensemble-based and semi-supervised methods to any of these tasks, except for the work of Sagae and Tsujii (2007) discussed in Sect. 2.6. However, tri-training has been applied to Chinese chunking (Chen et al., 2006), question classification (Nguyen et al., 2008) and POS tagging (Søgaard, 2010).

We compare generalized tri-training to other semi-supervised learning algorithms, incl. self-training, the original tri-training algorithm based on bootstrap samples (Li and Zhou, 2005), co-forests (Li and Zhou, 2007) and semi-supervised support vector machines (Sindhwani and Keerthi, 2006).

Sect. 2 introduces dependency parsing and stacked learning. Stacked learning is generalized to dependency parsing, and previous work is briefly surveyed. We then describe how stacked dependency parsers can be further stacked as input for two end classifiers that can be combined to produce dependency structures. These two classifiers will learn multinomial variables (attachment and labeling) from a combination of labeled data and unlabeled data using a generalization of tri-training. Sect. 3 describes our experiments. We describe the data sets, and how the unlabeled data was prepared. Sect. 4 presents our results. Sect. 5 presents an error analysis and discusses the results in light of other results in the literature, and Sect. 6 concludes the paper.

2 Background and related work

2.1 Dependency parsing

Dependency parsing models a sentence as a tree where words are vertices and grammatical functions are directed edges (dependencies). Each word thus has a single incoming edge, except one called the root of the tree. Dependency parsing is thus a structured prediction problem with trees as structured variables. Each sentence has exponentially many possible dependency trees. Our observed variables are sentences with words labeled with part-of-speech tags. The task for each sentence is to find the dependency tree that maximizes an objective function which in our case is learned from a combination of labeled and unlabeled data.

More formally, a dependency tree for a sentence $x = w_1, \ldots, w_n$ is a tree $T = \langle\{0, 1, \ldots, n\}, A\rangle$ with $A \subseteq V \times V$ the set of dependency arcs. Each vertex corresponds to a word in the sentence, except 0 which is the root vertex, i.e. for any $i \leq n$ $\langle i, 0\rangle \notin A$. Since a dependency tree is a tree it is acyclic. A tree is projective if every vertex has a continuous projection, i.e. if and only if for every arc $\langle i, j\rangle \in A$ and node $k \in V$, if $i < k < j$ or $j < k < i$ then there is a subset of arcs $\{\langle i, i_1\rangle, \langle i_1, i_2\rangle, \ldots, \langle i_{k-1}, i_k\rangle\} \in A$ such that $i_k = k$.

In this paper we use a maximum spanning tree algorithm, the so-called Chu-Liu-Edmonds algorithm (CLE) (Edmonds, 1967) to turn the predictions of our semi-supervised classifiers into a dependency tree.

2.2 Stacked learning

Stacked generalization, or simply *stacking*, was first proposed by Wolpert (1992). Stacking is an ensemble-based learning method where multiple weak classifiers are combined in a strong end classifier. The idea is to train the end classifier directly on the predictions of the input classifiers.

Say each input classifier c_i with $1 \leq i \leq n$ receives an input $\mathbf{x}$ and outputs a prediction $c_i(\mathbf{x})$. The end classifier then takes as input $\langle\mathbf{x}, c_1(\mathbf{x}), \ldots, c_n(\mathbf{x})\rangle$ and outputs a final prediction $c_0(\langle\mathbf{x}, c_1(\mathbf{x}), \ldots, c_n(\mathbf{x})\rangle)$. Training is done by cross-validation. In sum, stacking is training a classifier on the output of classifiers.

2.3 Stacked dependency parsing

Stacked learning can be generalized to structured prediction tasks such as dependency parsing. Architectures for stacking dependency parsers typically only use one input parser, but otherwise the intuition is the same: the input parser is used to augment the dependency structures that the end parser is trained and evaluated on.

Nivre and McDonald (2008) first showed how the MSTParser (McDonald et al., 2005) and the MaltParser (Nivre et al., 2007) could be improved by stacking each parser on the predictions of the other. Martins et al. (2008) generalized their work, considering more combinations of parsers, and stacking the end parsers on non-local features from the predictions of the input parser, e.g. siblings and grand-parents. In this work we use three stacked dependency parsers for each language: mst2 (p_1), malt/mst2 (p_2) and malt/mst1 (p_3).

The notation ”malt/mst2” means that the second-order MSTParser has been stacked on the MaltParser. The capital letters refer to feature configurations. Configuration D stacks a level 1 parser on several (non-local) features of the predictions of the level 0 parser (along with the input features): the predicted edge, siblings, grand parents and predicted head of candidate modifier if predicted edge is 0. Configuration E stacks a level 1 parser on the features in configuration D and all the predicted children of the candidate head. The chosen parser configurations are those that performed best in Martins et al. (2008) across the different datasets.

2.4 Stacking stacked dependency parsing

The input features of the input classifiers in stacked learning $\mathbf{x}$ can of course be removed from the input of the end classifier. It is also possible to stack stacked classifiers. This leaves us with four strategies for recursive stacking; namely to constantly augment the feature set, with level n classifiers trained on the predictions of the classifiers at all $n - 1$ lower levels with or without the input features $\mathbf{x}$, or simply to train a level n classifier on the predictions of the level $n - 1$ classifiers with or without $\mathbf{x}$.

In this work we stack stacked dependency parsers by training classifiers on the output of three stacked dependency parsers and POS tags. Consequently, we use one of the features from $\mathbf{x}$. Note that we train classifiers and not parsers on this new level 2.

The reduction is done the following way: First we train a classifier on the relative distance from a word to its head to induce attachments. For example, we may obtain the following features from the predictions of our level 1 parsers:

label	p_1	p_2	p_3	POS
1	1	-1	1	NNP
0	0	0	0	VBD

In the second row all input parsers, p_{1-3} in columnsaa 2–4, agree that the verb is the root of the sentence. Column 1 tells us that this is correct. In the first row, two out of three parsers agree on attaching the noun to the verb, which again is correct. We train level 2 classifiers on feature vectors produced this way. Note that oracle performance of the ensemble is no upper bound on the accuracy of a classifier trained on level 1 predictions this way, since a classifier may learn the right decision from three wrong predictions and a POS tag.

Second we train a classifier to predict dependency relations. Our feature vectors are similar to the ones just described, but now contain dependency label predictions, e.g.:

label	p_1	p_2	p_3	POS
SBJ	SBJ	SBJ	SBJ	NN
ROOT	ROOT	ROOT	COORD	VBN

2.5 Generalized tri-training

Tri-training was originally introduced in Li and Zhou (2005). The method involves three learners that inform each other.

Let L denote the labeled data and U the unlabeled data. Assume that three classifiers c_1, c_2, c_3 have been trained on L. In the original algorithm, the three classifiers are obtained by applying the same learning algorithm to three bootstrap samples of the labeled data; but in generalized algorithms, three different learning algorithms are used. An unlabeled datapoint in U is labeled for a classifier, say c_1, if the other two classifiers agree on its label, i.e. c_2 and c_3. Two classifiers inform the third. If the two classifiers agree on a labeling, we assume there is a good chance that they are right. In the original algorithm, learning stops when the classifiers no longer change; in generalized tri-training, a fixed stopping criterion is used. The three classifiers are combined by voting. Li and Zhou (2005) show that under certain conditions the increase in classification noise rate is compensated by the amount of newly labeled data points.

The most important condition is that the three classifiers are diverse. If the three classifiers are identical, tri-training degenerates to *self-training*. As already mentioned, Li and Zhou (2005) obtain this diversity by training classifiers on bootstrap samples. In their experiments, they consider classifiers based on decision trees, BP neural networks and naïve Bayes inference.

```
for $i \in \{1..3\}$ do
    $c_i \leftarrow train\_classifier(l_i, L)$
end for
repeat
    for $i \in \{1..3\}$ do
        for $x \in U$ do
            $L_i \leftarrow \emptyset$
            if $c_j(x) = c_k(x) (j, k \neq i)$ then
                $L_i \leftarrow L_i \cup \{(x, c_j(x))\}$
            end if
        end for
        $c_i \leftarrow train\_classifier(l_i, L \cup L_i)$
    end for
until stopping criterion is met
apply $c_1$
```

Figure 1: Generalized tri-training.

In this paper we generalize the tri-training algorithm and use three different learning algorithms rather than bootstrap samples to create diversity: a naïve Bayes algorithm (no smoothing), random forests (Breiman, 2001) (with 100 unpruned decision trees) and an algorithm that induces unpruned decision trees. The overall algorithm is sketched in Figure 1 with l_i a learning algorithm.

Our weights are those of the random forest classifier after a fixed number of rounds. The attachment classifier iterates once over the unlabeled data, while the dependency relations classifier uses three iterations. The optimal number of iterations could of course be estimated on development data instead. Given the weights for an input sentence we use CLE to find its most likely dependency tree.

2.6 Related work

This paper uses stacking rather than voting to construct ensembles, but voting has been more widely used in dependency parsing than stacking. Voting was first introduced in dependency parsing in Zeman and Zabokrtsky (2005). Sagae and Lavie (2006) later used weighted voting and reparsing, i.e. using CLE to find the dependency tree that reflects the maximum number of votes. They also showed that binning the vote over part-of-speech tags led to further improvements. This set-up was adopted by Hall et al. (2007) in the best performing system in the CONLL 2007 Shared Task. Fishel and Nivre (2009) later experimented with binning the vote on other features with modest improvements.

Semi-supervised dependency parsing has only recently been explored, and failures have been more frequent than successes. There are, however, noteable exceptions such as Koo et al. (2008), Wang et al. (2008), Suzuki et al. (2009) and Sagae and Gordon (2009).

The semi-supervised methods employed in these experiments are very different from more traditional scenarios such as self-training and co-training. Two approaches (Koo et al., 2008; Sagae and Gordon, 2009) use clusters obtained from large amounts of unlabeled data to augment their labeled data by introducing new features, and two approaches (Wang et al., 2008; Suzuki et al., 2009) combine probability distributions obtained from labeled data with probability distributions obtained from unlabeled data.

Successes with self-training and co-training are rare, and several authors report negative results, e.g. Spreyer and Kuhn (2009). A notable exception in constituent-based parsing is the work of McClosky et al. (2006) who show that self-training is possible if a reranker is used to inform the underlying parser.

Sagae and Tsujii (2007) participated in (and won) the CONLL 2007 Shared Task on domain adaptation. They first trained a maximum entropy-based transition-based dependency parser on the out-of-domain labeled data and an SVM-based transition-based dependency parser on the *reversed* out-of-domain labeled data. The two parsers parse the in-domain labeled data (reversed, in the case of the SVM-based parser). Identical analyses are added to the original training set. The first parser is retrained and used to parse the test data. In sum, the authors do one round of co-training with the following selection criterion: If the two parsers produce the same dependency structures for a sentence, the dependency structure is added to the labeled data. This criterion is also the selection criterion in tri-training.

3 Experiments

3.1 Data

We use five datasets from the CONLL-X Shared Task (Buchholz and Marsi, 2006).[1] Lemmas and morphological features (FEATS) are ignored, since we only add POS and CPOS tags to unlabeled data. For German and Swedish, we use 100,000 sentences from the Leipzig Corpora Collection (Biemann et al., 2007) as unlabeled data. For Danish, Dutch, and Portuguese we use 100,000 sentences from the Europarl corpus (Koehn, 2005). The data characteristics are provided in Figure 2. The unlabeled data were POS tagged using the freely available SVMTool (Gimenez and Marquez, 2004) (model 4, left-right-left).

3.2 Algorithm

Once our data has been prepared, we train the stacked dependency parsers and use them to label training data for our classifiers (∼4,000 tokens), our test data and our unlabeled data. This gives us three sets of predictions for each of the three data sets. Using the features described in Sect. 2.4 we then construct data for training our two triads of classifiers (for attachment and dependency relations). The entire architecture can be depicted as in Figure 3.

We first stack three dependency parsers as described in Martins et al. (2008). We then stack three classifiers on top of these dependency parsers (and POS tags): a naïve Bayes classifier, a random forest, and a decision tree. Finally,

[1] The CONLL-X Shared Task consists of 12 datasets, but we did not have consistently tokenized unlabeled data for Arabic, Chinese, Japanese, Slovene and Turkish. Martins et al. (2008) ignore Czech. Our experiment with the Spanish dataset crashed unexpectedly. We will post results on the website as soon as possible.

		tokens	sents	tokens/sents	POSs	DEPRELs
Danish	train	94,386	5,190	18.2	24	52
	unl (Europarl)	2,422,144	100,000	24.2	-	-
	test	5,852	322	18.2	-	-
Dutch	train	195,069	13,349	14.6	13	26
	unl (Europarl)	2,336,176	100,000	23.4	-	-
	test	5,585	386	14.5	-	-
German	train	699,610	39,216	17.8	52	46
	unl (LCC)	1,763,281	100,000	17.6	-	-
	test	5,694	357	15.9	-	-
Portuguese	train	206,678	9,071	22.3	21	55
	unl (Europarl)	2,882,967	100,000	28.8	-	-
	test	5,867	288	22.8	-	-
Swedish	train	191,467	11,042	17.4	37	56
	unl (LCC)	1,727,068	100,000	17.3	-	-
	test	5,656	389	14.5	-	-

Figure 2: Characteristics of the data sets.

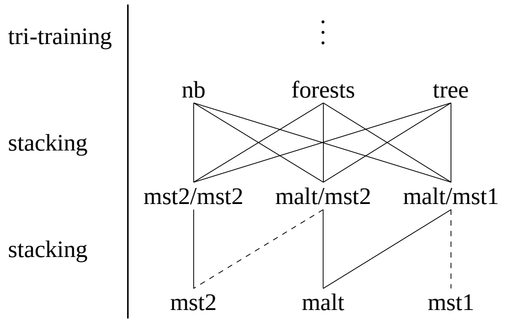


Figure 3: Tri-training stacked classifiers.

we tri-train these three stacked classifiers and for each test sentence output the weights provided by the random forest classifier. These weights are used to find the best possible dependency tree using CLE.

3.3 Baselines

The best of the stacked input parsers is of course our natural baseline.

Since we have generalized tri-training, we also compare generalized tri-training to the original tri-training algorithm based on bootstrap samples. The original tri-training algorithm is run with the same decomposition and the same features as our generalized tri-training algorithm. We use the learning algorithm originally used in Li and Zhou (2005), namely C4.5. We also compare our results to self-training (no pool, no growth rate) and co-forests (Li and Zhou, 2007). Finally, we compare our results to semi-supervised support vector machines (S3VMs) (Sindhwani and Keerthi, 2006). Since S3VMs produce binary classifiers, and one-vs.-many combination would be very time-consuming, we train a binary classifier that produces a probability that any candidate arc is correct and do greedy head selection. We optimized the feature set and included a total of seven features (head POS, dependent POS, dependent left neighbor POS, distance+direction, predictions of the three classifiers).

4 Results

Our results are presented in Figure 4. Labeled (LAS) and unlabeled attachment scores (UAS) and labeling accuracy (LA) are defined as usual and include punctuation signs unless otherwise noted. Difference (Δ) in LAS, error reduction and p-value compare our results to the best input stacked parser (malt/mst2, excerpt for Swedish).

Generalized tri-training (tri-training-CLE), i.e. using CLE to find the best well-formed dependency trees given the weights provided by our tri-trained random forest classifier, leads to highly significant improvements on *all* data sets ($p < 0.001$) with an average error reduction of 14,9%. The results for the other semi-supervised learning algorithms are presented in Figure 5. We only used 10% of the unlabeled data (10k sentences) in this experiment and only did unlabeled parsing, but it is quite evident that these learning strategies seem less promising than gen-

Danish	LAS(%)	UAS(%)	LA(%)	EM(%)	Δ LAS	err.red(%)	p-value
mst2	84.64	89.11	91.35	24.84			
malt/mst2	86.36	90.50	92.09	27.64			
malt/mst1	86.11	90.23	91.87	25.78			
tri-training-CLE	**87.76**	**92.11**	**92.87**	**27.95**	1.40	10.26	$<$0.0001
tri-training-CLE (excl. pnc.)	87.54	92.61	91.68				
CONLL-X best (excl. pnc.)	84.79	90.58	89.22				
Martins et al. (excl. pnc.)	86.79	-	-				
Dutch							
mst2	80.27	84.32	84.96	23.32			
malt/mst2	81.00	84.58	85.46	24.35			
malt/mst1	80.72	84.17	85.34	26.17			
tri-training-CLE	**83.42**	**88.18**	**87.82**	**28.00**	2.42	12.74	$<$0.0001
tri-training-CLE (excl. pnc.)	81.73	86.97	86.61				
CONLL-X best (excl. pnc.)	79.19	83.57	83.89				
Martins et al. (excl. pnc.)	81.61	-	-				
German							
mst2	87.32	89.88	93.05	35.85			
malt/mst2	88.06	90.53	93.52	40.06			
malt/mst1	88.04	90.50	93.48	38.10			
tri-training-CLE	**90.41**	**93.22**	**94.61**	**43.14**	2.35	19.68	$<$0.0001
tri-training-CLE (excl. pnc.)	90.30	93.49	93.87				
CONLL-X best (excl. pnc.)	87.34	90.38	92.11				
Martins et al. (excl. pnc.)	88.66	-	-				
Portuguese							
mst2	84.83	88.44	92.04	25.69			
malt/mst2	85.39	88.80	92.59	28.13			
malt/mst1	85.00	88.39	92.23	25.69			
tri-training-CLE	**88.03**	**91.89**	**93.54**	**29.86**	2.64	18.07	$<$0.0001
tri-training-CLE (excl. pnc.)	89.18	93.69	92.43				
CONLL-X best (excl. pnc.)	87.60	91.36	91.54				
Martins et al. (excl. pnc.)	88.46	-	-				
Swedish							
mst2	81.82	87.36	87.29	27.76			
malt/mst2	84.42	89.57	88.68	31.62			
malt/mst1	84.74	89.83	89.07	31.11			
tri-training-CLE	**86.83**	**92.04**	**90.65**	**32.65**	2.09	13.70	$<$0.0001
tri-training-CLE (excl. pnc.)	86.66	92.45	89.58				
CONLL-X best (excl. pnc.)	84.58	89.50	87.39				
Martins et al. (excl. pnc.)	85.16	-	-				
AV					2.18	14.89	

Figure 4: Results on CONLL-X datasets. Scores are **including punctuation** unless otherwise noted. Δ and p-value is difference with respect to best input parser.

UAS	malt-mst2	S3VMs	self-training	orig-tri-training	co-forests	tri-training	tri-training[full]
Danish	90.50	90.47	89.68	89.66	88.79	**90.60**	92.21
Dutch	84.58	85.34	84.06	83.83	83.97	**86.07**	88.06
German	90.53	90.15	89.83	89.92	88.47	**90.81**	93.20
Portuguese	88.80	65.64	87.60	87.62	87.06	**89.16**	91.87
Swedish	89.83	81.46	89.09	89.20	88.65	**90.22**	92.24
AV	88.80	82.61	88.05	88.05	87.44	**89.37**	91.52

Figure 5: Comparison of different semi-supervised learning algorithms (10% of unlabeled data) using 2-fold CV and no reparsing, UAS **including punctuation**.

eralized tri-training.

5 Error analysis and discussion

Error reductions are higher with dependencies to the root node and long distance dependencies than with local dependencies. The table below lists the labeled attachment F_1-scores for the five datasets binned on dependency length. The average error reduction is the same for root dependencies and long distance dependencies (length $>$7), but significantly lower for local dependencies. This seems to indicate that large amounts of data are necessary for the parser to recover long distance dependencies.

	root	1	2	4–7	>7
Da(F_1)	98.45	96.21	92.09	88.17	90.93
– err.red	**41.34**	10.69	13.92	15.75	21.92
Du(F_1)	83.65	94.47	88.60	82.40	81.54
– err.red	28.39	16.74	20.72	17.00	**31.88**
Ge(F_1)	97.33	96.47	94.28	92.42	93.94
– err.red	26.65	19.77	17.46	25.25	**38.97**
Po(F_1)	96.23	97.05	95.17	84.80	87.11
– err.red	22.47	19.56	24.86	22.56	**26.97**
Sw(F_1)	96.37	95.67	93.46	88.42	89.57
– err.red	**32.85**	14.10	15.04	25.97	31.50
AV err.red	**30.34**	16.17	18.40	21.31	30.25

Our results for Danish, Dutch, German and Portuguese are to the best of our knowledge the best reported results in the literature. Zhang and Chan (2009) obtain a LAS of 87.20 for Swedish with transition-based parsing based on reinforcement learning. They evaluate their system on a subset of the CONLL-X datasets and obtain their (by far) best improvement on the Swedish dataset. They speculate that ”the reason might be that [long distance dependencies] are not popular in Swedish”. Since our parser is particularly good at long distance dependencies, this may also explain why a supervised parser outperforms our system on this dataset. Interestingly, our unlabeled attachment score is a lot better than the one reported by Zhang and Chan (2009), namely 92.45 compared to 91.84.

Generally, our UASs are better than our LASs. Since we separate attachment and labeling out in two independent steps, improvements in UAS and improvements in LA do not necessarily lead to improvements in LAS. While our average error reduction in LAS is 14.9%, our average error reductions in UAS is 23.6%. The average error reduction in LA is 14.0%. In two-stage dependency parsers or dependency parsers with joint models, improvements in UAS are typically followed by comparable improvements in LAS.

6 Conclusion

This paper showed how the stacked dependency parsers introduced in Martins et al. (2008) can be improved by inference from unlabeled data. Briefly put, we stack three diverse classifiers on triads of stacked dependency parsers and let them label unlabeled data for each other in a co-training-like architecture. Our average error reductions in LAS over the best of our stacked input parsers is 14.9%; in UAS, it is 23.6%. The code is available at http://cst.dk/anders/tridep.html.

References

Abney, Steven. 2008. *Semi-supervised learning for computational linguistics*. Chapman & Hall.

Bennett, Kristin, Ayhan Demiriz, and Richard Maclin. 2003. Exploiting unlabeled data in ensemble methods. In *KDD*.

Biemann, Chris, G. Heyer, U. Quasthoff, and M. Richter. 2007. The Leipzig corpora collection. In *Corpus Linguistics*.

Blum, Avrim and Tom Mitchell. 1998. Combining labeled and unlabeled with-co-training. In *COLT*.

Breiman, Leo. 2001. Random forests. *Machine Learning*, 45:5–32.

Buchholz, Sabine and Erwin Marsi. 2006. CONLL-X shared task on multilingual dependency parsing. In *CONLL*.

Chen, Wenliang, Yujie Zhang, and Hitoshi Isahara. 2006. Chinese chunking with tri-training learning. In *Computer processing of oriental languages*, pages 466–473. Springer, Berlin, Germany.

Didaci, Luca and Fabio Roli. 2006. Using co-training and self-training in semi-supervised multiple classifier systems. In *SSPR& SPR*, pages 522–530. Springer, Berlin, Germany.

Edmonds, J. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71:233–240.

Fishel, Mark and Joakim Nivre. 2009. Voting and stacking in data-driven dependency parsing. In *NODALIDA*.

Gimenez, Jesus and Lluis Marquez. 2004. SVM-Tool: a general POS tagger generator based on support vector machines. In *LREC*.

Hady, Mohamed and Friedhelm Schwenker. 2008. Co-training by committee. *International Journal of Software and Informatics*, 2:95–124.

Hall, Johan, Jens Nilsson, Joakim Nivre, Gulsen Eryigit, Beata Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? In *CONLL*.

Koehn, Philipp. 2005. Europarl: a parallel corpus for statistical machine translation. In *MT-Summit*.

Koo, Terry, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *ACL*.

Li, Ming and Zhi-Hua Zhou. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.

Li, Ming and Zhi-Hua Zhou. 2007. Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. *IEEE Transactions on Systems, Man and Cybernetics*, 37(6):1088–1098.

Martins, André, Dipanjan Das, Noah Smith, and Eric Xing. 2008. Stacking dependency parsers. In *EMNLP*.

McClosky, David, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *HLT-NAACL*.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *HLT-EMNLP*.

Milidiu, Ruy and Julio Duarte. 2009. Improving BAS committee performance with a semi-supervised approach. In *European Symposium on Artificial Neural Networks*.

Nguyen, Tri, Le Nguyen, and Akira Shimazu. 2008. Using semi-supervised learning for question classification. *Journal of Natural Language Processing*, 15:3–21.

Nivre, Joakim and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *ACL-HLT*.

Nivre, Joakim, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser. *Natural Language Engineering*, 13(2):95–135.

Sagae, Kenji and Andrew Gordon. 2009. Clustering words by syntactic similarity improves dependency parsing of predicate-argument structures. In *IWPT*.

Sagae, Kenji and Alon Lavie. 2006. Parser combination by reparsing. In *HLT-NAACL*.

Sagae, Kenji and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with lr models and parser ensembles. In *EMNLP-CONLL*.

Sindhwani, Vikas and Sathiya Keerthi. 2006. Large scale semi-supervised linear SVMs. In *ACM SIGIR*.

Søgaard, Anders. 2010. Simple semi-supervised training of part-of-speech taggers. In *ACL*.

Spreyer, Kathrin and Jonas Kuhn. 2009. Data-driven dependency parsing of new languages using incomplete and noisy training data. In *CONLL*.

Surdeanu, Mihai and Christopher Manning. 2010. Ensemble models for dependency parsing: cheap and good? In *NAACL*.

Suzuki, Jun, Hideki Isozaki, Xavier Carreras, and Michael Collins. 2009. Semi-supervised convex training for dependency parsing. In *EMNLP*.

Wang, Qin, Dekang Lin, and Dale Schuurmans. 2008. Semi-supervised convex training for dependency parsing. In *ACL*.

Wolpert, David. 1992. Stacked generalization. *Neural Networks*, 5:241–259.

Zeman, Daniel and Zdeněk Žabokrtský. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *IWPT*.

Zhang, Lidan and Kwok Chan. 2009. Dependency parsing with energy-based reinforcement learning. In *IWPT*.

Zhou, Zhi-Hua. 2009. When semi-supervised learning meets ensemble learning. In *MCS*.

SemanticRank: Ranking Keywords and Sentences Using Semantic Graphs

George Tsatsaronis [1] and **Iraklis Varlamis** [2] and **Kjetil Nørvåg** [1]
[1]Department of Computer and Information Science,
Norwegian University of Science and Technology
[2] Department of Informatics and Telematics, Harokopio University of Athens

Abstract

The selection of the most descriptive terms or passages from text is crucial for several tasks, such as feature extraction and summarization. In the majority of the cases, research works propose the ranking of all candidate keywords or sentences and then select the top-ranked items as features, or as a text summary respectively. Ranking is usually performed using statistical information from text (i.e., frequency of occurrence, inverse document frequency, co-occurrence information). In this paper we present *SemanticRank*, a graph-based ranking algorithm for keyword and sentence extraction from text. The algorithm constructs a *semantic graph* using implicit links, which are based on semantic relatedness between text nodes and consequently ranks nodes using different ranking algorithms. Comparative evaluation against related state of the art methods for keyword and sentence extraction shows that *SemanticRank* performs favorably in previously used data sets.

1 Introduction

Graph based ranking algorithms can be very helpful when searching for important pages in the World Wide Web, members in a social network, or authors in a publication database. Such algorithms capitalize on the existence of explicit links (e.g., hyperlinks, citations) between the graph vertices. In the case of flat text collections, neither links nor citations exist, so the need to devise implicit edges between text keywords or sentences arises. One feasible solution is to exploit the contextual information of terms and create semantic graphs from text based on content similarity. However, conceptual analysis of text has a strong potential in this direction, since it reveals latent similarities between text segments that discuss the same subject but with different terminology. In this direction, we introduce *SemanticRank*, a new algorithm for ranking text segments. *SemanticRank* comprises two steps: (a) creation of semantic graphs from text using both semantic and statistical information, and (b) application of a graph-based ranking algorithm that exploits the edges' weights.

The key contributions of this work are: (1) a novel method for the construction and weighting of the semantic graph, which contains text segments (terms or sentences) as nodes and weighted edges that capture the semantic relatedness (i.e., relatedness in meaning) between nodes but also consider statistical information, (2) the modular design of the method, which allows any graph-based ranking algorithm to be employed, and (3) thorough experimental evaluation of *SemanticRank* in the keyword extraction and text summarization tasks, and evaluation of several alternatives for the graph-based ranking component.

The paper is organized as follows: Section 2 discusses related work. Section 3 presents the preliminaries of *SemanticRank*: the semantic relatedness measure, the graph creation process and the ranking algorithm alternatives. Section 4 provides the details of our method. Section 5 presents the experimental evaluation of *SemanticRank* in two different tasks: keyword extraction and text summarization. Finally, Section 6 concludes and provides pointers to future work.

2 Related Work

This work addresses the problem of extracting the most representative keywords and sentences from text as a means of text summarization. More specifically, *SemanticRank* capitalizes on the creation of term and sentence graphs from text and on graph-based ranking algorithms in order to support the following tasks: (a) keyword extraction from text, which is performed by selecting the top-ranked terms as the most representative ones, and (b) text summarization, which is done by selecting the top ranked sentences as the most representative ones. For this reason, a survey of research works in keyword extraction and text summarization, with emphasis on graph-based approaches is necessary to understand the requirements for these tasks, to locate benchmark data sets, and state of the art graph-based approaches for the comparative evaluation.

2.1 Keyword Extraction

Keyword extraction is an important task in document indexing, and strongly affects the performance of re-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1074–1082,
Beijing, August 2010

trieval, classification clustering, and summarization. Most keyword extraction approaches rely on statistical measures such as term frequency (TF), inverse document frequency (IDF), and variations (Aizawa, 2003). Several works in keyword extraction construct semantic networks from text in order to capture the implicit relations between the individual candidates. Huang et al. (2006) propose the construction of one semantic network per document, and use edges that capture syntactic relations between document terms. Mihalcea and Tarau (2004) suggest a semantic network model where edges express the co-occurrence of terms in the document's sentences. Wang et al. (2007) employ the well known *PageRank* algorithm to perform word sense disambiguation (WSD) and keyword extraction from documents. The graphs that they construct are always subgraphs of the *WordNet* thesaurus[1], which results in low text coverage.

In this work we aim at the design and implementation of a keyword extraction algorithm that takes into account different aspects of text, such as statistical information and the semantic relatedness between keywords. To the best of our knowledge, the current work is the firs to propose a semantic network construction model for keyword extraction based on measures of semantic relatedness between keywords. In Section 3 we discuss the employed measure of semantic relatedness, which utilizes both *WordNet* and *Wikipedia*[2] to increase term coverage.

2.2 Text Summarization

The aim of automatic text summarization is to generate a summary of a pre-specifie length for a given input text. The *Document Understanding Conference* (DUC) [3] series have provided benchmark data sets with documents and manually generated summaries, which can be used for the evaluation of any automatic text summarizer. Research works in this area conduct automatic text summarization by selecting the most important sentences from the input texts (Steinberger and Jezek, 2009). Baseline methods are based on the observation that important sentences inside a text usually occur at its beginning. Thus, a straightforward baseline is to select the firs k sentences as a summary of the text, and setting k in such a way that does not violate the summary length restriction.

Another important class of automated text summarization methods is that of cohesion-based methods. Such methods assume that the important sentences or paragraphs of a given text document are the most connected entities in more or less elaborate semantic structures inside the text. In this direction, the methods construct a graph for each text document, with the vertices being the document sentences, and attempt to determine the most connected vertices in each graph. These methods are further classifie depending on how the graph's edges are constructed, for example using word co-occurrences (Salton et al., 1997), local salience and grammatical relations (Boguarev and Kennedy, 1997), co-reference (Baldwin and Morton, 1998), and combinations of the aforementioned (Mani and Bloedorn, 1998).

More recently, some cohesion-based methods have attempted to capture the semantic similarity of sentences inside a text document, and rank sentences in the constructed semantic graph. For example in the method of Mihalcea and Tarau (2004), the graph contains a vertex for each sentence of the given text document, and weighted edges between sentence. The weights represent the semantic similarity between sentences and are actually the contextual overlap between the sentences' terms. The *PageRank* algorithm is then applied to rank sentences in each of the constructed semantic graphs. In the method of Litvak and Last (2008), the vertices are again the sentences of the given text document and the edges represent the syntactic relations between them. Finally, the *HITS* algorithm is applied on the graph for ranking the sentences.

The conclusion from the literature review is that modern trends in graph-based approaches focus on novel methodologies for weighting edges and constructing semantic graphs, and employ standard techniques for ranking vertices, such as *PageRank*, *HITS*, or variations. Another important findin is that the potentiality of creating the edges between the vertices based on measures of semantic relatedness among the respective nodes is unexploited so far, and this is the core of the current work. Thus, the main difference between *SemanticRank* and the aforementioned approaches is that our edge weighting method employs a measure of semantic relatedness between sentences, that is based on *WordNet* and *Wikipedia*. The motivation behind such a perspective is that such semantic graphs would capture the similarity in meaning among the graph vertices, which was neglected by previous approaches. Finally, regarding the data sets used for evaluation, most works in text summarization use past *DUC* data, whereas in the case of keyword extraction a subset from the *Inspec* bibliographic database has been used in several cases in the past (Mihalcea and Tarau, 2004; Hulth, 2003).

[1] http://wordnet.princeton.edu/

[2] http://wikipedia.org

[3] http://duc.nist.gov/. Recently it has been renamed to *Text Understanding Conference* (TAC).

3 Terminology and Preliminaries

A graph-based method for ranking keywords or sentences by constructing semantic graphs comprises two steps: (a) the creation of the semantic graph, with keywords or sentences as vertices, and edges constructed based on a semantic similarity measure between vertices (c.f. Section 3.2), and (b) the adaptation of a new or existent ranking algorithm which analyzes the graph structure and ranks the nodes. Section 3.1 introduces the used terminology. In Section 3.2 we explain how the firs step is done by *SemanticRank*, and in Section 3.3 we present several alternatives for the second step.

3.1 Terminology

In the following we denote with $T(t_i, t_j)$ a pair of terms that occur in text document T. We also assume that T is a member of a document collection D given as input to our method. O represents the used knowledge-base (e.g., thesaurus, dictionary); in our case we are using two such knowledge-bases, namely *WordNet* and *Wikipedia*. With $\mathit{SR}_O(t_i, t_j)$ we denote the semantic relatedness between terms t_i and t_j using O for its computation, and $\mathit{SRS}(A, B)$ represents the semantic relatedness between text segments A and B (e.g., documents, sentences).

Concerning *WordNet*, S_i (S_j) represents the set of the different meanings (senses) with which t_i (t_j) may appear in O. P_{ij} denotes the set of paths connecting senses in S_i with senses in S_j, as these may be found using O. P_{ij}^k represents one such path in the set of paths P_{ij}, namely the k_{th} path. S_{ij} stands for the set of all possible sense pairs between the set of senses S_i and the set of senses S_j. Thus, $|S_{ij}| = |S_i| * |S_j|$. Respectively, S_{ij}^m stands for one such combination, namely the m_{th} combination.

With regards to Wikipedia, W refers to all the *Wikipedia* articles. With a_i we denote the *Wikipedia* article for term t_i. $\mathit{In}(a_i)$ is the set of *Wikipedia* articles that contain at least one link to a_i.

Finally, if d_i is the i_{th} document of D and t_a a term in d_i, then we denote with $\mathit{TF\text{-}IDF}(t_a, d_i) = \frac{\mathit{Count}(t_a,d_i)}{|d_i|} \cdot \log_2 \frac{\mathit{Count}(t_a,D)+1}{|D|}$ the *TF-IDF* weight of t_a in d_i, where $|d_i|$ is the number of term occurrences in d_i, $|D|$ is the number of documents in D, $\mathit{Count}(t_a, d_i)$ the number of occurrences of t_a in d_i, and $\mathit{Count}(t_a, D)$ the number of documents in D that contain t_a.

3.2 Creating Semantic Graphs from Text

A huge volume of literature has been created on how to construct semantic graphs addressing various applications, such as word sense disambiguation (Agirre and Soroa, 2009), keyword and sentence extraction (Mihalcea and Tarau, 2004; Litvak and Last, 2008; Yeh et al., 2008), and computation of semantic relatedness or similarity between terms (Gabrilovich and Markovitch, 2007; Budanitsky and Hirst, 2006; Milne and Witten, 2008).

In this work we adopt a semantic graph construction method which is able to capture the semantic relatedness between terms, as well as text segments. For our purposes we adopt *Omiotis*, the measure proposed by Tsatsaronis et al. (2010) in order to construct and weigh the edges of the semantic graph. *Omiotis* is a knowledge-based measure of semantic relatedness that may capture the semantic relatedness between both keywords and text segments (e.g., sentences, documents), allowing us to construct both semantic keyword graphs for keyword extraction, and semantic sentence graphs for sentence extraction and summarization. Our selection also lies in the fact that *Omiotis* has been shown to perform very well compared to other known measure of semantic relatedness or similarity in tasks such as term-to-term similarity (Tsatsaronis et al., 2010; Budanitsky and Hirst, 2006). However, since *Omiotis* relies solely in *WordNet*, we enhance the coverage of *SemanticRank* by complementing the edge weighting with an additional *Wikipedia*-based measure, namely the measure proposed by Milne and Witten (*MLN*) (2008). In Section 3.2.1 we explain how these two measures are combined in order to compute the semantic relatedness between terms, and in Section 3.2.2 we explain how semantic relatedness is captured between sentences.

3.2.1 Semantic Relatedness Between Terms

The measure presented in (Tsatsaronis et al., 2010) define the semantic relatedness between a pair of terms as shown in Equation 1, where the knowledge-base *O* is *WordNet* (*WN*).

$$\mathit{SR}_{\mathit{WN}}(t_i,t_j)=\max_m\{\max_k\{\mathit{SCM}(S_{ij}^m,P_{ij}^k)\cdot\mathit{SPE}(S_{ij}^m,P_{ij}^k)\}\} \quad (1)$$

where *SCM* and *SPE* are called *Semantic Compactness* and *Semantic Path Elaboration* respectively. Their product measures the weight of the path connecting the two senses in S_{ij}^m, taking into account: the path length, the type of the semantic edges comprising it, and the depth of the intermediate nodes in the *WN* senses hierarchy. The semantic relatedness between two terms t_i, t_j, when $t_i \in \mathit{WN}$ and $t_j \notin \mathit{WN}$, or vice versa, is considered 0. The intuition behind Equation 1 is that the semantic relatedness between two terms should be computed based on the *highest value* path connecting any pair of senses of the two terms. The computation of the *value* takes into account in tandem all of the aforementioned factors.

In order to enhance the coverage of the measure in Equation 1, we combine it with the *WLM Wikipedia*-

based measure of Milne and Witten (2008), which is a low-cost solution for measuring relatedness between terms using the *Wikipedia* articles and link structure as a knowledge base. The semantic relatedness between two terms t_i and t_j according to *WLM* is defined as shown in Equation 2. The intuition behind this formula is that the semantic similarity between two terms becomes higher, as the number of articles pointing to both respective *Wikipedia* articles increases (i.e., as the percentage of the articles linking to both pages compared to the number of articles linking to either of them increases).

$$SR_{Wiki}(t_i,t_j)=\frac{log(\max\{|In(a_i)|,|In(a_j)|\})-log(|In(a_i)\cap In(a_j)|)}{log(|W|)-log(\min\{|In(a_i)|,|In(a_j)|\})} \tag{2}$$

We combine the two measures in a single measure $SRT(t_i, t_j)$, as shown in Equation 3. The reason we prioritize $SR_{WN}(t_i, t_j)$ from $SR_{Wiki}(t_i, t_j)$, when both terms exist in *WN*, is because the former measure has shown much better performance in capturing the semantic relatedness between terms.

$$SRT(t_i, t_j) = \begin{cases} 1, & t_i = t_j \\ SR_{WN}(t_i, t_j), & \text{if } t_i, t_j \in \textit{WordNet} \\ SR_{Wiki}(t_i, t_j), & \text{if } t_i, t_j \in \textit{Wikipedia} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

3.2.2 Semantic Relatedness Between Texts

To quantify the semantic relatedness for a pair of text segments, we build upon the *SRT* measure, but also take into account the statistical importance of the terms occurring in the respective texts. Given two text segments A and B, and two terms $t_a \in A$ and $t_b \in B$, a measure that combines the statistical importance of t_a and t_b, according to (Tsatsaronis et al., 2010), is the harmonic mean of their *TF-IDF* weights. We denote this quantity as λ_{t_a,t_b}. Then for each term $t_a \in A$, we search for the corresponding term $t_b \in B$, which we symbolize with b_*, that maximizes the product of their combined statistical importance and semantic similarity. In our case, b_* is found by Equation 4. Similarly we can find for each $t_b \in B$ the corresponding a_*.

$$b_* = \arg\max_{t_b \in B}\{\lambda_{t_a,t_b} \cdot SRT(t_a, t_b)\} \tag{4}$$

After finding the set of all b_* and a_* terms, the semantic relatedness between the two texts A and B is computed as shown in Equation 5.

$$SRS(A, B) = \frac{\theta(A, B) + \theta(B, A)}{2} \tag{5}$$

where $\theta(A, B) = \frac{1}{|A|}\sum_{t_a \in A} \lambda_{t_a,b_*} \cdot SRT(t_a, b_*)$, and $\theta(B, A)$ can be computed respectively. The measure in Equation 5 is the measure used by *SemanticRank* to construct the edges between sentence vertices in the case of the semantic sentence graphs for text summarization. Regarding which sense of each term is used for the computation of its semantic relatedness with any other term, the senses that maximize the measure in Equation 3 are picked in each case.

3.3 Ranking Nodes in Semantic Graphs

For the purposes of our experimentation we will be evaluating *SemanticRank* with variations of the known *PageRank* and *HITS* algorithms. Some of those variations are applied for the first time in the framework of ranking nodes in semantic graphs. However, as will be explained in Section 4, in *SemanticRank*, any available vertex ranking methodology can be used instead.

The original versions of *PageRank* and *HITS* rely on the *"rich get richer"* model, which is based on explicit links and ignores edges weights. More specifically, HITS prioritizes good hubs and authorities, whereas PageRank uses a dampening factor (β) in order to avoid clique attacks and promote the centrality of nodes. However, in the case of graphs with implicitly devised links, like in semantic graphs, the edges carry weights, which must be taken into account. In this direction, we employ a modified version of the original *PageRank* algorithm, first introduced by Mihalcea and Tarau (2004). The modified *PageRank* is shown in Equation 6.

$$WPR(i) = (1 - \beta) + \beta \cdot \sum_{j \in IN(i)} \frac{w_{ij} \cdot WPR(j)}{\sum_{k \in OUT(j)} w_{jk}} \tag{6}$$

where i, j, k represent vertices, $IN(i)$ and $OUT(j)$ are the sets of *inlink* nodes of i and *outlink* nodes of j respectively, and w_{ij} is the weight of the edge between nodes i and j. In the case of semantic graphs constructed for keyword extraction, nodes are terms, and, thus, $w_{ij} = SRT(t_i, t_j)$. In the case of semantic graphs constructed for text summarization, i and j are sentences, and, thus, $w_{ij} = SRS(i, j)$.

Similarly to the modification shown in Equation 6 for *PageRank*, we can define a weighted version of *HITS*. The respective *authority* and *hub* scores are shown in Equations 7 and 8.

$$authority(i) = \sum_{j \in In(i)} w_{i,j} \cdot hub(j) \tag{7}$$

$$hub(i) = \sum_{j \in Out(i)} w_{i,j} \cdot authority(j) \tag{8}$$

The aforementioned modifications have been already applied in the past in the case of semantic graphs, with application to keyword extraction and text summarization (Mihalcea and Tarau, 2004; Mihalcea, 2004), although using different semantic graphs. For the extraction of the most important nodes, the modified *PageRank* version is used to rank the nodes according to their final *PageRank* values, and the modified *HITS* to rank nodes according to their final *authority* values. In this work, we also consider and evaluate two additional modifications of *PageRank* in order to rank vertices in the case of the semantic keyword graphs. The first modification that we call *Averaged PageRank Weighting* (*APW*) is presented in Equation 9, and is used after the weighted *PageRank* of Equation 6 has executed. The intuition behind *APW* is that each vertex t_i in the case of the keyword semantic graphs, has a known importance based on its frequency of occurrence (*TF-IDF* weight) inside the given document collection D. Thus, *APW* considers both the importance of vertex t_i inside its semantic graph, and inside its document collection.

$$APW(t_i) = \frac{1}{2}\left(\frac{WPR(t_i)}{WPR_{max}} + \frac{TF\text{-}IDF(t_i, d_j)}{TF\text{-}IDF_{max}}\right) \quad (9)$$

where d_j the specific document from which the semantic keyword graph is created, WPR_{max} is the maximum *PageRank* score found in this graph, and $TF\text{-}IDF_{max}$ is the maximum *TF-IDF* weight found in document d_j.

The second *PageRank* modification that is employed for the first time in the case of semantic keyword graphs is the *priors biased PageRank* (*P-PR*) discussed in (White and Smyth, 2003). The idea is very similar to the works in (Haveliwala, 2002) and (Agirre and Soroa, 2009), and pertain to ranking the nodes in the graph, with regards to a given set of nodes called *priors*. In short, while *PageRank* provides a global ranking of the nodes in the graph, *P-PR* provides a ranking of the nodes with regards to the set of the given *prior* nodes. This is expressed in Equation 10. The only difference with equation 6 is that each node i has its own "*random jump*" probability to the *prior* nodes. Thus, for each node i, *P-PR* has a β_i, which expresses how often we may jump back to the set of the *prior* nodes from node i. The intuition behind *priors* is that certain nodes in the graph are favored against other. In a keyword extraction task the *priors* set may contain the keywords appearing in the document's title.

$$P\text{-}PR(i) = (1-\beta_i)+\beta_i \cdot \sum_{j \in IN(i)} \frac{w_{ij} \cdot P\text{-}PR(j)}{\sum_{k \in OUT(j)} w_{jk}} \quad (10)$$

4 SemanticRank

In this section we present *SemanticRank* (illustrated in Algorithm 1), our algorithm for ranking terms and sentences based on their semantic relatedness. The first step of *SemanticRank* is the semantic graph creation. In the case of semantic keyword graphs, and given a document d_j which belongs in a document collection D, as a preprocessing step, the algorithm detects all $n-$grams of size up to 5 words using a dictionary look-up (i.e., both *WordNet* and *Wikipedia*), and a sliding window, in order to identify candidate keywords, which may be essentially composite terms. The resulting set of terms (i.e., can be terms of 1 to 5 words), which we denote as K_{d_j}, is used for the creation of a graph G with the vertices being all the distinct terms $t_i \in K_{d_j}$. As edge weights w_{ij} *SemanticRank* uses $SRT(t_i, t_j)$ which captures the semantic relatedness between terms t_i and t_j. However, ideally we would also like to incorporate in w_{ij} the statistical information of terms t_i, t_j that we have from

Algorithm 1 SemanticRank(D,*Mode*)

```
INPUT: A text document collection $D$, and a Mode flag
OUTPUT: A ranking $R$ of the semantic graph nodes for every document $d_j \in D$.
Execute(D,Mode)
if Mode is Keywords then
    Identify composite terms of length up to 5 words
end if
Compute and index TF-IDF values for all terms
for all $d_j \in D$ do
    $G$: An initially empty graph
    $G$ = Construct_Semantic_Graph($d_j$,Mode)
    $R$ = Rank_Nodes($G$)
end for
Construct_Semantic_Graph($d_j$,Mode)
$G$: an initially empty graph
if Mode is Keywords then
    Initialize $G$ with $K_{d_j}$
else
    Initialize $G$ with $Sen_{d_j}$
end if
for all pairs of vertices $(v_i, v_j)$ do
    if Mode is Keywords then
        $w_{i,j} = w_{j,i} = \lambda_{v_i,v_j} \cdot SRT(v_i, v_j)$
    else
        $w_{i,j} = w_{j,i} = SRS(v_i, v_j)$
    end if
end for
RETURN $G$
Rank_Nodes(G)
Execute Weighted PageRank in $G$
$R$ = Rank vertices of $G$ in descending order of PageRank values
RETURN $R$ with their PageRank values
```

their frequency of occurrence inside d_j and D. Thus, Equation 11 shows this combination, and it is the formula according to which *SemanticRank* computes the edge weights w_{ij} in the case of the semantic keyword graphs. In the case of semantic sentence graphs creation, *SemanticRank* initializes G with all the distinct sentences Sen_i in d_j as vertices, and it uses Equation 5 to compute the weights between every pair of vertices (i.e., between every pair of sentences). In Algorithm 1 we denote the set of distinct sentences in d_j with Sen_{d_j}.

$$w_{ij} = \lambda_{t_i,t_j} \cdot \mathit{SRT}(t_i, t_j) \tag{11}$$

In both cases, for the given document d_j, and after the creation of the semantic graph, nodes may be ranked according to the values produced by applying either Equation 6, or Equations 7 and 8. For the case of semantic keyword graphs, the top-k ranked nodes are selected as the most important keywords of d_j. For the case of semantic sentence graphs, the top-k ranked nodes are selected as the set of sentences, put together to constitute the automatically generated summary of d_j. In Algorithm 1 we may substitute line 26 with any of the ranking options discussed in Section 3.3. An analogy can be also drawn with PageRank's *random surfer model*, where a user browses the Web by following links from any given Web page. In the context of text modelling, *SemanticRank* implements what we refer to as *text surfing*, which relates to the concept of text cohesion (Halliday and Hasan, 1976), i.e., from a certain concept in a text, we are likely to *follow* ”links” to related concepts, meaning concepts that have lexical or semantic relation to the current concept.

5 Experimental Evaluation

The experimental evaluation is performed in two tasks: (a) keyword extraction, and (b) text summarization. In both cases we create a semantic graph for each document and we rank the nodes accordingly, using Algorithm 1. For our evaluation we use all the ranking algorithm alternatives described in Section 3.3, and compare results with state of the art approaches that use the same ranking algorithms but different graph creation and edge weighting approaches. The various tested ranking alternatives are: weighted *SemanticRank* (*Sem*) using *PageRank* (*WPR*), and *HITS* (*WHITS*), and unweighted *SemanticRank* (*USem*) using the original versions of *PageRank* (*UPR*) and *HITS* (*UHITS*). In the case of keyword extraction we evaluate additionally the *Averaged PageRank Weighting* (*APW*) and *PageRank Priors* (*PPR*), where the *prior* nodes were set to the terms occurring in each abstract's title.

Method		P	R	F
Sem (k=5)	WPR	0.396	0.121	0.1853
	WHITS	0.348	0.088	0.14
	APW	0.556	0.185	0.278
	P-PR	**0.659**	0.226	0.337
Sem (k=10)	WPR	0.368	0.2463	0.296
	WHITS	0.335	0.138	0.195
	APW	0.498	0.331	0.398
	P-PR	0.524	0.352	0.422
Sem (k=15)	WPR	0.371	0.364	0.368
	WHITS	0.355	0.241	0.287
	APW	0.449	0.442	0.446
	P-PR	0.451	0.441	0.446
Sem (k=20)	WPR	0.376	0.466	0.417
	WHITS	0.374	0.312	0.34
	APW	0.421	**0.532**	**0.47**
	P-PR	0.418	0.514	0.46
USem (k=5)	UPR	0.057	0.046	0.048
	UHITS	0.061	0.053	0.055
USem (k=10)	UPR	0.06	0.102	0.07
	UHITS	0.06	0.108	0.072
USem (k=15)	UPR	0.052	0.116	0.069
	UHITS	0.054	0.123	0.072
USem (k=20)	UPR	0.052	0.14	0.074
	UHITS	0.053	0.151	0.076
Michalcea (2004)		0.312	0.431	0.362
Hulth (2003)		0.252	0.517	0.339

Table 1: Results of the keyword extraction task in the Inspec database.

5.1 Keyword Extraction

We applied *SemanticRank* in an automated keyword extraction task on the Inspec database[4]. The Inspec database stores abstracts of journal papers from computer science and information technology and the keyword extraction task aims in selecting the most descriptive keywords for each abstract. Each abstract has been already assigned keywords by professional indexers, which constitute the gold standards for systems' comparison. The mean number of assigned terms per abstract from the experts is 7.63. The goal is to extract as many of the keywords suggested by the professional indexers as possible for each abstract. In this data set our results are directly comparable to the works in (Mihalcea and Tarau, 2004) and (Hulth, 2003).

We evaluate *SemanticRank* (*Sem*) using varying k values $(5, 10, 15$, and $20)$, where k stands for the number of keywords to be extracted from each abstract. In Table 1 we report the results of macro-averaged pre-

[4] Many thanks to Anette Hulth for providing us the data set used in her keyword extraction experiments.

System		F-Measure
Sem	WPR	$0.40996(0.39067 - 0.4292)$
	WHITS	$0.3651(0.3435 - 0.38609)$
USem	UPR	$0.2951(0.2727 - 0.3195)$
	UHITS	$0.3132(0.2901 - 0.3375)$
T		$0.4131(0.3922 - 0.434)$
P		$0.4039(0.3843 - 0.4226)$
O		$0.3905(0.3663 - 0.4132)$
V		$0.3885(0.368 - 0.4085)$
Q		$0.3857(0.3616 - 0.4089)$
Baseline		$0.3549(0.3329 - 0.3756)$

Table 2: Results (F-Measure) of the single-document summarization task, (DUC 2001).

System		F-Measure
Sem	WPR	$0.4971(0.4799 - 0.5164)$
	WHITS	$0.3836(0.3815 - 0.4047)$
USem	UPR	0.3086(0.297-0.32084)
	UHITS	0.2851(0.2735-0.297)
TextRank		0.4904
S27		0.5011
S31		0.4914
S28		0.489
S21		0.4869
S29		0.4681
Baseline		0.4779

Table 3: Results (F-Measure) of the single-document summarization task, (DUC 2002).

cision (P), recall (R), and F-Measure (F) over all abstracts. Precision for each abstract is the number of correctly extracted keywords, divided by the number of extracted keywords, and recall differs only in the denominator (number of keywords suggested by the indexers). We also present the best reported results for the algorithms in (Mihalcea and Tarau, 2004), and (Hulth, 2003).

Results show that *SemanticRank* with weighted *PageRank* gives better F-Measure from the approaches in (Mihalcea and Tarau, 2004) and (Hulth, 2003) for $k = 15$ and $k = 20$ and always better from weighted *HITS*. *APW* and *P-PR* have higher F-Measure than *WPR*, achieving top performance (bold values), with *APW* producing the best F-Measure for $k = 20$. In this case, the difference between *APW* and *TextRank*, both in precision and recall, was found statistically significant at the 0.95 confidence level, using Fisher's exact test. In addition, we can observe that the unweighted versions of *PageRank* and *HITS* produce very poor results. This shows that our method benefits greatly from the suggested edges' weighing scheme.

5.2 Text Summarization

We evaluated *SemanticRank* in two different text summarization tasks: single-document, and multi-document summarization. As in the keyword extraction task, we evaluate both the weighted and the unweighted versions of *SemanticRank* (*Sem* and *USem*) using *WPR*, *WHITS*, *UPR*, and *UHITS* respectively. We also compare against state of the art results in the used data sets, and we report on results from related methods (i.e., *TextRank*) when possible.

5.2.1 Single Document Summarization

In the single-document summarization task we have used the data sets of the *Document Understanding Conference* (*DUC*) from the 2001 and 2002 competitions. The two data sets comprise 308 and 567 news articles respectively. For both data sets, two reference summaries per document were provided. The task for the participating systems in both competitions was to provide for each document a summary of at most 100 words. Thus, we apply *SemanticRank* by first ranking sentences following Algorithm 1, and then by merging them, starting from the top ranked sentences, until the 100 words limit is reached. For the evaluation against the reference summaries, we are using the *ROUGE* toolkit, which is based on $N-$grams, and has been the standard evaluation methodology for the summarization task (Lin and Hovy, 2003) in all the recent DUC competitions. Since in *DUC* 2001 and *DUC* 2002 the *ROUGE* system was not the standard evaluation toolkit, we implemented the evaluation of the two tasks in *ROUGE*. The setup we adopted for ROUGE was (*Ngram(1,1)*, stemmed words and no stopwords), identical to the one adopted in (Mihalcea and Tarau, 2004).

In Table 2 we present the F-Measure values produced from *ROUGE* for *SemanticRank*, and the top 5 performing systems (participating systems *T*, *P*, *O*, *V*, and *Q*), for the 2001 data set. Similarly, Table 3 presents the results for the 2002 data set. In both cases we report the performance of a simple baseline method, that takes the first sentences from each article, until the limit of 100 words is reached. When available, we also present the results from (Mihalcea and Tarau, 2004), and also the 0.95 confidence intervals for the F-Measure values, as these were generated by *ROUGE*. The results in the two tables show that *SemanticRank*, when the weighted version of PageRank is used, produces very high F-Measure score. In both cases, our system ranks among the top 2 systems in the task.

System		F (R-2)	F (R-SU4)
Sem	WPR	0.093	0.133
	WHITS	0.078	0.115
USem	UPR	0.031	0.069
	UHITS	0.028	0.062
S40		0.111	0.143
S55		0.098	0.135
S45		0.096	0.132
S44		0.093	0.136
S47		0.093	0.130
Baseline		0.085	0.122

Table 4: Results of the multi-document summarization task (DUC 2007 update task).

5.2.2 Multi Document Summarization

For the multi document summarization task we used the data from the *DUC 2007 update task*. The data set consists of 250 documents organized in topics, and each topic is further divided into three clusters, for each of which gold standard summaries are provided by evaluators. In this case, the average of *ROUGE-2* and *ROUGE-SU4* scores are used for evaluation. Table 4 presents the average F-Measure values for both scores. We also report the top-5 performing systems in the respective task, as well as the performance of the generic baseline that was used in this case. As Table 4 shows, the combination of *SemanticRank* with the weighted *PageRank* produces better results than weighted *HITS* and the unweighted versions. This drop in performance compared to the results in the single-document summarization task can be partly explained by the fact that in this case the *DUC 2007 update task* allows for the system to assume previous knowledge for the document clusters B and C of each topic. In our case, we have not embedded any methodology that takes advantage of this knowledge.

Regarding the top system in Table 4, system *S40*, is the system called *GISTEXTER* (Hickl et al., 2007). *GISTEXTER* uses textual inference and textual contradiction to construct representations of knowledge encoded in a document collection. The system comprises four components: question processing, sentence retrieval, sentence ranking, and summary generation. However, for the summary generation component, a set of heuristics is used to generate the summary. In a similar approach (Amini and Usunier, 2007), where coherent text fragments are sought with regards to the initial question, the authors show that query expansion using a contextual approach may lead to find important terms for the summary, among different related documents. Their system ranked among the top in the main task of *DUC 2007*, leading to the conclusion that for a multi-document summarization system, a contextual approach might be more efficient than *SemanticRank*.

However, from the results presented in Tables 2, 3 and 4, we experienced a very good performance of *SemanticRank* in ranking sentences for the text summarization task, with the weighted ranking variations producing always better results than the unweighted.

6 Conclusions and Future Work

In this paper we introduced *SemanticRank*, a new algorithm for ranking keywords and text segments using measures of semantic relatedness. The novelty of the algorithm is its semantic graph creation step, which is based on a measure of semantic relatedness that combines *WordNet* and *Wikipedia*. We evaluated *SemanticRank* using several alternatives for its ranking step, all based on weighted and unweighted variations of *PageRank* and *HITS*. Results in keyword extraction and text summarization experiments show that it performs favorably over state of the art related methods, and that the selected edges' weighting boosts its performance. In our future work we will examine the potentiality of more graph-based ranking methods, and it is on our next plans to embed *SemanticRank* on more linguistic tasks, such as sentiment analysis and opinion mining.

References

Agirre, Eneko and Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. In *Proc. of the 12th Conference of the European Chapter of the ACL*, pages 33–41.

Aizawa, Akiko N. 2003. An information-theoretic perspective of tf-idf measures. *Information Processing Management*, 39(1):45–65.

Amini, Massih and Nicolas Usunier. 2007. A contextual query expansion approach by term clustering for robust text summarization. In *Proc. of the DUC 2007 Conference*.

Baldwin, Breck and Thomas S. Morton. 1998. Dynamic coreference-based summarization. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*.

Boguarev, Branimir and Christopher Kennedy. 1997. Salience-based content characterization of text documents. In *Proc. of ACL/EACL Workshop on Intelligent Scalable Text Summarization*.

Budanitsky, Alexander and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

Gabrilovich, Evgeniy and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proc. of the 20th International Joint Conference on Artificial Intelligence*, pages 1606–1611.

Halliday, Michael A.K. and Ruqaiya Hasan. 1976. *Cohesion in English*. Longman.

Haveliwala, Taher H. 2002. Topic-sensitive pagerank. In *Proc. of the World Wide Web Conference*, pages 517–526.

Hickl, Andrew, Kirk Roberts, and Finley Lacatusu. 2007. Lcc's gistexter at duc 2007: Machine reading for update summarization. In *Proc. of the DUC 2007 Conference*.

Huang, Chong, Yong Hong Tian, Zhi Zhou, Charles X. Ling, and Tiejun Huang. 2006. Keyphrase extraction using semantic networks structure analysis. In *Proc. of the 6th International Conference on Data Mining*, pages 275–284.

Hulth, Anette. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 216–223.

Lin, Chin-Yew and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proc. of the Human Language Technology Conference and the North American Chapter of the ACL Conference*.

Litvak, Marina and Mark Last. 2008. Graph-based keyword extraction for single-document summarization. In *Proc. of the ACL Workshop on Multi-source Multilingual Information Extraction and Summarization*, pages 17–24.

Mani, Interjeet and Eric Bloedorn. 1998. Machine learning of generic and user-focused summarization. In *Proc. of the 15th National Conference on A.I. and 10th Innovative Applications of A.I. Conference*, pages 821–826.

Mihalcea, Rada and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 404–411.

Mihalcea, Rada. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proc. of the 42nd Annual Meeting of the ACL*.

Milne, David N. and Ian H. Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proc. of the 1st AAAI Workshop on Wikipedia and Artificial Intelligence*.

Salton, Gerard, Amit Singhal, Mandar Mitra, and Chris Buckley. 1997. Automatic text structuring and summarization. *Information Processing Management*, 33(2):193–207.

Steinberger, Josef and Karel Jezek. 2009. Text summarization: An old challenge and new approaches. In *Foundations of Computational Intelligence (6)*, pages 127–149.

Tsatsaronis, George, Iraklis Varlamis, and Michalis Vazirgiannis. 2010. Text relatedness based on a word thesaurus. *Journal of Artificial Intelligence Research*, 37:1–39.

Wang, Jinghua, Jianyi Liu, and Cong Wang. 2007. Keyword extraction based on pagerank. In *Proc. of the 11th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 857–864.

White, Scott and Padhraic Smyth. 2003. Algorithms for estimating relative importance in networks. In *Proc. of the 9th International Conference on Knowledge Discovery and Data Mining*, pages 266–275.

Yeh, Jen-Yuan, Hao-Ren Ke, and Wei-Pang Yang. 2008. iSpreadRank: Ranking sentences for extraction-based summarization using feature weight propagation in the sentence similarity network. *Expert Syst. Appl.*, 35(3):1451–1462.

Chinese CCGbank:
extracting CCG derivations from the Penn Chinese Treebank

Daniel Tse and **James R. Curran**
School of Information Technologies
University of Sydney
{dtse6695,james}@it.usyd.edu.au

Abstract

Automated conversion has allowed the development of wide-coverage corpora for a variety of grammar formalisms without the expense of manual annotation. Analysing new languages also tests formalisms, exposing their strengths and weaknesses.

We present Chinese CCGbank, a 760,000 word corpus annotated with Combinatory Categorial Grammar (CCG) derivations, induced automatically from the Penn Chinese Treebank (PCTB). We design parsimonious CCG analyses for a range of Chinese syntactic constructions, and transform the PCTB trees to produce them. Our process yields a corpus of 27,759 derivations, covering 98.1% of the PCTB.

1 Introduction

An annotated corpus is typically used to develop statistical parsers for a given formalism and language. An alternative to the enormous cost of hand-annotating a corpus for a specific formalism is to convert from an existing corpus.

The Penn Treebank (PTB; Marcus et al., 1994) has been converted to HPSG (Miyao et al., 2004), LFG (Cahill et al., 2002), LTAG (Xia, 1999), and CCG (Hockenmaier, 2003). Dependency corpora, e.g. the German Tiger corpus, have also been converted (Hockenmaier, 2006). The Penn Chinese Treebank (PCTB; Xue et al., 2005) provides analyses for 770,000 words of Chinese. Existing PCTB conversions have targeted TAG (Chen et al., 2005) and LFG (Burke and Lam, 2004; Guo et al., 2007).

We present Chinese CCGbank, a Chinese corpus of CCG derivations automatically induced from the PCTB. Combinatory Categorial Grammar (CCG; Steedman, 2000) is a lexicalised grammar formalism offering a unified account of local and non-local dependencies. We harness the facilities of CCG to provide analyses of Chinese syntax including topicalisation, pro-drop, zero copula, extraction, and the 把 *ba-* and 被 *bei*-constructions.

Pushing the boundaries of formalisms by subjecting them to unfamiliar syntax also tests their universality claims. The freer word order of Turkish (Hoffman, 1996) and the complex morphology of Korean (Cha et al., 2002) led to the development of extensions to the CCG formalism.

We present our analysis of Chinese syntax under CCG, and provide an algorithm, modelled after Hockenmaier and Steedman (2007), to incrementally transform PCTB trees into CCG derivations. The algorithm assigns CCG categories which directly encode head and subcategorisation information. Instances of Chinese syntax demanding special analysis, such as extraction, pro-drop or topicalisation, are pin-pointed and given elegant analyses which exploit the expressivity of CCG.

Our conversion yields CCG analyses for 27,759 PCTB trees (98.1%). Coverage on lexical items, evaluated by 10-fold cross-validation, is 94.46% (by token) and 73.38% (by type).

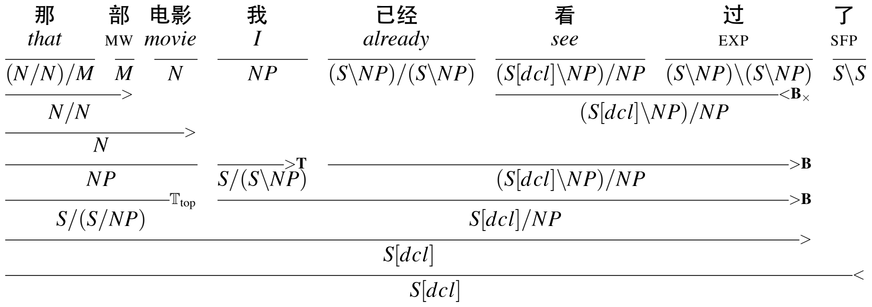
We present the first CCG analysis of Chinese syntax and obtain a wide-coverage CCG corpus of Chinese. Highly efficient statistical parsing using a CCGbank has recently been demonstrated for English (Clark and Curran, 2007). Our Chinese CCGbank will enable the development of similarly efficient wide-coverage CCG parsers for Chinese.

2 Combinatory Categorial Grammar

CCG (Steedman, 2000) is a lexicalised grammar formalism, with a transparent syntax-semantics interface, a flexible view of constituency enabling concise accounts of various phenomena, and a consistent account of local/non-local dependencies.

It consists of *categories*, which encode the type and number of arguments taken by lexical items, and *combinators*, which govern the possible interactions between categories.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1083–1091,
Beijing, August 2010



那 部 电影 我 已经 看 过 了
that MW movie I already see EXP SFP
(N/N)/M M N NP (S\NP)/(S\NP) (S[dcl]\NP)/NP (S\NP)\(S\NP) S\S
N/N
(S[dcl]\NP)/NP
N
NP S/(S\NP) (S[dcl]\NP)/NP
S/(S/NP) S[dcl]/NP
S[dcl]
S[dcl]

Figure 1: Chinese CCG derivation: "That movie, I've already seen."

A CCG grammar defines *atomic categories*, e.g. *NP* and *S*, which may be recursively constructed into *complex categories*, e.g. N/N and $S\backslash NP$.[1] Figure 1 shows how combinators govern the interaction of categories for lexical items, while slashes specify argument directionality.

The combinators allow us to reduce lexical ambiguity, by preserving a word's canonical category even when displaced from its canonical position. This facility is a strength of CCG, but elevates its generative power to mild context-sensitivity.

Some combinators may be disabled in a given language – the *multi-modal* CCG (Baldridge, 2002) allows these distinctions to be lexically specified.

Introducing non-CCG rules decrease categorial ambiguity at the expense of deviating from the formalism. Hockenmaier and Steedman (2002) show that these greatly improve lexical coverage. Their analysis of English employs non-CCG rules to coerce a verb phrase headed by a participle (category $S[ng]\backslash NP$) to a post-nominal modifier:

$$S[ng]\backslash NP \longrightarrow NP\backslash NP \qquad (1)$$

This frees verbs from having to possess a distinct category in each position, thus trading off lexical ambiguity for derivational ambiguity. Honnibal and Curran (2009) extended CCG with *hat categories*, enabling the lexical specification of these unary type-change rules.

Hockenmaier and Steedman (2002, 2007) developed CCGbank, the first wide-coverage English CCG corpus, by converting 1.2 million words from the Wall Street Journal section of the PTB. CCGbank has made possible the development of wide-coverage statistical parsers for CCG in English, notably C&C (Clark and Curran, 2007).

3 Penn Chinese Treebank

Xue et al. (2005) developed the Penn Chinese Treebank (PCTB), the first syntactically annotated corpus for Chinese. The corpus includes newswire text, magazine articles, and transcribed speech.[2]

Xue et al. establishes several principles for a more disciplined and consistent style of annotation compared to the original PTB. These principles include *complement/adjunct marking:* allowing the recovery of predicate-argument structure; *limited semantic role marking:* the annotation of modifier phrases with semantic roles; *covert argument marking:* the retention of traces of arguments deleted through pro-drop; and *NP internal structure:* bracketing of NP structure where the intended interpretation is clear.

The *one relation per bracketing* principle unambiguously encodes a grammatical relation (chiefly, predication, adjunction, or complementation) through the configuration of a node and its children. Xue et al. developed this principle to assist conversions from the PTB, e.g. Hockenmaier (2003), in resolving argument/adjunct distinctions.

PCTB derivations are pre-segmented, pre-tokenised, and POS tagged. Owing to the dearth of morphology in Chinese, the concept of *part of speech* is more fluid than that of English – the word 比较 *bijiao* 'compare' might be glossed as a verb, adjective, adverb, or noun depending on its context. Noun/verb mis-taggings are a frequent error case for PCFG parsing on PCTB data, compounded in Chinese by the lack of function words and morphology (Levy and Manning, 2003). This ambiguity is better handled by the adaptive multitagging approach used by Clark and Curran (2007) for CCG supertagging, in which each lexical item is tagged with a set of CCG categories.

We present our CCG analysis of Chinese syntax below, followed by our conversion algorithm.

[1] *Abbreviations in this paper:* The directionless slash | stands for one of $\{/, \backslash\}$. We also use the verbal category abbreviations $VP \equiv S\backslash NP$ and $TV \equiv (S\backslash NP)/NP$.

[2] We use the Penn Chinese Treebank 6.0 (LDC2007T36).

4 The syntax of Chinese

4.1 Basic clause structure

Chinese is typologically SVO, with some OV elements (relative clauses, adjunct PPs and noun modifiers precede their heads). Numbers and determiners may not modify nouns directly; a *measure word* must intervene.

The category structure of the grammar may be inferred directly from headedness information. Heads subcategorise for the type, number and directionality of their arguments, while adjuncts receive modifier categories of the form $X \mid X$.

(2)	我	在	超市	买
	I	at	supermarket	buy
	NP	$(VP/VP)/NP$	NP	VP/NP
	了	一	盒	鸡蛋
	PERF	one	box:MW	eggs
	$VP\backslash VP$	$(N/N)/M$	M	N

I bought a box of eggs at the supermarket.

4.2 Topicalisation

In topic-prominent languages, the *topic* refers to information which the speaker assumes is known by the listener. In Mandarin, topicalisation manifests as left-dislocation of the topic phrase (Li and Thompson, 1989). We distinguish *gap* and *non-gap* topicalisation depending on whether the topic is co-referent with a gap in the sentence.[3]

For gapped topicalisation (cf. Figure 1), we adopt the Steedman (1987) topicalisation analysis:

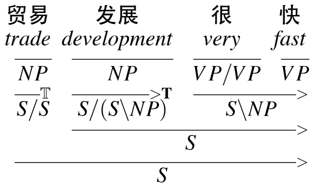
$$T \rightarrow S/(S/T) \text{ for parametrically licensed } T \quad (3)$$

For non-gap topicalisation (Example 5), we use a variation of the analysis described in Hockenmaier and Steedman (2005), which treats the topicalised constituent as a sentential modifier. Under this analysis, the determiner in a topicalised NP receives $(S/S)/N$ instead of its canonical category NP/N. Instead, we propose a unary rule:

$$T \rightarrow S/S \text{ for topicalisation candidate } T \quad (4)$$

This delays the coercion to sentential modifier type (i.e. $NP \rightarrow S/S$) until after the NP has been consolidated, allowing the words under the topicalised NP to preserve their canonical categories.

(5) (As for) trade, it has developed rapidly.



Topicalisation is far less marked in Chinese than in English, and the structure of topicalised constituents is potentially quite complex. The additional categorial ambiguity in Hockenmaier and Steedman (2005) compounds the data sparsity problem, leading us to prefer the unary rule.

4.3 Pro-drop

Since Chinese exhibits *radical pro-drop* (Neeleman and Szendrői, 2007), in which the viability of the pro-drop is not conditioned on the verb, the categorial ambiguity resulting from providing an additional argument-dropped category for every verb is prohibitive.

Rather than engendering sparsity on verbal categories, we prefer derivational ambiguity by choosing the unary rule analysis $S[dcl] \mid NP \rightarrow S[dcl]$ to capture Chinese pro-drop.

4.4 Zero copula

Although the Chinese copula 是 *shi* is obligatory when equating NPs, it may be omitted when equating an NP and a QP or PP (Tiee and Lance, 1986).[4]

(6)	她	今年	十八	岁
	NP	VP/VP	$(S\backslash NP)/M$	M
	3SG	this-year	18	years-old

She is 18 this year.

A solution involving a binary rule $NP\ QP \rightarrow S[dcl]$ is not properly headed, and thus violates the Principle of Lexical Head Government (Steedman, 2000). Conversely, a solution where, for example, 十八 '18' would have to receive the category $(S[dcl]\backslash NP)/M$ instead of its canonical category QP/M would lead to both data sparsity and over-generation, with VP modifiers becoming able to modify the QP directly. Tentatively, we ignore the data sparsity consequences, and have 十八 '18' receive the category $(S[dcl]\backslash NP)/M$ in this context.

[3] Non-gap topicalisation is also known as the *double subject construction* (Li and Thompson, 1989).

[4] The copula is ungrammatical in predication on an adjectival verb, such as 高兴 'happy'. However, we analyse such words as verbs proper, with category $S[dcl]\backslash NP$.

4.5 把 *ba*- and 被 *bei*-constructions

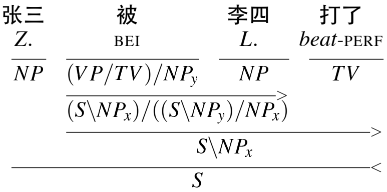
被 *bei* and 把 *ba* introduce a family of passive-like constructions in Chinese. Although superficially similar, the resulting constructions exhibit distinct syntax, as our CCG analysis reflects and clarifies.

In the 被 *bei*-construction, the patient argument of a verb moves to subject position, while the agent either becomes the complement of a particle 被 *bei* (the *long passive*), or disappears (the *short passive*; Yip and Rimmington, 1997). Although the two constructions are superficially similar (apparently differing only by the deletion of the agent NP), they behave differently in more complex contexts (Huang et al., 2008).

The long passive occurs with or without an object gap (deleted by identity with the subject of the matrix verb). We analyse this construction by assigning 被 *bei* a category which permutes the surface positions of the agent and patient. Co-indexation of heads allows us to express long-distance dependencies.

Bei receives $((S\backslash NP_y)/((S\backslash NP_x)/NP_y))/NP_x$ in the gapped case (cf. Example 7) and $((S\backslash NP)/(S\backslash NP_x))/NP_x$ in the non-gapped case.

(7) Zhangsan was beaten by Lisi.



Short passives also occur with or without an object gap, receiving $(S\backslash NP_x)/((S\backslash NP)/NP_x)$ in the gapped case and $(S\backslash NP)\backslash(S\backslash NP)$ in the non-gapped case. Our analysis agrees with Huang et al. (2008)'s observation that short-*bei* is isomorphic to English *tough*-movement: our short-*bei* category is the same as Hockenmaier and Steedman (2005)'s category for English *tough*-adjectives.

In the 把 *ba* construction, a direct object becomes the complement of the morpheme 把 *ba*, and gains semantics related to "being affected, dealt with, or disposed of" (Huang et al., 2008). As for 被 *bei*, we distinguish two variants depending on whether the object is deleted under coreference with the complement of 把 *ba*.

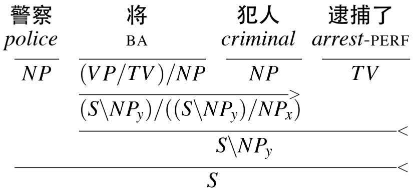
Ba receives $((S\backslash NP_y)/((S\backslash NP_y)/NP_x))/NP_x$ in the gapped case (cf. Example 8), and $((S\backslash NP_y)/(S\backslash NP_y))/NP$ in the non-gapped case.

As Levy and Manning (2003) suggest, we reshape the PCTB analysis of the *ba*-construction so that *ba* subcategorises for its NP and VP, rather than subcategorising for an IP sibling, which allows the NP to undergo extraction.

Tag	*Headedness*	*Example*
VSB	head-final	规划 建设 'plan [then] build'
VRD	right-adjunction	煮 熟 'cook done'
VCP	head-initial	确认 为 'confirm as'
VCD	appositive	投资 设厂 'invest [&] build-factory'
VNV	special	去 不 去 'go [or] not go'
VPT	special	离 得 开 'leave able away'

Table 1: Verb compounds in PCTB

(8) The criminals were arrested by the police.



4.6 Verbal compounding

Verbs resulting from compounding strategies are tagged and internally bracketed. Table 1 lists the types distinguished by the PCTB, and the headedness we assign to compounds of each type.

Modifier-head compounds (PCTB tag VSB) exhibit clear head-final semantics, with the first verb V_1 causally or temporally preceding V_2. Verb coordination compounds (VCD) project multiple heads, like ordinary lexical coordination.

In a resultative compound (VRD), the result or direction of V_1 is indicated by V_2, which we treat as a post-verbal modifier. The *V-not-V* construction (VNV) forms a yes/no question where $V_1 = V_2$. In the *V-bu/de-V* or potential verb construction (VPT), a disyllabic verb $V = V_1V_2$ receives the infix 得 *de* or 不 *bu* with the meaning *can/cannot V*. In both these cases, it is the infixed particle 得 *de* or 不 *bu* which collects its arguments on either side.

4.7 Extraction

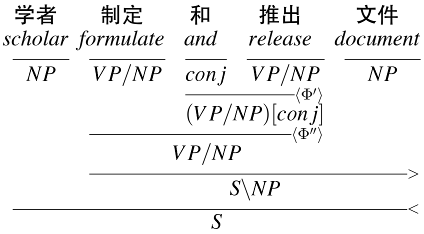
In the Chinese relative clause construction, the particle 的 *de* links a sentence with a subject or object gap with a NP to which that gap co-refers, in an analysis similar to the English construction described by Hockenmaier and Steedman (2005), mediated by the relative pronoun *that*.

As in the English object extraction case, forward type-raising on the subject argument, and forward composition into the verbal category allows us to obtain the correct object gap category S/NP.

4.8 Right node raising

Two coordinated verbs may share one or more contiguous arguments under right node raising. This analysis follows directly from the CCG definition of coordination, requiring no new lexical categories.

(9) Scholars have formulated and are releasing the documents.



4.9 Apposition

Apposition is the juxtaposition of two phrases referring to the same entity. Unlike noun modification, no clear modification relationship holds between the two phrases. The direct juxtaposition rules out Hockenmaier's (2003) analysis where a delimiting comma mediates the apposition. Chinese also allows full sentence/NP apposition:

(10) (用户 浪费 水)$_S$ 事件$_{NP}$
(users waste water)$_S$ incident$_{NP}$
incidents of users wasting water

This gives rise to the Chinese apposition binary rules $NP\ NP \rightarrow NP$ and $S[dcl]\ NP \rightarrow NP$.

5 The translation pipeline

5.1 Tagging

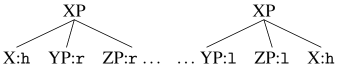
Each PCTB internal node structurally encodes a *configuration*, which lets us distinguish head-initial and head-final complementation from adjunction and predication (Xue et al., 2000).

The tagging mechanism annotates the PCTB tag of each internal node with a *marker*, which preserves this headedness information, even after the nodes are re-structured in the binarisation phase.

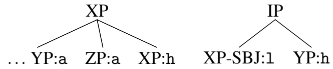
Hockenmaier's (2003) conversion algorithm uses the Magerman (1994) head-finding heuristics, a potential source of noise. Fortunately, the PCTB encodes gold standard headedness data.

The tagging algorithm is straightforward: if a node and its children unify with one of the schemata below, then the markers (e.g. `:l` or `:n`) are attached to its children. The markers `l` and `r` indicate complements *left*, or *right* of the *head* `h`; adjuncts are marked with `a`.

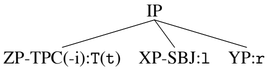
Head-initial, -final complementation



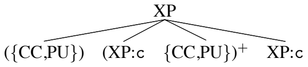
Adjunction, predication



Topicalisation (gap and non-gap)



Coordination



Others identify nodes with special syntax, such as topicalisation (`t/T`), apposition (`A`) or coordination (`c`), for special treatment in following phases.

NP internal structure

To speed annotation, NP internal structure is often left underspecified in PCTB (Xue et al., 2005), as in the Penn Treebank. As a result, 68% of non-trace `NP`s in PCTB have only a flat bracketing.

We assume that the internal structure of flat NPs is right-branching and head-final (Li and Thompson, 1989), following Hockenmaier and Steedman (2005), who assume this structure for English. A re-analysis of PCTB, like Vadas and Curran (2007) for the PTB, could restore this structure, and allow our conversion algorithm to yield the correct CCG analysis with no further modifications.

To obtain this default analysis, each node under NP internal structure receives the marker `n`, except the the final node, the head, which receives `N`.

5.2 Binarisation

CCG combinators take at most two categories, inducing binary derivation trees. As such, PCTB trees must be re-shaped to accommodate a CCG analysis.

Our markers control the shape of the binarised structure: head-initial complementation yields a left-branching tree, while head-final complementation, adjunction, predication, coordination, and NP internal structure all yield right-branching trees. Following Hockenmaier (2003), sentence-final punctuation is attached high.

Although the distinction between word-level tags (such as `NN`, `VA`) and phrasal tags (such as `NP`, `VP`, `LCP`) enables the configurational encoding of grammatical relations, it leaves a large number of

VP	←	VV,VE,VA,VRD	ADJP	←	JJ
ADVP	←	AD, CS	CLP	←	M
LCP	←	LC	DP	←	DT, OD
LST	←	OD	INTJ	←	IJ
FLR	←	*any node*	PP	←	P

Figure 2: Pruned unary projections

LCP	Localiser phrase	PP	Prepositional phrase
M	Measure word	QP	Quantifier phrase
N	Bare noun	S	Sentence
NP	Noun phrase	conj	Conjunction word

Table 2: Chinese CCGbank atomic category set

unary projections. While an intransitive verb (e.g. 睡觉 'sleep') would carry the verbal PCTB tag VV, and a transitive verb combined with its object (e.g. 吃了晚饭 'ate dinner') is annotated as VP, under CCG's freer concept of constituency, both receive the category *S\NP*.

Pruning the unary projections in Fig. 2 prevents spurious category labellings in the next phase.

5.3 Labelling

We label each node of the binarised tree with CCG categories, respecting the headedness information encoded in the markers.

Atomic categories

The chosen mapping from PCTB tags to categories defines the *atomic category set* for the grammar. The richer representation in CCG categories permits some constituents to be expressed using a smaller set of atoms (e.g. an adjective is simply a noun modifier – *N/N*). Despite their critical importance in controlling the degree of under-/over-generation in the corpus, little guidance exists as to the selection of atomic categories in a CCG grammar. We observed the following principles:

Modifier proliferation: when two classes of words can be modified by the same class of modifiers, they should receive a single category;

Over-generation: the atom set should not overgeneralise to accept ungrammatical examples;

Efficiency: the representation may be motivated by the needs of applications such as parsers.

Table 2 shows the eight atomic categories chosen for our corpus. Two of these categories: *LCP* (localisers) and *M* (measure words) have variously been argued to be special sub-classes of nouns (Huang et al., 2008). However, based on our overgeneration criterion, we decided to represent these as atomic categories.

We adopt the bare/non-bare noun distinction from Hockenmaier and Steedman (2007) on parsing efficiency grounds. Although they roughly correspond to English *PP*s, the distributional differences between *PP*s, *LCP*s and *QP*s justify their inclusion as atoms in Chinese. Future work in training a wide-coverage parser on Chinese CCGbank will evaluate the impact of these choices.

Labelling algorithm

We developed a recursive algorithm which applies one of several labelling functions based on the markers on a node and its children.

The algorithm proceeds top-down and assigns a CCG category to every node. The markers on a node's children are matched against the schema of Table 3, applying the categories of the matching schema to the children. The algorithm is then called recursively on each child. If the algorithm is called on an unlabelled node, the mapping from PCTB tags is used to assign a CCG category.

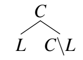
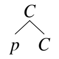
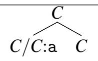
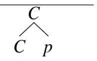
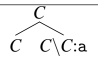
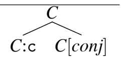
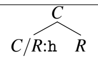
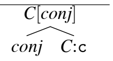
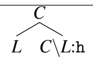
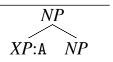
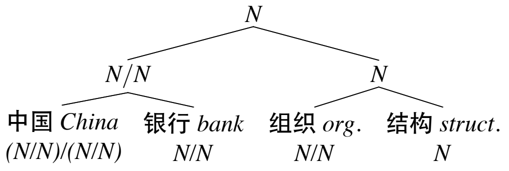
Predication		Left absorption	
Adjunction		Right absorption	
Right adjunction		Coordination	
Head-initial		Partial coordination	
Head-final		Apposition	

Table 3: Category labelling schemata

Left- and right-absorption are non-CCG rules which functionally ignore punctuation, assuming that they project no dependencies and combine to yield the same category as their non-punctuation sibling (Hockenmaier and Steedman, 2007). In the schema, *p* represents a PCTB punctuation POS tag.

NPs receive a head-final bracketing (by our right-branching assumption), respecting NP internal structure where provided by PCTB:



6 Post-processing

A number of cases remain which are either not covered by the general translation algorithm, or otherwise could be improved in a post-processing step. The primary disharmony at this stage is the presence of *traces*, the empty categories which the PCTB annotation style uses to mark the canonical position of extraposed or deleted constituents. 19,781 PCTB derivations (69.9%) contain a trace. Since CCG aims to provide a transparent interface between surface string syntax and semantics, traces are expressly disallowed (Steedman, 2000). Hence, we eliminate traces from the annotation, by devising alternate analyses in terms of categories and combinatory rules.

Subject/object extraction

8966 PCTB derivations (31.7%) contain a subject extraction, while 3237 (11.4%) contain an object extraction. Figure 3 shows the canonical representation of subject extraction in the PCTB annotation style. The PCTB annotation follows the X' analysis of the relative clause construction as described by Wu (2004), which we transform into an equivalent, trace-free CCG analysis.

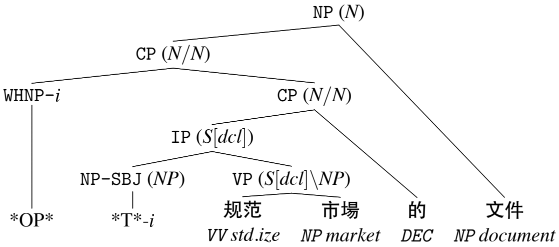


Figure 3: ‘*the document which standardises the market*’

First, the *Spec* trace, WHNP-*i*, coindexed with the extracted argument(s), is deleted. Next, the extracted argument(s) with matching indices are deleted, and category structure is adjusted to generate the correct gap category.

Modifier categories

Under our analysis, aspect particles such as 了 *le* (perfective) and 过 *guo* (experiential) are verbal post-modifiers, corresponding to *right adjunction* in Table 3. Accordingly, an aspect particle following a transitive verb VP/NP will receive the modifier category $(VP/NP)\backslash(VP/NP)$. Under this analysis, every verbal category gives rise to one possible modifier category for each aspect particle, leading to detrimental categorial ambiguity.

However, the generalised backward crossed composition combinator (Steedman, 2000) lets aspect particles retain their canonical category $(S\backslash NP)\backslash(S\backslash NP)$ regardless of the arity of the verb they modify.

Transformations

The PCTB annotation style posits traces to account for gapping, control/raising, argument sharing, pro-drop and topicalisation. To effect the parsimonious CCG analyses of Section 4, structural transformations on the original PCTB trees are necessary to accommodate the new analyses.

We developed a `tgrep`-like language which identifies instances of Chinese constructions, such as right node raising and pro-drop, whose PCTB annotation posits traces. The local trees are then reshaped to accommodate trace-free CCG analyses.

7 Evaluation

This section explores the coverage characteristics of Chinese CCGbank, in comparison with the English and German CCGbanks generated by Hockenmaier. Our analysis follows Hockenmaier (2006) in establishing *coverage* as the metric reflecting how well the target corpus has accounted for constructions in the source corpus.

7.1 Corpus coverage

The Chinese CCGbank conversion algorithm completes for 28,227 of the 28,295 (99.76%) PCTB trees. Annotation noise, and rare but legitimate syntax, such as ellipsis, account for the coverage lost in this phase. Following Hockenmaier and Steedman (2005), we adjust the PCTB annotation only for systematic tagging errors that lead to category mis-assignments, maintaining as far as possible the PCTB bracketing.

269 derivations (0.95%) contain unresolved traces, resulting from annotation noise and rare constructions (such as ellipsis) not currently handled by our translation algorithm. In 468 (1.66%) derivations, residues of PCTB tags not eliminated by the translation algorithm generate malformed categories outside the allowed set (Table 2). Excluding these cases, our conversion algorithm results in a corpus of 27,759 (98.1%) valid derivations.

7.2 Category set

The Chinese CCGbank category set is compared against existing CCG corpora derived from similar automatic corpus conversions, to determine how well we have generalised over syntactic phenomena in the source corpus.

A total of 1197 categories appear in the final corpus, of which 329 occur at least ten times, and 478 are attested only once. By comparison, English CCGbank, contains 1286 categories, 425 of which occur at least ten times, and 440 only once, while German CCGbank has a category inventory of 2506 categories, with 1018 attested only once.[5]

7.3 Lexicon coverage

Lexical item coverage establishes the extent to which data sparsity due to unseen words is problematic in the source corpus, and hence in any corpus derived from it. Hockenmaier and Steedman (2001) showed that formalisms with rich tagsets, such as CCG, are particularly sensitive to this sparsity – while a lexical item may be attested in the training data, it may lack the necessary category.

We divided the 27,759 valid derivations into ten contiguous sections, performing ten-fold cross-validation to determine the coverage of lexical items and CCG categories in the resulting corpus.

Average coverage on lexical items is 73.38%, while average coverage on categories is 88.13%. 94.46% of token types from the held-out set are found in the training set. These figures compare to 86.7% lexical coverage (by type) and 92% (by token) in German CCGbank (Hockenmaier, 2006). Although lexical coverage by token is comparable to the German corpus, we observe a marked difference in coverage by type.

To explain this, we examine the most frequent POS tags among the missing tokens. These are NN (common nouns; 16,552 tokens), NR (proper noun; 8458), VV (verb; 6879), CD (numeral; 1814) and JJ (adjective; 1257). The 100 most frequent missing tokens across the ten folds comprise 48 NR tokens, 46 NR, 3 NT (temporal nouns), 2 JJ (adjectives) and one VA (verbal adjective). Personal names are also not tokenised into surnames and forenames in the PCTB, increasing unseen NR tokens.

The missing VVs (verbs) include 1342 *four-character compounds*, fossilised idiomatic expressions which are considered atomic verbs in the PCTB annotation. Another source of verb sparsity stems from the PCTB analysis of verbal infixation. Given a polysyllabic verb (e.g. 离开 *leave-away* “leave”), we can add the adverbial infix 不 *not* to form a potential verb 离不开 *leave-not-away* “unable to leave”. In the PCTB annotation, however, this results in lexical items for the two cleaved parts, even though 离 *leave* can no longer stand alone as a verb in modern Chinese. In this case, a morphologically decomposed representation which does not split the lexical item could mitigate against this sparsity. Alternatively, candidate verbs for this construction could have the first verb fragment subcategorise for the second.

8 Conclusion

We have developed the first analysis of Chinese with Combinatory Categorial Grammar, crafting novel CCG analyses for a range of constructions including topicalisation, pro-drop, zero copula, verb compounding, and the long-range dependencies resulting from the 把 *ba*- and 被 *bei*-constructions.

We have presented an elegant and economical account of Chinese syntax that exploits the power of CCG combinatory rules, supporting Steedman’s claim to its language-independence.

We have designed a conversion algorithm to extract this analysis from an existing treebank, avoiding the massive cost of hand re-annotation, creating a corpus of 27,759 CCG derivations, covering 98.1% of the PCTB. The corpus will be publicly released, together with the converter, providing the tools to create CCGbanks in new languages.

At release, Chinese CCGbank will include gold-standard head co-indexation data, as required for the training and evaluation of head-driven dependency parsers. Co-indexation analyses, like those provided for the 把 *ba*- and 被 *bei*-constructions, will be extended to all categories.

Future refinements which could be brought to bear on Chinese CCGbank include the integration of PropBank data into CCGbank (Honnibal and Curran, 2007; Boxwell and White, 2008) using Chinese PropBank (Xue, 2008). The *hat categories* of Honnibal and Curran (2009) may better handle form/function discrepancies such as the Chinese zero copula construction, leading to cleaner, more general analyses.

We have presented a wide-coverage Chinese corpus which exploits the strengths of CCG to analyse a range of challenging Chinese constructions. We are now ready to develop rich NLP tools, including efficient, wide-coverage CCG parsers, to address the ever-increasing volumes of Chinese text now available.

[5] All German verbs having at least two categories to account for German verbal syntax contributes to the greater size of the category set (Hockenmaier, 2006).

Acknowledgements

James Curran was supported by Australian Research Council (ARC) Discovery grant DP1097291 and the Capital Markets Cooperative Research Centre.

References

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Stephen Boxwell and Michael White. 2008. Projecting Propbank roles onto the CCGbank. *Proceedings of LREC 2008*.

Michael Burke and Olivia Lam. 2004. Treebank-based acquisition of a Chinese lexical-functional grammar. In *Proceedings of the 18th Pacific Asia Conference on Language, Information and Computation*, pages 161–172.

Aoife Cahill, Mairead McCarthy, Josef van Genabith, and Andy Way. 2002. Automatic annotation of the Penn Treebank with LFG F-structure information. In *LREC 2002 Workshop on Linguistic Knowledge Acquisition and Representation-Bootstrapping Annotated Language Data*, pages 8–15.

Jeongwon Cha, Geunbae Lee, and Jonghyeok Lee. 2002. Korean combinatory categorial grammar and statistical parsing. *Computers and the Humanities*, 36(4):431–453.

John Chen, Srinivas Bangalore, and K. Vijay-Shanker. 2005. Automated extraction of Tree-Adjoining Grammars from treebanks. *Natural Language Engineering*, 12(03):251–299.

Stephen Clark and James R. Curran. 2007. Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. In *Computational Linguistics*, volume 33, pages 493–552.

Yuqing Guo, Josef van Genabith, and Haifeng Wang. 2007. Treebank-based acquisition of LFG resources for Chinese. In *Proceedings of LFG07 Conference*, pages 214–232.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Julia Hockenmaier. 2006. Creating a CCGbank and a wide-coverage CCG lexicon for German. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 505–512. Morristown, NJ, USA.

Julia Hockenmaier and Mark Steedman. 2001. Generative models for statistical parsing with combinatory categorial grammar. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 335–342. Association for Computational Linguistics, Morristown, NJ, USA.

Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 1974–1981.

Julia Hockenmaier and Mark Steedman. 2005. CCGbank: Users' manual. Technical report, MS-CIS-05-09, Computer and Information Science, University of Pennsylvania.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Beryl Hoffman. 1996. *The computational analysis of the syntax and interpretation of free word order in Turkish*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Matthew Honnibal and James R. Curran. 2007. Improving the complement/adjunct distinction in CCGbank. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics (PACLING-07)*, pages 210–217.

Matthew Honnibal and James R. Curran. 2009. Fully Lexicalising CCGbank with Hat Categories. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1212–1221.

C.-T. James Huang, Y.-H. Audrey Li, and Yafei Li. 2008. *The syntax of Chinese*. Cambridge University Press.

Roger Levy and Christopher Manning. 2003. Is it harder to parse Chinese, or the Chinese Treebank? In *Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 439–446. Morristown, NJ, USA.

Charles N. Li and Sandra A. Thompson. 1989. *Mandarin Chinese: A functional reference grammar*. University of California Press.

David M. Magerman. 1994. *Natural language parsing as statistical pattern recognition*. Ph.D. thesis, Stanford University.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 2004. Corpus-Oriented Grammar Development for Acquiring a Head-Driven Phrase Structure Grammar from the Penn Treebank. pages 684–693.

Ad Neeleman and Kriszta Szendrői. 2007. Radical pro drop and the morphology of pronouns. *Linguistic Inquiry*, 38(4):671–714.

Mark Steedman. 1987. Combinatory grammars and parasitic gaps. *Natural Language & Linguistic Theory*, 5(3):403–439.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press. Cambridge, MA, USA.

Henry H.Y. Tiee and Donald M. Lance. 1986. *A reference grammar of Chinese sentences with exercises*. University of Arizona Press.

David Vadas and James R. Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Association for Computational Linguistics*, volume 45, page 240.

Xiu-Zhi Zoe Wu. 2004. *Grammaticalization and language change in Chinese: A formal view*. Routledge.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of Natural Language Processing Pacific Rim Symposium '99*, pages 398–403.

Nianwen Xue. 2008. Labeling chinese predicates with semantic roles. *Computational Linguistics*, 34(2):225–255.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(02):207–238.

Nianwen Xue, Fei Xia, Shizhe Huang, and Anthony Kroch. 2000. The Bracketing Guidelines for the Penn Chinese Treebank (3.0). *IRCS Report 00-08, University of Pennsylvania*.

Po Ching Yip and Don Rimmington. 1997. *Chinese: An essential grammar*. Routledge.

Dependency Forest for Statistical Machine Translation

Zhaopeng Tu [†] **Yang Liu** [†] **Young-Sook Hwang** [‡] **Qun Liu** [†] **Shouxun Lin** [†]

[†]Key Lab. of Intelligent Info. Processing
Institute of Computing Technology
Chinese Academy of Sciences
{tuzhaopeng,yliu,liuqun,sxlin}@ict.ac.cn

[‡]HILab Convergence Technology Center
C&I Business
SKTelecom
yshwang@sktelecom.com

Abstract

We propose a structure called *dependency forest* for statistical machine translation. A dependency forest compactly represents multiple dependency trees. We develop new algorithms for extracting string-to-dependency rules and training dependency language models. Our forest-based string-to-dependency system obtains significant improvements ranging from 1.36 to 1.46 BLEU points over the tree-based baseline on the NIST 2004/2005/2006 Chinese-English test sets.

1 Introduction

Dependency grammars have become increasingly popular in syntax-based statistical machine translation (SMT). One important advantage of dependency grammars is that they directly capture the dependencies between words, which are key to resolving most parsing ambiguities. As a result, incorporating dependency trees proves to be effective in improving statistical machine translation (Quirk et al., 2005; Ding and Palmer, 2005; Shen et al., 2008).

However, most dependency-based translation systems suffer from a major drawback: they only use 1-best dependency trees for rule extraction, dependency language model training, and decoding, which potentially introduces translation mistakes due to the propagation of parsing errors (Quirk and Corston-Oliver, 2006). While the treelet system (Quirk et al., 2005) takes a dependency tree as input, the string-to-dependency system (Shen et al., 2008) decodes on a source-language string. However, as we will show, the string-to-dependency system still commits to using degenerate rules and dependency language models learned from noisy 1-best trees.

To alleviate this problem, an obvious solution is to offer more alternatives. Recent studies have shown that SMT systems can benefit from widening the annotation pipeline: using packed forests instead of 1-best trees (Mi and Huang, 2008), word lattices instead of 1-best segmentations (Dyer et al., 2008), and weighted alignment matrices instead of 1-best alignments (Liu et al., 2009).

Along the same direction, we propose a structure called *dependency forest*, which encodes exponentially many dependency trees compactly, for dependency-based translation systems. In this paper, we develop two new algorithms for extracting string-to-dependency rules and for training dependency language models, respectively. We show that using the rules and dependency language models learned from dependency forests leads to consistent and significant improvements over that of using 1-best trees on the NIST 2004/2005/2006 Chinese-English test sets.

2 Background

Figure 1 shows a dependency tree of an English sentence *he saw a boy with a telescope*. Arrows point from the child to the parent, which is often referred to as the head of the child. For example, in Figure 1, *saw* is the head of *he*. A dependency tree is more compact than its constituent counterpart because there is no need to build a large superstructure over a sentence.

Shen et al. (2008) propose a novel string-to-dependency translation model that features two important advantages. First, they define that a string-to-dependency rule must have a *well-formed* dependency structure on the target side, which makes efficient dynamic programming possible and manages to retain most useful non-constituent rules. A well-formed structure can be either *fixed* or *floating* . A fixed structure is a

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1092–1100,
Beijing, August 2010

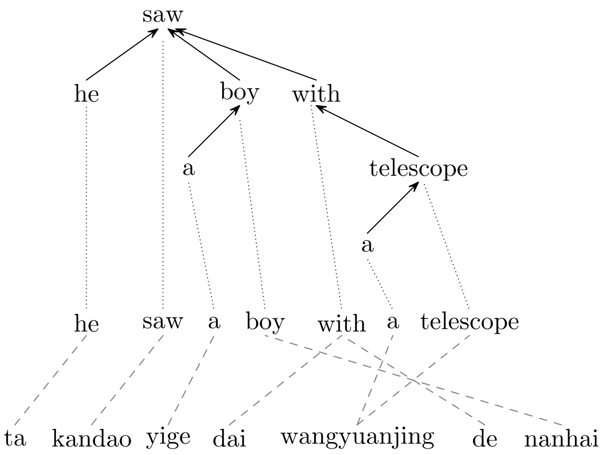


Figure 1: A training example for tree-based rule extraction.

dependency tree with all the children complete. Floating structures consist of sibling nodes of a common head, but the head itself is unspecified or floating. For example, Figure 2(a) and Figure 2(b) are two fixed structures while Figure 2(c) is a floating one.

Formally, for a given sentence $w_{1:l} = w_1 \dots w_l$, $d_1 \dots d_l$ represent the parent word IDs for each word. If w_i is a root, we define $d_i = 0$.

Definition 1. *A dependency structure $d_{i..j}$ is* **fixed on head** *h, where h $\notin [i,j]$, or* **fixed** *for short, if and only if it meets the following conditions*

- $d_h \notin [i,j]$
- $\forall k \in [i,j]$ and $k \neq h, d_k \in [i,j]$
- $\forall k \notin [i,j]$, $d_k = h$ or $d_k \notin [i,j]$

Definition 2. *A dependency structure $d_{i..j}$ is* **floating with children** *C, for a non-empty set C* $\subseteq \{i, ..., j\}$*, or* **floating** *for short, if and only if it meets the following conditions*

- $\exists h \notin [i,j], s.t. \forall k \in C, d_k = h$
- $\forall k \in [i,j]$ and $k \notin C, d_k \in [i,j]$
- $\forall k \notin [i,j]$, $d_k \notin [i,j]$

A dependency structure is **well-formed** if and only if it is either **fixed** or **floating**.

2.1 Tree-based Rule Extraction

Figure 1 shows a training example consisting of an English dependency tree, its Chinese translation, and the word alignments between them. To facilitate identifying the correspondence between the English and Chinese words, we also gives the English sentence. Extracting string-to-dependency rules from aligned string-dependency pairs is similar to extracting SCFG (Chiang, 2007) except that the target side of a rule is a well-formed structure. For example, we can first extract a string-to-dependency rule that is consistent with the word alignment (Och and Ney, 2004):

with ((*a*) *telescope*) $\rightarrow$ *dai wangyuanjing de*

Then a smaller rule

(*a*) *telescope* $\rightarrow$ *wangyuanjing*

can be subtracted to obtain a rule with one non-terminal:

with (X_1) $\rightarrow$ *dai* X_1 *de*

where X is a non-terminal and the subscript indicates the correspondence between non-terminals on the source and target sides.

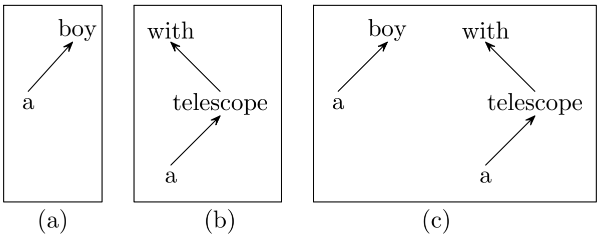


Figure 2: Well-formed dependency structures corresponding to Figure 1. (a) and (b) are fixed and (c) is floating.

2.2 Tree-based Dependency Language Model

As dependency relations directly model the semantics structure of a sentence, Shen et al. (2008) introduce *dependency language model* to better account for the generation of target sentences. Compared with the conventional n-gram language models, dependency language model excels at capturing non-local dependencies between words (e.g., *saw ... with* in Figure 1). Given a dependency tree, its dependency language model probability is a product of three sub-models defined between headwords and their dependants. For example, the probability of the tree in Figure 1 can

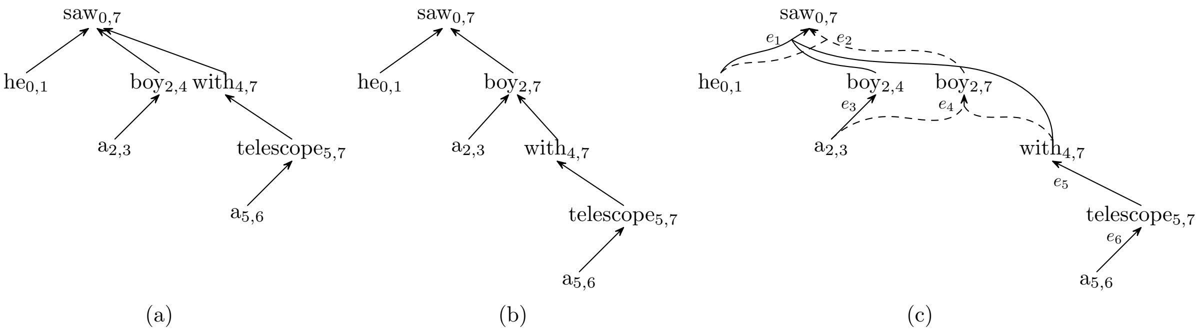


Figure 3: (a) the dependency tree in Figure 1, (b) another dependency tree for the same sentence, and (c) a dependency forest compactly represents the two trees.

be calculated as:

$$\begin{aligned} Prob &= P_T(saw) \\ &\quad \times P_L(he|saw\text{-as-head}) \\ &\quad \times P_R(boy|saw\text{-as-head}) \\ &\quad \times P_R(with|boy, saw\text{-as-head}) \\ &\quad \times P_L(a|boy\text{-as-head}) \\ &\quad \times P_R(telescope|with\text{-as-head}) \\ &\quad \times P_L(a|telescope\text{-as-head}) \end{aligned}$$

where $P_T(x)$ is the probability of word x being the root of a dependency tree. P_L and P_R are the generative probabilities of left and right sides respectively.

As the string-to-tree system relies on 1-best trees for parameter estimation, the quality of rule table and dependency language model might be affected by parsing errors and therefore ultimately results in translation mistakes.

3 Dependency Forest

We propose to encode multiple dependency trees in a compact representation called dependency forest, which offers an elegant solution to the problem of parsing error propagation.

Figures 3(a) and 3(b) show two dependency trees for the example English sentence in Figure 1. The prepositional phrase *with a telescope* could either depend on *saw* or *boy*. Figure 3(c) is a dependency forest compactly represents the two trees by sharing common nodes and edges.

Each **node** in a dependency forest is a word. To distinguish among nodes, we attach a **span** to each node. For example, in Figure 1, the span of the first *a* is $(2, 3)$ because it is the third word in the sentence. As the fourth word *boy* dominates the node $a_{2,3}$, it can be referred to as $boy_{2,4}$. Note that the position of *boy* itself is taken into consideration. Similarly, the word *boy* in Figure 3(b) can be represented as $boy_{2,7}$.

The nodes in a dependency forest are connected by **hyperedges**. While an edge in a dependency tree only points from a dependent to its head, a hyperedge groups all the dependants that have a common head. For example, in Figure 3(c), the hyperedge

$$e_1\text{: } \langle (he_{0,1}, boy_{2,4}, with_{4,7}), saw_{0,7} \rangle$$

denotes that $he_{0,1}$, $boy_{2,4}$, and $with_{4,7}$ are dependants (from left to right) of $saw_{0,7}$.

More formally, a *dependency forest* is a pair $\langle V, E \rangle$, where V is a set of nodes, and E is a set of hyperedges. For a given sentence $w_{1:l} = w_1 \ldots w_l$, each node $v \in V$ is in the form of $w_{i,j}$, which denotes that w dominates the substring from positions i through j (i.e., $w_{i+1} \ldots w_j$). Each hyperedge $e \in E$ is a pair $\langle tails(e), head(e) \rangle$, where $head(e) \in V$ is the head and $tails(e) \in V$ are its dependants.

A dependency forest has a structure of a *hypergraph* such as packed forest (Klein and Manning, 2001; Huang and Chiang, 2005). However, while each hyperedge in a packed forest naturally treats the corresponding PCFG rule probability as its weight, it is challenging to make dependency forest to be a weighted hypergraph because dependency parsers usually only output a score, which can be either positive or negative, for each edge in a dependency tree rather than a hyperedge in a

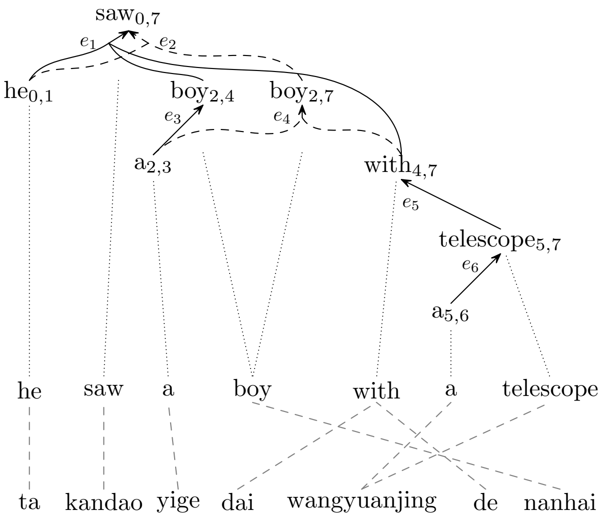


Figure 4: A training example for forest-based rule extraction.

dependency forest. For example, in Figure 3(a), the scores for the edges *he* → *saw*, *boy* → *saw*, and *with* → *saw* could be 13, 22, and -12, respectively.

To assign a probability to each hyperedge, we can first obtain a positive number for a hyperedge using the scores of the corresponding edges:[1]

$$c(e) = exp\left(\frac{\sum_{v \in tails(e)} s\big(v, head(e)\big)}{|tails(e)|}\right) \qquad (1)$$

where $c(e)$ is the count of a hyperedge e, $head(e)$ is a head, $tails(e)$ is a set of dependants of the head, v is one dependant, and $s(v, head(e))$ is the score of an edge from v to $head(e)$. For example, the count of the hyperedge e_1 in Figure 3(c) is

$$c(e_1) = exp\left(\frac{13 + 22 - 12}{3}\right) \qquad (2)$$

Then, the probability of a hyperedge can be obtained by normalizing the count among all hyperedges with the same head collected from a training corpus:

$$p(e) = \frac{c(e)}{\sum_{e':head(e')=head(e)} c(e')} \qquad (3)$$

Therefore, we obtain a weighted dependency forest in which each hyperedge has a probability.

Algorithm 1 Forest-based Initial Phrase Extraction

```
Input: a source sentence $\psi$, a forest $F$, an alignment $a$, and $k$
Output: minimal initial phrase set $\mathcal{R}$
for each node $v \in V$ in a bottom-up order do
    for each hyperedge $e \in E$ and $head(e) = v$ do
        $W \leftarrow \emptyset$
        fixs $\leftarrow$ EnumFixed($v, modifiers(e)$)
        floatings $\leftarrow$ EnumFloating($modifiers(e)$)
        add structures fixs, floatings to $W$
        for each $\omega \in W$ do
            if $\omega$ is consistent with $a$ then
                generate a rule $r$
                $\mathcal{R}$.append($r$)
    keep $k$-best dependency structures for $v$
```

4 Forest-based Rule Extraction

In tree-based rule extraction, one just needs to first enumerate all bilingual phrases that are consistent with word alignment and then check whether the dependency structures over the target phrases are well-formed. However, this algorithm fails to work in the forest scenario because there are usually exponentially many well-formed structures over a target phrase.

The GHKM algorithm (Galley et al., 2004), which is originally developed for extracting tree-to-string rules from 1-best trees, has been successfully extended to packed forests recently (Mi and Huang, 2008). The algorithm distinguishes between minimal and composed rules. Although there are exponentially many composed rules, the number of minimal rules extracted from each node is rather limited (e.g., one or zero). Therefore, one can obtain promising composed rules by combining minimal rules.

Unfortunately, the GHKM algorithm cannot be applied to extracting string-to-dependency rules from dependency forests. This is because the GHKM algorithm requires a complete subtree to exist in a rule while neither fixed nor floating dependency structures ensure that all dependants of a head are included. For example, the floating structure shown in Figure 2(c) actually contains two trees.

Alternatively, our algorithm searches for well-formed structures for each node in a bottom-up style. Algorithm 1 shows the algorithm for extracting initial phrases, that is, rules without non-

[1] It is difficult to assign a probability to each hyperedge. The current method is arbitrary, and we will improve it in the future.

terminals from dependency forests. The algorithm maintains k-best well-formed structures for each node (line 11). The well-formed structures of a head can be constructed from those of its dependants. For example, in Figure 4, as the fixed structure rooted at *telescope*$_{5,7}$ is

$$(a)\ \textit{telescope}$$

we can obtain a fixed structure rooted for the node *with*$_{4,7}$ by attaching the fixed structure of its dependant to the node (*EnumFixed* in line 4). Figure 2(b) shows the resulting fixed structure.

Similarly, the floating structure for the node *saw*$_{0,7}$ can be obtained by concatenating the fixed structures of its dependants *boy*$_{2,4}$ and *with*$_{4,7}$ (*EnumFloating* in line 5). Figure 2(c) shows the resulting fixed structure. The algorithm is similar to Wang et al. (2007), which binarize each constituent node to create some intermediate nodes that correspond to the floating structures.

Therefore, we can find k-best fixed and floating structures for a node in a dependency forest by manipulating the fixed structures of its dependants. Then we can extract string-to-dependency rules if the dependency structures are consistent with the word alignment.

How to judge a well-formed structure extracted from a node is better than others? We follow Mi and Huang (2008) to assign a **fractional count** to each well-formed structure. Given a tree fragment t, we use the inside-outside algorithm to compute its posterior probability:

$$\begin{aligned} \alpha\beta(t) &= \alpha(root(t)) \times \prod_{e \in t} p(e) \\ &\quad \times \prod_{v \in leaves(t)} \beta(v) \end{aligned} \tag{4}$$

where $root(t)$ is the root of the tree, e is an edge, $leaves(t)$ is a set of leaves of the tree, $\alpha(\cdot)$ is outside probability, and $\beta(\cdot)$ is inside probability.

For example, the subtree rooted at *boy*$_{2,7}$ in Figure 4 has the following posterior probability:

$$\begin{aligned} &\alpha(\textit{boy}_{2,7}) \times p(e_4) \times p(e_5) \\ &\times p(e_6) \times \beta(\textit{a}_{2,3}) \times \beta(\textit{a}_{5,6}) \end{aligned} \tag{5}$$

Now the fractional count of the subtree t is

$$c(t) = \frac{\alpha\beta(t)}{\alpha\beta(TOP)} \tag{6}$$

where TOP denotes the root node of the forest.

As a well-formed structure might be non-constituent, we approximate the fractional count by taking that of the minimal constituent tree fragment that contains the well-formed structure. Finally, the fractional counts of well-formed structures can be used to compute the relative frequencies of the rules having them on the target side (Mi and Huang, 2008):

$$\phi(r|lhs(r)) = \frac{c(r)}{\sum_{r':lhs(r')=lhs(r)} c(r')} \tag{7}$$

$$\phi(r|rhs(r)) = \frac{c(r)}{\sum_{r':rhs(r')=rhs(r)} c(r')} \tag{8}$$

Often, our approach extracts a large amount of rules from training corpus as we usually retain exponentially many well-formed structures over a target phrase. To maintain a reasonable rule table size, we discard any rule that has a fractional count lower that a threshold t.

5 Forest-based Dependency Language Model Training

Dependency language model plays an important role in string-to-dependency system. Shen et al. (2008) show that string-to-dependency system achieves 1.48 point improvement in BLEU along with dependency language model, while no improvement without it. However, the string-to-dependency system still commits to using dependency language model from noisy 1-best trees. We now turn to dependency forest for it encodes multiple dependency trees.

To train a dependency language model from a dependency forest, we need to collect all heads and their dependants. This can be easily done by enumerating all hyperedges. Similarly, we use the inside-outside algorithm to compute the posterior probability of each hyperedge e,

$$\begin{aligned} \alpha\beta(e) &= \alpha(head(e)) \times p(e) \\ &\quad \times \prod_{v \in tailes(e)} \beta(v) \end{aligned} \tag{9}$$

For example, the posterior probability of the hyperedge e_2 in Figure 4 is calculated as

$$\begin{aligned} \alpha\beta(e_2) &= \alpha(\textit{saw}_{0,7}) \times p(e_2) \\ &\quad \times \beta(\textit{he}_{0,1}) \times \beta(\textit{boy}_{2,7}) \end{aligned} \tag{10}$$

Rule	DepLM	NIST 2004	NIST 2005	NIST 2006	time
tree	tree	33.97	30.21	30.73	19.6
tree	forest	34.42*	31.06*	31.37*	24.1
forest	tree	34.60*	31.16*	31.45*	21.7
forest	forest	**35.33****	**31.57****	**32.19****	28.5

Table 1: BLEU scores and average decoding time (second/sentence) on the Chinese-English test sets. The baseline system (row 2) used the rule table and dependency language model learned both from 1-best dependency trees. We use " *" and "**" to denote a result is better than baseline significantly at $p < 0.05$ and $p < 0.01$, respectively.

Then, we can obtain the fractional count of a hyperedge e,

$$c(e) = \frac{\alpha\beta(e)}{\alpha\beta(TOP)} \tag{11}$$

Each n-gram (e.g., "boy-as-head a") is assigned the same fractional count of the hyperedge it belongs to.

We also tried training dependency language model as in (Shen et al., 2008), which means all hyperedges were on equal footing without regarding probabilities. However, the performance is about 0.8 point lower in BLEU. One possbile reason is that hyperedges with probabilities could distinguish high quality structures better.

6 Experiments

6.1 Results on the Chinese-English Task

We used the FBIS corpus (6.9M Chinese words + 8.9M English words) as our bilingual training corpus. We ran GIZA++ (Och and Ney, 2000) to obtain word alignments. We trained a 4-gram language model on the Xinhua portion of GIGAWORD corpus using the SRI Language Modeling Toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing (Kneser and Ney, 1995). We optimized feature weights using the minimum error rate training algorithm (Och and Ney, 2002) on the NIST 2002 test set. We evaluated the translation quality using case-insensitive BLEU metric (Papineni et al., 2002) on the NIST 2004/2005/2006 test sets.

To obtain dependency trees and forests, we parsed the English sentences of the FBIS corpus using a shift-reduce dependency parser that enables beam search (Huang et al., 2009). We only retained the best well-formed structure for each node when extracting string-to-tree rules from dependency forests (i.e., $k = 1$). We trained two 3-gram depLMs (one from trees and another from forests) on English side of FBIS corpus plus 2M sentence pairs from other LDC corpus.

Rules	Size	New Rules
tree	7.2M	-
forest	7.6M	16.86%

Table 2: Statistics of rules. The last column shows the ratio of rules extracted from non 1-best parses being used in 1-best derivations.

After extracting rules and training depLMs, we ran our replication of string-to-dependency system (Shen et al., 2008) to translate the development and test sets.

Table 1 shows the BLEU scores on the test sets. The first column "Rule" indicates where the string-to-dependency rules are learned from: 1-best dependency trees or dependency forests. Similarly, the second column "DepLM" also distinguish between the two sources for training dependency language models. The baseline system used the rule table and dependency language model both learned from 1-best dependency trees. We find that adding the rule table and dependency language models obtained from dependency forests improves string-to-dependency translation consistently and significantly, ranging from +1.3 to +1.4 BLEU points. In addition, using the rule table and dependency language model trained from forest only increases decoding time insignificantly.

How many rules extracted from non 1-best

Rule	DepLM	BLEU
tree	tree	22.31
tree	forest	22.73*
forest	tree	22.80*
forest	forest	**23.12****

Table 3: BLEU scores on the Korean-Chinese test set.

parses are used by the decoder? Table 2 shows the number of rules filtered on the test set. We observe that the rule table size hardly increases. One possible reason is that we only keep the best dependency structure for each node. The last row shows that 16.86% of the rules used in 1-best derivations are extracted from non 1-best parses in the forests, indicating that some useful rules cannot be extracted from 1-best parses.

6.2 Results on the Korean-Chinese Task

To examine the efficacy of our approach on different language pairs, we carried out an experiment on Korean-Chinese translation. The training corpus contains about 8.2M Korean words and 7.3M Chinese words. The Chinese sentences were used to train a 5-gram language model as well as a 3-gram dependency language model. Both the development and test sets consist of 1,006 sentences with single reference. Table 3 shows the BLEU scores on the test set. Again, our forest-based approach achieves significant improvement over the baseline ($p < 0.01$).

6.3 Effect of K-best

We investigated the effect of different k-best structures for each node on translation quality (BLEU scores on the NIST 2005 set) and the rule table size (filtered for the tuning and test sets), as shown in Figure 5. To save time, we extracted rules just from the first 30K sentence pairs of the FBIS corpus. We trained a language model and depLMs on the English sentences. We used 10 different k: 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10. Obviously, the higher the k is, the more rules are extracted. When k=10, the number of rules used on the tuning and test sets was 1,299,290 and the BLEU score was 20.88. Generally, both the number of rules and the BLEU score went up with the increase of k. However, this trend did not hold within the range [4,10]. We conjecture that when retaining more dependency structures for each node, low quality structures would be introduced, resulting in much rules of low quality.

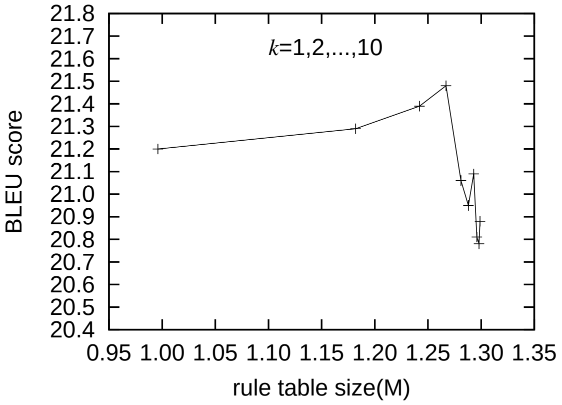


Figure 5: Effect of k-best on rule table size and translation quality.

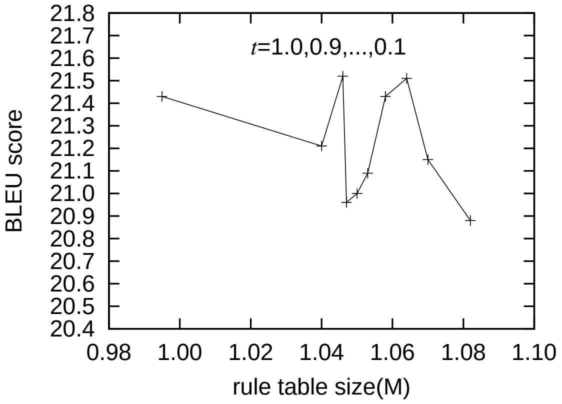


Figure 6: Effect of pruning threshold on rule table size and translation quality.

An interesting finding is that the rule table grew rapidly when k is in range [1,4], while gradually within the range [4,10]. One possible reason is that there are limited different dependency structures in the spans with a maximal length of 10, which the target side of rules cover.

6.4 Effect of Pruning Threshold

Figure 6 shows the effect of pruning threshold on translation quality and the rule table size. We retained 10-best dependency structures for each node in dependency forests. We used 10 different pruning thresholds: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0. Intuitively, the higher the pruning threshold is, the less rules are extracted. When t=0.1, the number of rules used on the tuning and test sets was 1,081,841 and the BLEU score was 20.68.

Lots of rules are pruned when the pruning threshold increases from 0.0 to 0.3 (around 20%). After pruning away these rules, we achieved 0.6 point improvement in BLEU. However, when we filtered more rules, the BLEU score went down.

Figures 5 and 6 show that using two parameters that have to be hand-tuned achieves a small improvement at the expense of an additional complexity. To simplify the approach, we only keep the best dependency structure for each node without pruning any rule.

7 Related Works

While Mi and Huang (2008) and we both use forests for rule extraction, there remain two major differences. Firstly, Mi and Huang (2008) use a packed forest, while we use a dependency forest. Packed forest is a natural weighted hypergraph (Klein and Manning, 2001; Huang and Chiang, 2005), for each hyperedge treats the corresponding PCFG rule probability as its weight. However, it is challenging to make dependency forest to be a weighted hypergraph because dependency parsers usually only output a score for each edge in a dependency tree rather than a hyperedge in a dependency forest. Secondly, The GHKM algorithm (Galley et al., 2004), which is originally developed for extracting tree-to-string rules from 1-best trees, has been successfully extended to packed forests recently (Mi and Huang, 2008). Unfortunately, the GHKM algorithm cannot be applied to extracting string-to-dependency rules from dependency forests, because the GHKM algorithm requires a complete subtree to exist in a rule while neither fixed nor floating dependency structures ensure that all dependants of a head are included.

8 Conclusion and Future Work

In this paper, we have proposed to use dependency forests instead of 1-best parses to extract string-to-dependency tree rules and train dependency language models. Our experiments show that our approach improves translation quality significantly over a state-of-the-art string-to-dependency system on various language pairs and test sets. We believe that dependency forest can also be used to improve the dependency treelet system (Quirk et al., 2005) that takes 1-best trees as input.

Acknowledgement

The authors were supported by SK Telecom C&I Business, and National Natural Science Foundation of China, Contracts 60736014 and 60903138. We thank the anonymous reviewers for their insightful comments. We are also grateful to Wenbin Jiang for his invaluable help in dependency forest.

References

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, pages 201–228.

Ding, Yuan and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of ACL*.

Dyer, Christopher, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL*.

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of NAACL*.

Huang, Liang and David Chiang. 2005. Better k-best parsing. In *Proceedings of IWPT*.

Huang, Liang, Wenbin Jiang, and Qun Liu. 2009. Bilingually-constrained (monolingual) shift-reduce parsing. In *Proceedings of EMNLP*.

Klein, Dan and Christopher D. Manning. 2001. Parsing and hypergraphs. In *Proceedings of IWPT*.

Kneser, R. and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of Acoustics, Speech, and Signal*.

Liu, Yang, Tian Xia, Xinyan Xiao, and Qun Liu. 2009. Weighted alignment matrices for statistical machine translation. In *Proceedings of EMNLP*.

Mi, Haitao and Liang Huang. 2008. Forest-based translation rule extraction. In *Proceedings of EMNLP*.

Och, Franz J. and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL*.

Och, Franz J. and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*.

Och, Franz J. and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Quirk, Chris and Simon Corston-Oliver. 2006. The impact of parsing quality on syntactically-informed statistical machine translation. In *Proceedings of EMNLP*.

Quirk, Chris, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: syntactically informed phrasal smt. In *Proceedings of ACL*.

Shen, Libin, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL*.

Stolcke, Andreas. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of ICSLP*.

Wang, Wei, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of EMNLP*.

Large Scale Parallel Document Mining for Machine Translation

Jakob Uszkoreit Jay M. Ponte Ashok C. Popat Moshe Dubiner

Google, Inc.
{uszkoreit,ponte,popat,moshe}@google.com

Abstract

A distributed system is described that reliably mines parallel text from large corpora. The approach can be regarded as cross-language near-duplicate detection, enabled by an initial, low-quality batch translation. In contrast to other approaches which require specialized metadata, the system uses only the textual content of the documents. Results are presented for a corpus of over two billion web pages and for a large collection of digitized public-domain books.

1 Introduction

While the World Wide Web provides an abundance of readily available monolingual text, parallel data is still a comparatively scarce resource, yet plays a crucially important role in training statistical machine translation systems.

We describe an approach to mining document-aligned parallel text to be used as training data for a statistical machine translation system. Previous approaches have focused on rather homogeneous corpora and relied on metadata such as publication dates (Munteanu and Marcu, 2005; Munteanu and Marcu, 2006; Udupa et al., 2009; Do et al., 2009; Abdul-Rauf and Schwenk, 2009) or information about document structure (Resnik and Smith, 2003; Chen and Nie, 2000). In large and unstructured collections of documents such as the Web, however, metadata is often sparse or unreliable. Our approach, in contrast, scales computationally to very large and diverse collections of documents and does not require metadata. It is based solely on the textual contents of the input documents.

Casting the problem as one of cross-language near duplicate detection, we use a baseline machine translation system to translate all input documents into a single language. However, the words and phrases that are most discriminatory for the purposes of information retrieval and duplicate detection are the relatively rare ones, precisely those that are less likely to be translated well by the baseline translation system.

Our approach to circumvent this problem and to avoid the prohibitive quadratic computational complexity of the naive approach of performing a comparison of every possible pair of input documents is similar to previous work in near duplicate detection (Broder, 2000; Henzinger, 2006; Manber, 1994) and noisy data retrieval (Harding et al., 1997).

We use shingles consisting of word n-grams to construct relatively rare features from more common, in-vocabulary words. For each input document, we identify a comparatively small set of candidate pairings with documents sharing at least a certain number of such features. We then perform a more expensive comparison between each document and all documents in its candidate set using lower order n-gram features that would typically be too frequent to be used efficiently in forming candidate pairings, but provide a higher coverage of the scored document pairs. Another important aspect of our approach is that it can be implemented in a highly parallel way, as we describe in the following section.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1101–1109,
Beijing, August 2010

2 System Description

The input is a set of documents from diverse sources such as web pages and digitized books. In a first stage, all documents are independently translated into English using a baseline statistical machine translation system.

We then extract two different sets of n-grams from the translated documents: matching n-grams that are used to construct the candidate sets as well as scoring n-grams used only in the computation of a score for a given pair of documents. This stage generates two indexes: a *forward index* listing all extracted scoring n-grams, indexed by document; and an *inverted index* referencing all documents from which we extracted a given matching n-gram, indexed by n-grams. The inverted index is also used to accumulate global information about scoring n-grams, such as their document frequency, yet for scoring n-grams we do not accumulate a posting list of all documents in which they occur.

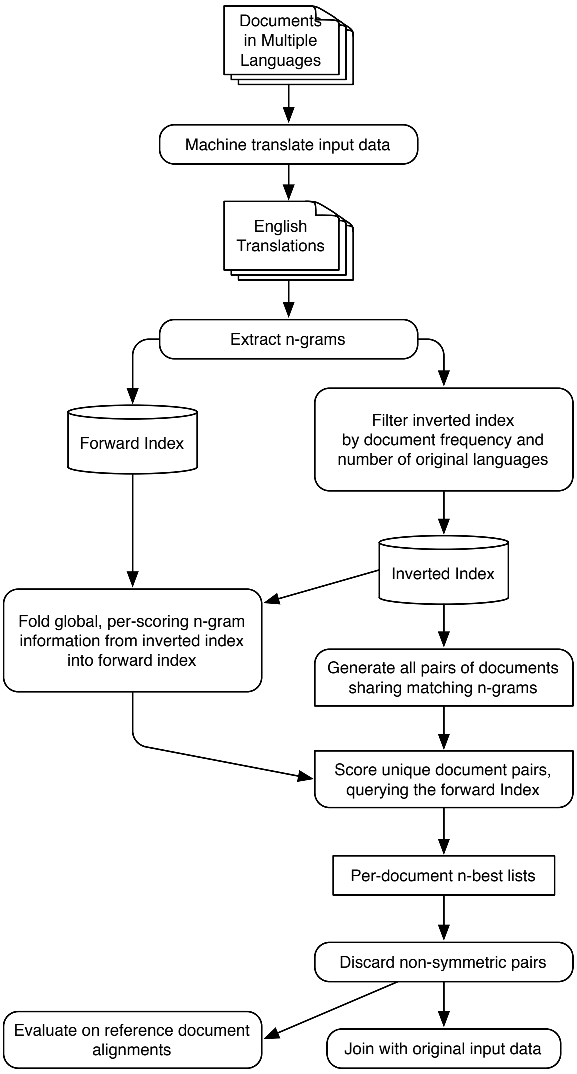

Documents
in Multiple
Languages
Machine translate input data
English
Translations
Extract n-grams
Forward Index
Filter inverted index
by document frequency and
number of original languages
Inverted Index
Fold global, per-scoring n-gram
information from inverted index
into forward index
Generate all pairs of documents
sharing matching n-grams
Score unique document pairs,
querying the forward Index
Per-document n-best lists
Discard non-symmetric pairs
Evaluate on reference document
alignments
Join with original input data

Figure 1: Architecture of the Parallel Text Mining System.

In the next step, the system generates all possible pairs of documents for each matching n-gram posting list in the inverted index. Since we keep only those pairs of documents that originated in different languages, we can discard posting lists from the inverted index that contain only a single document, i.e. those of singleton n-grams, or only documents in a single language.

Crucially, we further discard posting lists for matching n-grams whose frequency exceeds a certain threshold. When choosing a sufficiently large order for the matching n-grams, their long-tailed distribution causes only a small fraction of matching n-grams to be filtered out due to frequency, as we show empirically in Section 5. It is this filtering step that causes the overall runtime of the system to be linear in the size of the input data and allows the system to scale to very large document collections.

In parallel, global information about scoring n-grams accumulated in the inverted index that is required for pairwise scoring, such as their document frequency, is folded into the forward index by iterating over all forward index entries, requesting the respective per-feature quantities from the inverted index and storing them with each occurrence of a scoring n-gram in an updated forward index.

In the next stage, we compute pairwise scores for all candidate document pairs, accessing the forward index entry of each of the two scored documents to obtain the respective scoring n-grams. Document pairs with a score below a given threshold are discarded. For each input document, this results in one n-best list per language. In the last step we retain only those document pairs where each document is contained in the n-best list of the other document for its original language. Finally we perform a *join* of our identified translation pairs with the original text by making another pass over the original, untranslated input data where the contents of document pairs with sufficiently high scores are then aggregated and output. Document pairings involving all languages are identified simultaneously. Each stage of the system fits well into the MapReduce programming model (Dean and Ghemawat, 2004). The general architecture is shown in Figure 1.

2.1 Pairwise Scoring

For scoring a pair of documents d and d', the forward index is queried for the entries for both documents. Let $F_d = \{f_1, f_2, ...f_n\}$ and $F_{d'} = \{f'_1, f'_2, ...f'_{n'}\}$ be the sets of scoring n-grams in the forward index entries of d and d', respectively. Let $\text{idf}(f) = \log \frac{|D|}{df(f)}$ be the inverse document frequency of a scoring n-gram f, where $|D|$ is the number of documents in the input corpus and $df(f)$ is the number documents from which we extracted the feature f. Interpreting F_d and $F_{d'}$ as incidence vectors in the vector space of n-grams and replacing each non-zero component f with $\text{idf}(f)$, we compute the score of the document pair as the inverse document frequency weighted cosine similarity of F_d and $F_{d'}$

$$\text{score}(d, d') = \frac{F_d \cdot F_{d'}}{||F_d|| \cdot ||F_{d'}||} \tag{1}$$

The per-document n-best lists are sorted according to this score and document pairs for which the score is below a threshold are discarded completely.

We do not use term frequency in the scoring metric. In preliminary experiments, incorporating the term frequency to yield basic *tf/idf* as well as using other information retrieval ranking functions incorporating term frequencies such as *BM25* (Robertson et al., 1995) resulted in a degradation of performance compared to the simpler scoring function described above. We believe this is due to the fact that, in contrast to the standard information retrieval setting, the overall length of our queries is on par with that of the documents in the collection.

The scoring is completely agnostic regarding the scoring n-grams' positions in the documents. Since especially for long documents such as books this may produce spurious matches, we apply an additional filter to remove document pairs for which the relative ordering of the matching scoring n-grams is very different. Together with each scoring n-gram we also extract its relative position in each document and store it in the forward index. When scoring a document pair, we compute the normalized permutation edit distance (Cormode et al., 2001) between the two sequences of overlapping n-grams sorted by their position in the respective document. If this distance exceeds a certain threshold, we discard the document pair.

2.2 Computational Complexity

By limiting the frequency of matching n-grams, the complexity becomes linear. Let the tunable parameter c be the maximum occurrence count for matching n-grams to be kept in the inverted index. Let m be the average number of matching n-grams extracted from a single document whose count is below c and D be the set of documents in the input corpus. Then the system generates up to $|D| \cdot m \cdot c$ candidate pairings. Scoring a given candidate document pair according to cosine similarity involves computing three dot-products between sparse vectors with one non-zero component per scoring n-gram extracted and not filtered from the respective document. Let s be the average number of such scoring n-grams per document, which is bounded by the average document length. Then the time complexity of the entire document alignment is in

$$O(|D| \cdot m \cdot c \cdot s) \tag{2}$$

and therefore linear in the number of input documents in the corpus and the average document size.

The space complexity is dominated by the size of the inverted and forward indexes, both of which are linear in the size of the input corpus.

2.3 Sentence-Level Alignment

Further filtering is performed on a per-sentence basis during per-document-pair sentence alignment of the mined text with a standard dynamic programming sentence alignment algorithm using sentence length and multilingual probabilistic dictionaries as features. Afterwards we crudely align words within each pair of aligned source and target sentences. This crude alignment is used only to filter nonparallel sentences. Let S be the set of source words, T the set of target words and $S \times T$ the set of ordered pairs. Let the source sentence contain words $S_0 \subset S$ and the target sentence contain words $T_0 \subset T$. An alignment $A_0 \subset S_0 \times T_0$ will be scored by

$$\mathrm{score}(A_0) = \sum_{(s,t) \in A_0} \ln \frac{p(s,t)}{p(s)\, p(t)} \tag{3}$$

where the joint probabilities $p(s,t)$ and marginal probabilities $p(s)$, $p(t)$ are taken to be the respective empirical distributions (without smoothing) in an existing word aligned corpus. This is greedily maximized and the result is divided by its approximate expected value

$$\sum_{(s,t) \in S_0 \times T} \frac{p(s,t)}{p(s)} \ln \frac{p(s,t)}{p(s)\, p(t)} \tag{4}$$

We discard sentence pairs for which the ratio between the actual and the expected score is less than $1/3$. We also drop sentence pairs for which both sides are identical, or a language detector declares them to be in the wrong language.

2.4 Baseline Translation System

To translate the input documents into English we use phrase-based statistical machine translation systems based on the log-linear formulation of the problem (Och and Ney, 2002).

We train the systems on the Europarl Corpus (Koehn, 2002), the DGT Multilingual Translation Memory (European Commission Directorate-General for Translation, 2007) and the United Nations ODS corpus (United Nations, 2006). Minimum error rate training (Macherey et al., 2008) under the BLEU criterion is used to optimize the feature function weights on development data consisting of the *nv-dev2007* and *news-dev2009* data sets provided by the organizers of the 2007 and 2009 WMT shared translation tasks[1]. We use a 4-gram language model trained on a variety of large monolingual corpora. The BLEU scores of our baseline translation system on the test sets from various WMT shared translation tasks are listed in Table 5. An empirical analysis of the impact of the baseline translation system quality on the data mining system is given in Section 6.3.

3 Input Document Collections

We evaluate the parallel text mining system on two input data sets:

web A collection of 2.5 Billion general pages crawled from the Web, containing only pages in Czech, English, French, German, Hungarian and Spanish

books A collection of 1.5 Million public domain books digitized using an optical character recognition system. The collection consists primarily of English, French and fewer Spanish volumes

3.1 Reference Sets

We created reference sets of groups of documents in multiple languages which are true translations of one another for both the *web* and the *books* data set. Due to the presence of duplicates, each reference pairing can contain more than a single alternative translation per language. The *web* reference set was constructed by exploiting the systematic hyperlink structure of the web-site `http://america.gov/`, that links pages in one language to their respective translations into one or more other languages. The resulting reference set contains documents in Arabic, Chinese, English, French, Russian and Spanish, however, for most English pages there is only one translation into one of the other languages. Overall, the reference set contains 6,818 documents and 7,286 translation pairs.

The *books* reference set contains 30 manually aligned groups of translations covering a total of 103 volumes in English and French.

4 Evaluation Metrics

The fact that the system outputs pairs of documents and the presence of duplicate documents in the corpus motivate the use of modified versions of *precision* and *recall*.

[1] available at `http://statmt.org`

Let C be a set of candidate parallel document pairs and let R be a possibly incomplete reference set of groups of parallel documents known to exist in the corpus. Consider the following two subsets of C:

- *Matching* pairs which are in some reference cluster.
- *Touching* pairs which are non-matching but have at least one document in some reference cluster.

We define

$$\text{Precision} = \frac{|C_{\text{Matching}}|}{|C_{\text{Matching}}| + |C_{\text{Touching}}|}$$

and

$$\text{Recall} = \frac{|C_{\text{Matching}}|}{|R|} \qquad (5)$$

5 Parameter Selection

We conducted a series of small-scale experiments on only those documents contained in the *web* reference data set to empirically determine good settings for the tunable parameters of the text mining system. Among the most important parameters are the orders of the n-grams used for pairing documents as well as scoring them. Aside from the obvious impact on the quality of the output, these parameters have a very large influence on the overall computational performance of the system. The choice of the order of the extracted matching n-grams is mainly a trade-off between recall and efficiency. If the order is too large the system will miss valid pairs; if too small the the threshold on matching n-gram frequency will need to be increased.

Figure 2 shows the F1-scores obtained running only on the documents contained in the *web* reference set with different orders of matching and scoring n-grams. Figure 3 shows the corresponding number of pairwise comparisons made when using different orders of matching n-grams. While there is a drop of 0.01 in F1 score between using 2-grams and 5-grams as matching n-grams, this drop in quality seems to be well worth the 42-fold reduction in resulting pairwise comparisons.

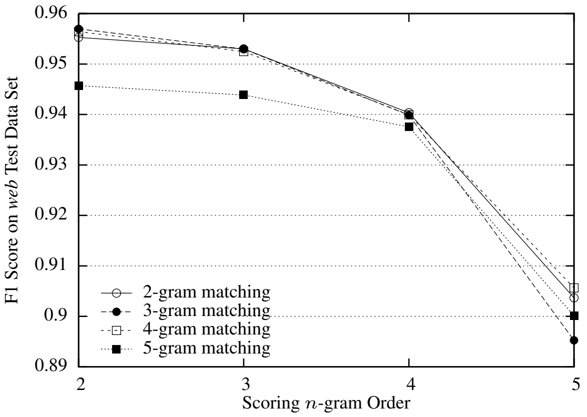


Figure 2: F1 scores on the *web* reference set for different scoring and matching n-gram orders.

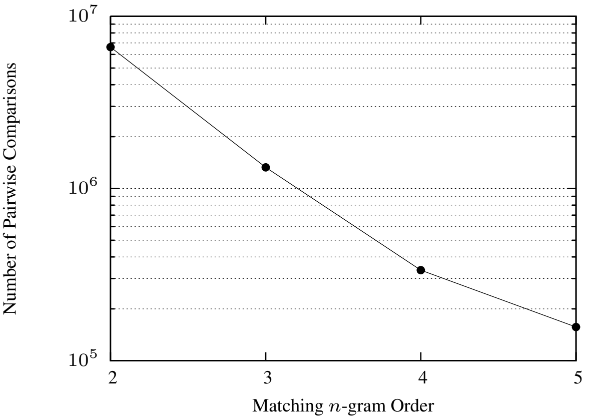


Figure 3: Number of pairwise comparisons made when using matching n-grams of different orders.

The largest portion of the loss in F1 score is incurred when increasing the matching n-gram order from 4 to 5, the reduction in pairwise comparisons, however, is still more than twofold.

Table 1 shows the precision and recall on the *web* reference set when running only on documents in the reference set using 5-grams as matching n-grams and bigrams for scoring for different values of the threshold on the cosine similarity score. In this setting as well as in large-scale experiments on both complete data sets described in section 6.1, a threshold of 0.1 yields the highest F1 score.

score threshold	0.06	0.10	0.12	0.16	0.20
precision	0.92	0.97	0.98	0.99	0.99
recall	0.91	0.91	0.90	0.89	0.83

Table 1: Precision and recall on the *web* reference set when running only on documents contained in the reference set.

6 Evaluation

We run the parallel text mining system on the *web* and *books* data sets using 5-grams for matching and bigrams for scoring. In both cases we discard matching n-grams which occurred in more than 50 documents and output only the highest scoring candidate for each document.

In case of the *web* data set, we extract every 5-gram as potential matching feature. For the *books* data set, however, we downsample the number of candidate matching 5-grams by extracting only those whose integer fingerprints under some hash function have four specific bits set, thus keeping on average only $1/16$ of the matching n-grams. Here, we also restrict the total number of matching n-grams extracted from any given document to 20,000. Scoring bigrams are dropped from the forward index if their document frequency exceeds 100,000, at which point their influence on the pairwise score would be negligible.

Running on the *web* data set, the system on average extracts 250 matching 5-grams per document, extracting a total of approximately 430 Billion distinct 5-grams. Of those, 78% are singletons and 21% only occur in a single language. Only approximately 0.8% of all matching n-grams are filtered due to having a document frequency higher than 50. The forward index initially contains more than 500 Billion bigram occurrences; after pruning out singletons and bigrams with a document frequency larger than 100,000, the number of indexed scoring feature occurrences is reduced to 40%. During scoring, approximately 50 Billion pairwise comparisons are performed.

In total the n-gram extraction, document scoring and subsequent filtering takes less than 24 hours on a cluster of 2,000 state-of-the-art CPUs.

The number of words after sentence-level filtering and alignment that the parallel text mining system extracted for the different languages from each dataset are listed in Table 2.

	baseline	*books*	*web*
Czech	27.5 M	0	271.9 M
French	479.8 M	228.5 M	4,914.3 M
German	54.2 M	0	3,787.6 M
Hungarian	26.9 M	0	198.9 M
Spanish	441.0 M	15.0 M	4,846.8 M

Table 2: The number of words per language in the baseline training corpora and extracted from the two different data sets.

score threshold	0.06	0.10	0.12	0.16	0.20
precision	0.88	0.93	0.95	0.97	0.97
recall	0.68	0.65	0.63	0.52	0.38

Table 3: Precision and recall on the reference set when running on the complete *web* data set with different score thresholds.

score threshold	0.06	0.10	0.12	0.16	0.20
precision	0.95	1.00	1.00	1.00	1.00
recall	0.71	0.71	0.71	0.48	0.38

Table 4: Precision and recall on the reference set when running on the complete *books* data set with different score thresholds.

6.1 Precision and Recall

Tables 3 and 4 show precision and recall on the respective reference sets for the *web* and the *books* input data sets. While the text mining system maintains a very high precision, recall drops significantly compared to running only on the documents in the reference set. One reason for this behavior is that the number of n-grams in the test data set which are sufficiently rare to be used as queries drops with increasing amounts of input data and in particular short documents which only share a small number of matching n-grams anyway, may happen to only share matching n-grams with a too high document frequency. Further analysis shows that another, more significant factor is the existence of multiple, possibly partial translations and near-duplicate documents which cause symmetrization to discard valid document pairs because each document in the pair is determined by the document pair score to be more similar to a different translation of a near-duplicate or sub-

Language Pair	Training Data	WMT 2007 news commentary	WMT 2008 news	WMT 2009 news
Czech English	baseline	21.59	14.59	16.46
	web	29.26 (+7.67)	20.16 (+5.57)	23.25 (+6.76)
German English	baseline	27.99	20.34	20.03
	web	32.35 (+4.36)	23.22 (+2.88)	23.35 (+3.32)
Hungarian English	baseline	-	10.21	11.02
	web	-	12.92 (+2.71)	14.68 (+3.66)
French English	baseline	34.26	22.14	26.39
	books	34.73 (+0.47)	22.39 (+0.25)	27.15 (+0.76)
	web	36.65 (+2.39)	23.22 (+1.08)	28.34 (+1.95)
Spanish English	baseline	43.67	24.15	26.88
	books	44.07 (+0.40)	24.32 (+0.17)	27.16 (+0.28)
	web	46.21 (+2.54)	25.52 (+1.37)	28.50 (+1.62)
English Czech	baseline	14.78	12.45	11.62
	web	20.65 (+5.86)	18.70 (+6.25)	16.60 (+4.98)
English German	baseline	19.89	14.67	14.31
	web	23.49 (+3.60)	16.78 (+2.11)	16.96 (+2.65)
English Hungarian	baseline	-	07.93	08.52
	web	-	10.16 (+2.23)	11.42 (+2.90)
English French	baseline	31.59	22.29	25.14
	books	31.92 (+0.33)	22.42 (+0.13)	25.46 (+0.32)
	web	34.35 (+2.76)	23.56 (+1.27)	27.05 (+1.91)
English Spanish	baseline	42.05	24.65	25.85
	books	42.05	24.79 (+0.14)	26.07 (+0.22)
	web	45.21 (+3.16)	26.46 (+1.81)	27.79 (+1.94)

Table 5: BLEU scores of the translation systems trained on the automatically mined parallel corpora and the baseline training data.

set of the document. This problem seems to affect news articles in particular where there are often multiple different translations of large subsets of the same or slightly changed versions of the article.

6.2 Translation Quality

Arabic English	NIST 2006	NIST 2008
Baseline (UN ODS)	44.31	42.79
Munteanu and Marcu	45.13	43.86
Present work	44.72	43.64

Chinese English	NIST 2006	NIST 2008
Baseline (UN ODS)	25.71	19.79
Munteanu and Marcu	28.11	21.69
Present work	28.08	22.02

Table 6: BLEU scores of the Chinese and Arabic to English translation systems trained on the baseline UN ODS corpus and after adding either the Munteanu and Marcu corpora or the training data mined using the presented approach.

We trained a phrase-based translation system on the mined parallel data sets and evaluated it on translation tasks for the language pairs Czech, French, German, Hungarian and Spanish to and from English, measuring translation quality with the BLEU score (Papineni et al., 2002). The translation tasks evaluated are the WMT 2007 news commentary test set as well the WMT 2008 and 2009 news test sets.

The parallel data for this experiment was mined using the general settings described in the previous section and a threshold of 0.1 on the pairwise score. We ensure that the test data is not included in the training data by filtering out all sentences from the training data that share more than 30% of their 6-grams with any sentence from one of the test corpora.

Table 5 shows the BLEU scores of the different translation systems. The consistent and significant improvements in BLEU score demonstrate the usefulness of the mined document pairs in training a translation system.

Even though the presented approach works on a less granular level than the sentence-level approach of Munteanu and Marcu (2005), we compare results on the same input data[2] used by those authors to automatically generate the

[2] LDC corpora LDC2005T12, LDC2005T14 and LDC2006T02, the second editions of the Arabic, Chinese and English Gigaword corpora.

Sampling Rate	WMT 2007 news commentary			WMT 2008 news			WMT 2009 news		
	degraded	Cz→En	En→Cz	degraded	Cz→En	En→Cz	degraded	Cz→En	En→Cz
1.0	21.59	29.26	20.65	14.59	20.16	18.70	16.46	23.25	16.60
0.5	20.12	29.16	20.55	13.65	20.16	18.71	15.44	23.16	16.56
0.25	18.59	29.09	20.61	12.79	20.09	18.58	14.35	23.18	16.50
0.125	16.69	29.10	20.39	11.87	20.07	18.48	13.05	23.06	16.53
0.0625	14.72	29.04	20.44	10.87	20.06	18.49	11.62	23.11	16.44
0.0312	12.60	28.75	20.28	09.71	19.97	18.45	10.43	23.04	16.41

Table 7: BLEU scores of the degraded Czech to English baseline systems used for translating Czech documents from the *web* data set as well as those of Czech to and from English systems trained on data mined using translations of varying quality created by sampling from the training data.

Arabic English and Chinese English sentence-aligned parallel LDC corpora LDC2007T08 and LDC2007T09. We trained Arabic and Chinese English baseline systems on the United Nations ODS corpus (United Nations, 2006); we also use these to translate the non-English portions of the input data to English. We then evaluate the effects of also training on either the LDC2007T08 and LDC2007T09 corpora or the parallel documents mined by our approach in addition to the United Nations ODS corpus on the NIST 2006 and 2008 MT evaluation test sets. The results are presented in Table 6.

The approach proposed in (Munteanu and Marcu, 2005) relies critically on the existence of publication dates in order to be computationally feasible, yet it still scales superlinearly in the amount of input data. It could therefore not easily be applied to much larger and less structured input data collections. While our approach neither uses metadata nor operates on the sentence level, in all but one of the tasks, the system trained on the data mined using our approach performs similarly or slightly better.

6.3 Impact of Baseline Translation Quality

In order to evaluate the impact of the translation quality of the baseline system on the quality of the mined document pairs, we trained artificially degraded Czech to English translation systems by sampling from the baseline training data at decreasing rates. We translate the Czech subset of the *web* document collection into English with each of the degraded systems and apply the parallel data mining system in the same configuration.

Table 7 shows the BLEU scores of the degraded baseline systems and those resulting from adding the different mined data sets to the non-degraded Czech English and English Czech systems. Degrading the input data translation quality by up to 8.9% BLEU results in a consistent but only comparatively small decrease of less than 0.6% BLEU in the scores obtained when training on the mined document pairs. This does not only show that the impact of variations of the baseline system quality on the data mining system is limited, but also that the data mining system will already work with a rather low quality baseline system.

7 Conclusion

We presented a scalable approach to mining parallel text from collections of billions of documents with high precision. The system makes few assumptions about the input documents. We demonstrated that it works well on different types of data: a large collection of web pages and a collection of digitized books. We further showed that the produced parallel corpora can significantly improve the quality of a state-of-the-art statistical machine translation system.

8 Acknowledgments

We thank the anonymous reviewers for their insightful comments.

References

Abdul-Rauf, Sadaf and Holger Schwenk. 2009. On the use of comparable corpora to improve SMT performance. In *EACL*, pages 16–23.

Broder, Andrei Z. 2000. Identifying and filtering near-duplicate documents. In *COM '00: Proceedings of the 11th Annual Symposium on Combinatorial Pat-*

tern Matching, pages 1–10, London, UK. Springer-Verlag.

Chen, Jiang and Jian-Yun Nie. 2000. Parallel web text mining for cross-language IR. In *In In Proc. of RIAO*, pages 62–77.

Cormode, Graham, S. Muthukrishnan, and Süleyman Cenk Sahinalp. 2001. Permutation editing and matching via embeddings. In *ICALP '01: Proceedings of the 28th International Colloquium on Automata, Languages and Programming,*, pages 481–492, London, UK. Springer-Verlag.

Dean, Jeffrey and Sanjay Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *Proceedings of the Sixth Symposium on Operating System Design and Implementation (OSDI-04)*, San Francisco, CA, USA.

Do, Thi-Ngoc-Diep, Viet-Bac Le, Brigitte Bigi, Laurent Besacier Eric, and Castelli. 2009. Mining a comparable text corpus for a Vietnamese - French statistical machine translation system. In *Proceedings of the 4th EACL Workshop on Statistical Machine Translation*, pages 165–172, Athens, Greece, March.

European Commission Directorate-General for Translation. 2007. DGT-TM parallel corpus. `http://langtech.jrc.it/DGT-TM.html`.

Harding, Stephen M., W. Bruce Croft, and C. Weir. 1997. Probabilistic retrieval of OCR degraded text using n-grams. In *ECDL '97: Proceedings of the First European Conference on Research and Advanced Technology for Digital Libraries*, pages 345–359, London, UK. Springer-Verlag.

Henzinger, Monika. 2006. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 284–291, New York, NY, USA. ACM.

Koehn, Philipp. 2002. Europarl: A multilingual corpus for evaluation of machine translation. Draft.

Macherey, Wolfgang, Franz Och, Ignacio Thayer, and Jakob Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 725–734, Honolulu, Hi, October. Association for Computational Linguistics.

Manber, Udi. 1994. Finding similar files in a large file system. In *Proceedings of the USENIX Winter 1994 Technical Conferenc.*

Munteanu, Dragos Stefan and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Comput. Linguist.*, 31(4):477–504.

Munteanu, Dragos Stefan and Daniel Marcu. 2006. Extracting parallel sub-sentential fragments from non-parallel corpora. In *ACL*.

Och, Franz Josef and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, Philadelphia, PA, USA.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, PA, USA.

Resnik, Philip and Noah A. Smith. 2003. The web as a parallel corpus. *Computational Linguistics*, 29:349–380.

Robertson, S E, S Walker, S Jones, M M Hancock-Beaulieu, and M Gatford. 1995. Okapi at TREC–3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*.

Udupa, Raghavendra, K. Saravanan, A. Kumaran, and Jagadeesh Jagarlamudi. 2009. Mint: A method for effective and scalable mining of named entity transliterations from large comparable corpora. In *EACL*, pages 799–807.

United Nations. 2006. ODS UN parallel corpus. `http://ods.un.org/`.

Hungarian Corpus of Light Verb Constructions

Veronika Vincze
University of Szeged
Department of Informatics
vinczev@inf.u-szeged.hu

János Csirik
Hungarian Academy of Sciences
Research Group on Artificial Intelligence
csirik@inf.u-szeged.hu

Abstract

The precise identification of light verb constructions is crucial for the successful functioning of several NLP applications. In order to facilitate the development of an algorithm that is capable of recognizing them, a manually annotated corpus of light verb constructions has been built for Hungarian. Basic annotation guidelines and statistical data on the corpus are also presented in the paper. It is also shown how applications in the fields of machine translation and information extraction can make use of such a corpus and an algorithm.

1 Introduction

In this paper, we report a corpus containing light verb constructions in Hungarian. These expressions are neither productive nor idiomatic and their meaning is not totally compositional (the noun is usually taken in one of its literal senses but the verb usually loses its original sense to some extent), as it can be seen in the examples from different languages shown below. Since their meaning is the same, only literal translations are provided:

- English: *to give a lecture*, *to come into bloom*, *the problem lies (in)*
- German: *halten eine Vorlesung* to hold a presentation, *in Blüte stehen* in bloom to stand, *das Problem liegt (in)* the problem lies (in)
- French: *faire une présentation* to make a presentation, *être en fleur* to be in bloom, *le problème réside (dans)* the problem resides (in)
- Hungarian: *előadást tart* presentation-ACC holds, *virágba borul* bloom-ILL falls, *probléma rejlik (vmiben)* problem hides (in sg)

Several terms like *complex verb structures*, *support verb constructions* or *light verb constructions* have been used[1] for these constructions in the literature (Langer, 2004). In this paper, the term *light verb constructions* will be employed.

The structure of the paper is as follows. First, the importance of the special NLP treatment of light verb constructions is emphasized in section 2. The precise identification of such constructions is crucial for the successful functioning of NLP applications, thus, it is argued that an algorithm is needed to automatically recognize them (section 4). In order to facilitate the development of such an algorithm, a corpus of light verb constructions has been built for Hungarian, which is presented together with statistical data in section 5. Finally, it is shown how NLP applications in the fields of machine translation and information extraction can profit from the implementation of an algorithm capable of identifying light verb constructions (section 6).

2 Light verb constructions in NLP

In natural language processing, one of the most challenging tasks is the proper treatment of col-

[1] There might be slight theoretical differences in the usage of these terms – e.g. semantically empty support verbs are called *light verbs* in e.g. Meyers et al. (2004a), that is, the term *support verb* is a hypernym of *light verb*. However, these differences are not analyzed in detail in this paper.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1110–1118,
Beijing, August 2010

locations, which term comprises light verb constructions as well. Every multiword expression is considered to be a collocation if its members often co-occur and its form is fixed to some extent (Siepmann, 2005; Siepmann, 2006; Sag et al., 2001; Oravecz et al., 2004; Váradi, 2006). Collocations are frequent in language use and they usually exhibit unique behaviour, thus, they often pose a problem to NLP systems.

Light verb constructions deserve special attention in NLP applications for several reasons. First, their meaning is not totally compositional, that is, it cannot be computed on the basis of the meanings of the parts of the collocation and the way they are related to each other. Thus, the result of translating the parts of the collocation can hardly be considered as the proper translation of the original expression. Second, light verb constructions (e.g. *make a mistake*) often share their syntactic pattern with other constructions such as literal verb + noun combinations (e.g. *make a cake*) or idioms (e.g. *make a meal*), thus, their identification cannot be based on solely syntactic patterns. Third, since the syntactic and the semantic head of the construction are not the same – the syntactic head being the verb and the semantic head being the noun –, they require special treatment when parsing. It can be argued that they form a complex verb similarly to phrasal or prepositional verbs (as reflected in the term complex verb structures). Thus, it is advisable to indicate their special syntacto-semantic relationship: in dependency grammars, the new role QUASI-ARGUMENT might be proposed for this purpose.

3 Related work

Light verb constructions – as a subtype of multiword expressions – have been paid special attention in NLP literature. Sag et al. (2001) classify them as a subtype of lexicalized phrases and flexible expressions. They are usually distinguished from productive or literal verb + noun constructions on the one hand and idiomatic verb + noun expressions on the other hand: e.g. Fazly and Stevenson (2007) use statistical measures in order to classify subtypes of verb + noun combinations and Diab and Bhutada (2009) developed a chunking method for classifying multiword expressions.

Identifying multiword expressions in general and light verb constructions in particular is not unequivocal since constructions with similar syntactic structure (e.g. verb + noun combinations) can belong to different subclasses on the productivity scale (i.e. productive combinations, light verb constructions and idioms). That is why well-designed and tagged corpora of multiword expressions are invaluable resources for training and testing algorithms that are able to identify multiword expressions. For instance, Grégoire (2007) describes the design and implementation of a lexicon of Dutch multiword expressions. Focusing on multiword verbs, Kaalep and Muischnek (2006; 2008) present an Estonian database and a corpus and Krenn (2008) describes a database of German PP-verb combinations. The Prague Dependency Treebank also contains annotation for light verb constructions (Cinková and Kolářová, 2005) and NomBank (Meyers et al., 2004b) provides the argument structure of common nouns, paying attention to those occurring in support verb constructions as well. On the other hand, Zarrieß and Kuhn (2009) make use of translational correspondences when identifying multiword expressions (among them, light verb constructions). A further example of corpus-based identification of light verb constructions in English is described in Tan et al. (2006).

Light verb constructions are considered to be semi-productive, that is, certain verbs tend to co-occur with nouns belonging to a given semantic class. A statistical method is applied to measure the acceptability of possible light verb constructions in Stevenson et al. (2004), which correlates reasonably well with human judgments.

4 Identifying light verb constructions

A database of light verb constructions and an annotated corpus might be of great help in the automatic recognition of light verb constructions. They can serve as a training database when implementing an algorithm for identifying those constructions.

The recognition of light verb constructions cannot be solely based on syntactic patterns for other (productive or idiomatic) combinations may exhibit the same verb + noun scheme (see section

2). However, in agglutinative languages such as Hungarian, nouns can have several grammatical cases, some of which typically occur in a light verb construction when paired with a certain verb. For instance, the verb *hoz* 'bring' is a transitive verb, that is, it usually occurs with a noun in the accusative case. On the other hand, when it is preceded or followed by a noun in the sublative or illative case (the typical position of the noun in Hungarian light verb constructions being right before or after the verb[2]), it is most likely a light verb construction. To illustrate this, we offer some examples:

> *vizet hoz*
>
> water-ACC bring
>
> 'to bring some water'

> *zavarba hoz*
>
> trouble-ILL bring
>
> 'to embarrass'

The first one is a productive combination (with the noun being in the accusative form) while the second one is a light verb construction. Note that the light verb construction also has got an argument in the accusative case (syntactically speaking, a direct object complement) as in:

> *Ez a megjegyzés mindenkit zavarba hozott.*
>
> this the remark everyone-ACC trouble-ILL bring-PAST-3SG
>
> 'This remark embarrassed everybody.'

Thus, the presence of an argument in the accusative does not imply that the noun + verb combination is a light verb construction. On the other hand, the presence of a noun in the illative or sublative case immediately preceding or following the verb strongly suggests that a light verb instance of *hoz* is under investigation.

Most light verb constructions have a verbal counterpart derived from the same stem as the noun, which entails that it is mostly deverbal nouns that occur in light verb constructions (as in *make/take a **decision*** compared to ***decide*** or ***döntést** hoz* vs. ***dönt*** in Hungarian). The identification of such nouns is possible with the help of a morphosyntactic parser that is able to treat derivation as well (e.g. hunmorph for Hungarian (Trón et al., 2005)), and the combination of a possible light verb and a deverbal noun typically results in a light verb construction.

Thus, an algorithm that makes use of morphosyntactic and derivational information and previously given lists can be constructed to identify light verb constructions in texts. It is important that the identification of light verb constructions precedes syntactic parsing, for the noun and the verb in the construction form one complex predicate, which has its effects on parsing: other arguments belong not solely to the verb but to the complex predicate.

To the best of our knowledge, there are no corpora of light verb constructions available for Hungarian. That is why we decided to build such a corpus. The corpus is described in detail in section 5. On the basis of the corpus developed, we plan to design an algorithm to automatically identify light verb constructions in Hungarian.

5 The corpus

In order to facilitate the extraction and the NLP treatment of Hungarian light verb constructions, we decided to build a corpus in which light verb constructions are annotated. The Szeged Treebank (Csendes et al., 2005) – a database in which words are morphosyntactically tagged and sentences are syntactically parsed – constitutes the basis for the annotation. We first selected the subcorpora containing business news, newspaper texts and legal texts for annotation since light verb constructions are considered to frequently occur in these domains (see B. Kovács (1999)). However, we plan to extend the annotation to other subcorpora as well (e.g. literary texts) in a later phase. Statistical data on the annotated subcorpora can be seen in Table 1.

5.1 Types of light verb constructions

As Hungarian is an agglutinative language, light verb constructions may occur in various forms.

[2] In a neutral sentence, the noun is right before the verb, in a sentence containing focus, it is right after the verb.

	sentences	words
business news	9574	186030
newspapers	10210	182172
legal texts	9278	220069
total	29062	582871

Table 1: Number of sentences and words in the annotated subcorpora

For instance, the verbal component may be inflected for tense, mood, person, number, etc. However, these inflectional differences can be easily resolved by a lemmatizer. On the other hand, besides the prototypical noun + verb combination, light verb constructions may be present in different syntactic structures, that is, in participles and infinitives and they can also undergo nominalization. These types are all annotated in the corpus texts since they also occur relatively frequently (see statistical data in 5.3). All annotated types are illustrated below.

- **Noun + verb combination <verb>**

  bejelentést tesz

  announcement-ACC makes

  'to make an announcement'

- **Participles <part>**

  – Present participle
    életbe lépő (intézkedés)
    life-ILL stepping (instruction)
    '(an instruction) taking effect'

  – Past participle
    csődbe ment (cég)
    bankrupt-ILL gone (firm)
    '(a firm) that went bankrupt'

  – Future participle
    fontolóra veendő (ajánlat)
    consideration-SUB to be taken (offer)
    '(an offer) that is to be taken into consideration'

  – Infinitive
    forgalomba hozni
    circulation-ILL bring-INF
    'to put into circulation'

- **Nominalization <nom>**

  bérbe vétel

  rent-ILL taking

  'hiring'

Split light verb constructions, where the noun and the verb are not adjacent, are also annotated and tagged. In this way, their identification becomes possible and the database can be used for training an algorithm that automatically recognizes (split) light verb constructions.

5.2 Annotation principles

Corpus texts contain single annotation, i.e. one annotator worked on each text. Light verb constructions can be found in between XML tags <FX> </FX>. In order to decide whether a noun + verb combination is a light verb construction or not, annotators were suggested to make use of a test battery developed for identifying Hungarian light verb constructions (Vincze, 2008).

The annotation process was carried out manually on the syntactically annotated version of the Szeged Treebank, thus, phrase boundaries were also taken into consideration when marking light verb constructions. Since the outmost boundary of the nominal component was considered to be part of the light verb construction, in several cases adjectives and other modifiers of the nominal head are also included in the construction, e.g.:

> *<FX>**nyilvános** ajánlatot tesz</FX>*
>
> public offer-ACC make
>
> 'to make a public offer'

In the case of participles, NP arguments may be also included (although in English, the same argument is expressed by a PP):

> *<FX>**Nyíregyházán** tartott ülésén</FX>*
>
> Nyíregyháza-SUP hold-PPT session-3SGPOSS-SUP
>
> 'at its session held in Nyíregyháza'

Constructions with a nominal component in the accusative case can be nominalized in two ways in Hungarian, as in:

szerződést köt

contract-ACC bind

’to make a contract’

<FX>szerződéskötés</FX>

contract+bind-GERUND

’making a contract’

<FX>adásvételi szerződések megkötése</FX>

sale contract-PL PREVERB-bind-GERUND-3SGPOSS

’making of sales contracts’

Both types are annotated in the corpus.

Besides the prototypical occurrences of light verb constructions (i.e. a bare common noun + verb[3]), other instances were also annotated in the corpus. For instance, the noun might be accompanied by an article or a modifier (recall that phrase boundaries were considered during annotation) or – for word order requirements – the noun follows the verb as in:

Ő hozta a jó döntést.

he bring-PAST-3SG-OBJ the good decision-ACC

’It was him who made the good decision.’

For the above reasons, a single light verb construction manifests in several different forms in the corpus. However, each occurrence is manually paired with its prototypical (i.e. bare noun + verb) form in a separate list, which is available at the corpus website.

5.3 Statistics on corpus data

The database contains 3826 occurrences of 658 light verb constructions altogether in 29062 sentences. Thus, a specific light verb construction occurs 5.8 times in the corpus on average. However, the participle form *irányadó* occurs in 607 instances (e.g. in *irányadó kamat* ’prime rate’) due to the topic of the business news subcorpus, which may distort the percentage rates. For this reason, statistical data in Table 2 are shown the occurrences of *irányadó* excluded.

	verb	part	nom	split	total
business news	565 58.6%	270 28%	90 9.3%	40 4.1%	965 25.2%
news- papers	458 59.3%	192 24.9%	55 7.1%	67 8.7%	772 20.2%
legal texts	640 30.7%	504 24.1%	709 33.9%	236 11.3%	2089 54.6%
total	1663 43.5%	966 25.2%	854 22.3%	236 9%	3826 100%

Table 2: Subtypes of light verb constructions in the corpus

It is revealed that although it is verbal occurrences that are most frequent, the percentage rate of participles is also relatively high. The number of nominalized or split constructions is considerably lower (except for the law subcorpus, where their number is quite high), however, those together with participles are responsible for about 55% of the data, which indicates the importance of their being annotated as well.

As for the general frequency of light verb constructions in texts, we compared the number of verb + argument relations found in the Szeged Dependency Treebank (Vincze et al., 2010) where the argument was a common noun to that of light verb constructions. It has turned out that about 13% of verb + argument relations consist of light verb constructions. This again emphasizes that they should be paid attention to, especially in the legal domain (where this rate is as high as 36.8%). Statistical data are shown in Table 3.

	V + argument	LVC
business news	9524	624 (6.6%)
newspapers	3637	539 (14.8%)
legal texts	2143	889 (36.8%)
total	15574	2052 (13.2%)

Table 3: Verb + argument relations and light verb constructions

The corpus is publicly available for re-

[3] As opposed to other languages where prototypical light verb constructions consist of a verb + a noun in accusative or a verb + a prepositional phrase (see e.g. Krenn (2008)), in Hungarian, postpositional phrases rarely occur within a light verb construction. However, annotators were told to annotate such cases as well.

search and/or educational purposes at www.inf.u-szeged.hu/rgai/nlp.

6 The usability of the corpus

As emphasized earlier, the proper treatment of light verb constructions is of primary importance in NLP applications. In order to achieve this, their identification is essential. The corpus created can function as the training database for the implementation of an algorithm capable of recognizing light verb constructions, which we plan to develop in the near future. In the following, the ways machine translation and information extraction can profit from such a corpus and algorithm are shortly presented.

6.1 Light verb constructions and machine translation

When translating collocations, translation programs face two main problems. On the one hand, parts of the collocation do not always occur next to each other in the sentence (split collocations). In this case, the computer must first recognize that the parts of the collocation form one unit (Oravecz et al., 2004), for which the multiword context of the given word must be considered. On the other hand, the lack (or lower degree) of compositionality blocks the possibility of word-by-word translation (Siepmann, 2005; Siepmann, 2006). However, a (more or less) compositional account of light verb constructions is required for successful translation (Dura and Gawrońska, 2005).

To overcome these problems, a reliable method is needed to assure that the nominal and verbal parts of the construction be matched. This requires an algorithm that can identify light verb constructions. In our corpus, split light verb constructions are also annotated, thus, it is possible to train the algorithm to recognize them as well: the problem of split collocations can be eliminated in this way.

A comprehensive list of light verb constructions can enhance the quality of machine translation – if such lists are available for both the source and the target language. Annotated corpora (especially and most desirably, parallel corpora) and explanatory-combinatorial dictionaries[4] are possible sources of such lists. Since in foreign language equivalents of light verb constructions, the nominal components are usually literal translations of each other (Vincze, 2009), by collating the corresponding noun entries in these lists the foreign language variant of the given light verb construction can easily be found. On the other hand, in order to improve the building of such lists, we plan to annotate light verb constructions in a subcorpus of SzegedParalell, a Hungarian-English manually aligned parallel corpus (Tóth et al., 2008).

6.2 Light verb constructions and information extraction

Information extraction (IE) seeks to process large amounts of unstructured text, in other words, to collect relevant items of information and to classify them. Even though humans usually overperform computers in complex information processing tasks, computers also have some obvious advantages due to their capacity of processing and their precision in performing well-defined tasks.

For several IE applications (e.g. relationship extraction) it is essential to identify phrases in a clause and to determine their grammatical role (subject, object, verb) as well. This can be carried out by a syntactic parser and is a relatively simple task. However, the identification of the syntactic status of the nominal component is more complex in the case of light verb constructions for it is a quasi-argument of the verb not to be confused with other arguments (Alonso Ramos, 1998). Thus, the parser should recognize the special status of the quasi-argument and treat it in a specific way as in the following sentences, one of which contains a light verb construction while the other one a verbal counterpart of the construction:

Pete ***made a decision*** *on his future.*

Pete ***decided*** *on his future.*

[4] Explanatory combinatorial dictionaries are essential for relation descriptions (up to the present, only fractions of the dictionary have been completed for Russian (Mel'čuk and Žolkovskij, 1984) and for French (see Mel'čuk et al. (1984 1999)), besides, trial entries have been written in Polish, English and German that contain the relations of a certain lexical unit to other lexemes given by means of lexical functions (see e.g. Mel'čuk et al. (1995)). These dictionaries indicate light verb constructions within the entry of the nominal component.

In the sentence with the verbal counterpart, the event of deciding involves two arguments: *he* and *his future*. In the sentence with the light verb construction, the same arguments can be found, however, it is unresolved whether they are the arguments of the verb (*made*) or the nominal component (*decision*). If a precise syntactic analysis is needed, it is crucial to know which argument belongs to which governor. Nevertheless, it is still debated if syntactic arguments should be divided between the nominal component and the verb (see Meyers et al. (2004a) on argument sharing) and if yes, how (Alonso Ramos, 2007).

For the purpose of information extraction, such a detailed analysis is unnecessary and in general terms, the nominal component can be seen as part of the verb, that is, they form a complex verb similarly to phrasal or prepositional verbs and this complex verb is considered to be the governor of arguments. Thus, the following data can be yielded by the IE algorithm: there is an event of **decision-making**, **Pete** is its subject and it is about **his future** (and not an event of **making** with the arguments **decision**, **Pete** and **his future**). Again, the precise identification of light verb constructions can highly improve the performance of parsers in recognizing relations between the complex verb and its arguments.

7 Conclusion

In this paper, we have presented the development of a corpus of Hungarian light verb constructions. Basic annotation guidelines and statistical data have also been included. The annotated corpus can serve as a training database for implementing an algorithm that aims at identifying light verb constructions. Several NLP applications in the fields of e.g. machine translation and information extraction may profit from the successful integration of such an algorithm into the system, which we plan to develop in the near future.

Acknowledgements

This work was supported in part by the National Office for Research and Technology of the Hungarian government within the framework of the project MASZEKER.

The authors wish to thank György Szarvas for his help in developing the annotation tool and Richárd Farkas for his valuable comments on an earlier draft of this paper.

References

Alonso Ramos, Margarita. 1998. *Etude sémantico-syntaxique des constructions à verbe support*. Ph.D. thesis, Université de Montréal, Montreal, Canada.

Alonso Ramos, Margarita. 2007. Towards the Synthesis of Support Verb Constructions. In Wanner, Leo, editor, *Selected Lexical and Grammatical Issues in the Meaning-Text Theory. In Honour of Igor Mel'čuk*, pages 97–138, Amsterdam / Philadelphia. Benjamins.

B. Kovács, Mária. 1999. A funkcióigés szerkezetek a jogi szaknyelvben [Light verb constructions in the legal terminology]. *Magyar Nyelvőr*, 123(4):388–394.

Cinková, Silvie and Veronika Kolářová. 2005. Nouns as Components of Support Verb Constructions in the Prague Dependency Treebank. In Šimková, Mária, editor, *Insight into Slovak and Czech Corpus Linguistics*, pages 113–139. Veda Bratislava, Slovakia.

Csendes, Dóra, János Csirik, Tibor Gyimóthy, and András Kocsor. 2005. The Szeged TreeBank. In Matousek, Václav, Pavel Mautner, and Tomás Pavelka, editors, *Proceedings of the 8th International Conference on Text, Speech and Dialogue, TSD 2005*, Lecture Notes in Computer Science, pages 123–132, Berlin / Heidelberg, September. Springer.

Diab, Mona and Pravin Bhutada. 2009. Verb Noun Construction MWE Token Classification. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, pages 17–22, Singapore, August. Association for Computational Linguistics.

Dura, Elżbieta and Barbara Gawrońska. 2005. Towards Automatic Translation of Support Verbs Constructions: the Case of Polish robic/zrobic and Swedish göra. In *Proceedings of the 2nd Language & Technology Conference*, pages 450–454, Poznań, Poland, April. Wydawnictwo Poznańskie Sp. z o.o.

Fazly, Afsaneh and Suzanne Stevenson. 2007. Distinguishing Subtypes of Multiword Expressions Using Linguistically-Motivated Statistical Measures. In *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions*, pages 9–16, Prague, Czech Republic, June. Association for Computational Linguistics.

Grégoire, Nicole. 2007. Design and Implementation of a Lexicon of Dutch Multiword Expressions. In *Proceedings of the Workshop on A Broader Perspective on Multiword Expressions*, pages 17–24, Prague, Czech Republic, June. Association for Computational Linguistics.

Kaalep, Heiki-Jaan and Kadri Muischnek. 2006. Multi-Word Verbs in a Flective Language: The Case of Estonian. In *Proceedings of the EACL Workshop on Multi-Word Expressions in a Multilingual Contexts*, pages 57–64, Trento, Italy, April. Association for Computational Linguistics.

Kaalep, Heiki-Jaan and Kadri Muischnek. 2008. Multi-Word Verbs of Estonian: a Database and a Corpus. In *Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions (MWE 2008)*, pages 23–26, Marrakech, Morocco, June.

Krenn, Brigitte. 2008. Description of Evaluation Resource – German PP-verb data. In *Proceedings of the LREC Workshop Towards a Shared Task for Multiword Expressions (MWE 2008)*, pages 7–10, Marrakech, Morocco, June.

Langer, Stefan. 2004. A Linguistic Test Battery for Support Verb Constructions. *Lingvisticae Investigationes*, 27(2):171–184.

Mel'čuk, Igor and Aleksander Žolkovskij. 1984. *Explanatory Combinatorial Dictionary of Modern Russian*. Wiener Slawistischer Almanach, Vienna, Austria.

Mel'čuk, Igor, André Clas, and Alain Polguère. 1995. *Introduction à lexicologie explicative et combinatoire*. Duculot, Louvain-la-Neuve, France.

Mel'čuk, Igor, et al. 1984–1999. *Dictionnaire explicatif et combinatoire du français contemporain: Recherches lexico-sémantiques I–IV.* Presses de l'Université de Montréal, Montreal, Canada.

Meyers, Adam, Ruth Reeves, and Catherine Macleod. 2004a. NP-External Arguments: A Study of Argument Sharing in English. In Tanaka, Takaaki, Aline Villavicencio, Francis Bond, and Anna Korhonen, editors, *Second ACL Workshop on Multiword Expressions: Integrating Processing*, pages 96–103, Barcelona, Spain, July. Association for Computational Linguistics.

Meyers, Adam, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004b. The NomBank Project: An Interim Report. In Meyers, Adam, editor, *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation*, pages 24–31, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Oravecz, Csaba, Károly Varasdi, and Viktor Nagy. 2004. Többszavas kifejezések számítógépes kezelése [The treatment of multiword expressions in computational linguistics]. In Alexin, Zoltán and Dóra Csendes, editors, *MSzNy 2004 – II. Magyar Számítógépes Nyelvészeti Konferencia*, pages 141–154, Szeged, Hungary, December. University of Szeged.

Sag, Ivan A., Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2001. Multiword Expressions: A Pain in the Neck for NLP. In *Proceedings of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002*, pages 1–15, Mexico City, Mexico.

Siepmann, Dirk. 2005. Collocation, colligation and encoding dictionaries. Part I: Lexicological Aspects. *International Journal of Lexicography*, 18(4):409–444.

Siepmann, Dirk. 2006. Collocation, colligation and encoding dictionaries. Part II: Lexicographical Aspects. *International Journal of Lexicography*, 19(1):1–39.

Stevenson, Suzanne, Afsaneh Fazly, and Ryan North. 2004. Statistical Measures of the Semi-Productivity of Light Verb Constructions. In Tanaka, Takaaki, Aline Villavicencio, Francis Bond, and Anna Korhonen, editors, *Second ACL Workshop on Multiword Expressions: Integrating Processing*, pages 1–8, Barcelona, Spain, July. Association for Computational Linguistics.

Tan, Yee Fan, Min-Yen Kan, and Hang Cui. 2006. Extending corpus-based identification of light verb constructions using a supervised learning framework. In *Proceedings of the EACL Workshop on Multi-Word Expressions in a Multilingual Contexts*, pages 49–56, Trento, Italy, April. Association for Computational Linguistics.

Tóth, Krisztina, Richárd Farkas, and András Kocsor. 2008. Hybrid algorithm for sentence alignment of Hungarian-English parallel corpora. *Acta Cybernetica*, 18(3):463–478.

Trón, Viktor, György Gyepesi, Péter Halácsy, András Kornai, László Németh, and Dániel Varga. 2005. hunmorph: Open Source Word Analysis. In *Proceedings of the ACL Workshop on Software*, pages 77–85, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Váradi, Tamás. 2006. Multiword Units in an MT Lexicon. In *Proceedings of the EACL Workshop on Multi-Word Expressions in a Multilingual Contexts*, pages 73–78, Trento, Italy, April. Association for Computational Linguistics.

Vincze, Veronika, Dóra Szauter, Attila Almási, György Móra, Zoltán Alexin, and János Csirik. 2010. Hungarian Dependency Treebank. In Calzolari, Nicoletta, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

Vincze, Veronika. 2008. A puszta köznév + ige komplexumok státusáról [On the status of bare common noun + verb constructions]. In Sinkovics, Balázs, editor, *LingDok 7. Nyelvész-doktoranduszok dolgozatai*, pages 265–283, Szeged, Hungary. University of Szeged.

Vincze, Veronika. 2009. Főnév + ige szerkezetek a szótárban [Noun + verb constructions in the dictionary]. In Váradi, Tamás, editor, *III. Alkalmazott Nyelvészeti Doktorandusz Konferencia*, pages 180–188, Budapest. MTA Nyelvtudományi Intézet.

Zarrieß, Sina and Jonas Kuhn. 2009. Exploiting Translational Correspondences for Pattern-Independent MWE Identification. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, pages 23–30, Singapore, August. Association for Computational Linguistics.

Syntax Based Reordering with Automatically Derived Rules for Improved Statistical Machine Translation

Karthik Visweswariah
IBM Research
v-karthik@in.ibm.com

Jiri Navratil
IBM Research
jiri@us.ibm.com

Jeffrey Sorensen
Google, Inc.
sorenj@google.com

Vijil Chenthamarakshan
IBM Research
vijil.e.c@in.ibm.com

Nanda Kambhatla
IBM Research
kambhatla@in.ibm.com

Abstract

Syntax based reordering has been shown to be an effective way of handling word order differences between source and target languages in Statistical Machine Translation (SMT) systems. We present a simple, automatic method to learn rules that reorder source sentences to more closely match the target language word order using only a source side parse tree and automatically generated alignments. The resulting rules are applied to source language inputs as a pre-processing step and demonstrate significant improvements in SMT systems across a variety of languages pairs including English to Hindi, English to Spanish and English to French as measured on a variety of internal test sets as well as a public test set.

1 Introduction

Different languages arrange words in different orders, whether due to grammatical constraints or other conventions. Dealing with these word order permutations is one of the fundamental challenges of machine translation. Given an exceptionally large training corpus, a phrase-based system can learn these reordering on a case by case basis. But, if our systems are to generalize to phrases not seen in the training data, they must explicitly capture and model these reorderings. However, permutations are difficult to model and impractical to search.

Presently, approaches that handle reorderings typically model word and phrase movements via a *distortion model* and rely on the target language model to produce words in the right order. Early distortion models simply penalized longer jumps more than shorter jumps (Koehn et al., 2003) independent of the source or target phrases in question. Other models (Tillman, 2004), (Al-Onaizan and Papineni, 2006) generalize this to include lexical dependencies on the source.

Another approach is to incorporate features, based on the target syntax, during modeling and decoding, and this is shown to be effective for various language pairs (Yamada and Knight, 2001), (Zollmann and Venugopal, 2006). Hierarchical phrase-based decoding (Chiang, 2005) also allows for long range reordering without explicitly modeling syntax. While these approaches have been shown to improve machine translation performance (Zollmann et al., 2008) they usually combine chart parsing with the decoding process, and are significantly more computationally intensive than phrase-based systems.

A third approach, one that has proved to be useful for phrase-based SMT systems, is to reorder each source-side sentence using a set of rules applied to a parse tree of the source sentence. The goal of these rules is to make the word order of the source sentence more similar to the expected target sentence word order. With this approach, the reordering rules are applied before training and testing with an SMT system. The efficacy of these methods has been shown on various language pairs including: French to English (Xia and McCord, 2004), German to English (Collins et al., 2005), English to

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1119–1127,
Beijing, August 2010

Chinese, (Wang et al., 2007) and Hindi to English (Ramanathan et al., 2008).

In this paper, we propose a simple model for reordering conditioned on the source side parse tree. The model is learned using a parallel corpus of source-target sentence pairs, machine generated word alignments, and source side parses. We apply the reordering model to both training and test data, for four different language pairs: English → Spanish, English → French, English → Hindi, and English → German. We show improvements in machine translation performance for all of the language pairs we consider except for English → German. We use this negative result to propose extensions to our reordering model. We note that the syntax based reordering we propose can be combined with other approaches to handling reordering and does not have to be followed by an assumption of monotonicity. In fact, our phrase-based model, trained upon reordered data, retains its reordering models and search, but we expect that these facilities are employed much more sparingly with reordered inputs.

2 Related work

There is a significant quantity of work in syntax based reordering employed to improve machine translation systems. We summarize our contributions to be:

- Learning the reordering rules based on training data (without relying on linguistic knowledge of the language pair)
- Requiring only source side parse trees
- Experimental results showing the efficacy for multiple language pairs
- Using a lexicalized distortion model for our baseline decoder

There have been several studies that have demonstrated improvements with syntax based reordering based upon hand-written rules. There have also been studies investigating the sources of these improvements (Zwarts and Dras, 2007). Hand-written rules depend upon expert knowledge of the linguistic properties of the particular language pair. Initial efforts (Niessen and Ney, 2001) were made at improving German-English translation by handling two phenomena: question inversion and detachable verb prefixes in German. In (Collins et al., 2005), (Wang et al., 2007), (Ramanathan et al., 2008), (Badr et al., 2009) rules are developed for translation from German to English, Chinese to English, English to Hindi, and English to Arabic respectively. (Xu et al., 2009) develop reordering rules based upon a linguistic analysis of English and Korean sentences and then apply those rules to translation from English into Korean and four other languages: Japanese, Hindi, Urdu and Turkish. Unlike this body of work, we automatically learn the rules from the training data and show efficacy on multiple language pairs.

There have been some studies that try to learn rules from the data. (Habash, 2007) learns reordering rules based on a dependency parse and they report a negative result for Arabic to English translation. (Zhang et al., 2007) learn reordering rules on chunks and part of speech tags, but the rules they learn are not hierarchical and would require large amounts of training data to learn rules for long sentences. Additionally, we only keep a single best reordering (instead of a lattice with possible reorderings) which makes the decoding significantly more efficient. (Xia and McCord, 2004) uses source and target side parse trees to automatically learn rules to reorder French sentences to match English order. The requirement to have both source and target side parse trees makes this method inapplicable to any language that does not have adequate tree bank resources. In addition, this work reports results using monotone decoding, since their experiments using non-monotone decoding without a distortion model were actually worse.

3 Reordering issues in specific languages

In this section we discuss the reordering issues typical of translating between English and Hindi, French, Spanish and German which are the four language pairs we experiment on in this paper.

3.1 Spanish and French

Typical word ordering patterns common to these two European languages relate to noun phrases including groups of nouns and adjectives. In contrast to English, French and Spanish adjectives and adjunct nouns follow the main noun, i.e. we typically observe a reversal of word order in noun phrases, e.g., *"A beautiful red car"* translates into French as *"Une voiture rouge beau"*, and as *"Un coche rojo bonito"* into Spanish. Phrase-based MT systems are capable of capturing these patterns provided they occur with sufficient frequency for each example in the training data. For rare noun phrases, however, the MT may produce erroneous word order that can lead to serious distortions in the meaning. Particularly difficult are nominal phrases from specialized domains that involve challenging terminology, for example: "*group reference attribute*" and "*validation checking code*". In both instances, the baseline MT system generated translations with an incorrect word order and, consequently, possibly a different meaning. We will return to these two examples in Section 5.1 to compare the output of a MT system with and without reordering.

3.2 German

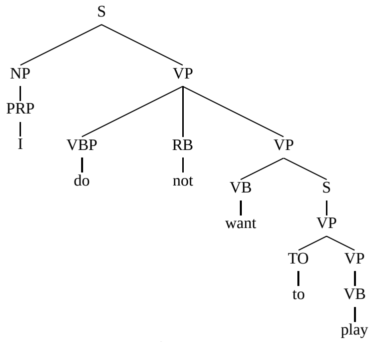
Unlike French and Spanish, German poses a considerably different challenge with respect to word ordering. The most frequent reordering in German relates to verbs, particularly verb groups consisting of auxiliary and main verbs, as well as verbs in relative clauses. Moreover, reordering patterns between German and English tend to span large portions of the sentence. We included German in our investigations to determine whether our automated rule extraction procedure can capture such long distance patterns.

3.3 Hindi

Hindi word order is significantly different than English word order; the typical order followed is Subject Object Verb (although Object Subject Verb order can be used if nouns are followed by appropriate case markers). This is in contrast to English which has a Subject Verb Object order. This can result in words that are close in English moving arbitrarily far apart in Hindi depending on the length of the noun phrase representing the object and the length of the verb phrase. These long range reorderings are generally hard for a phrase based system to capture. Another way Hindi and English differ is that prepositions in English become postpositions in Hindi and appear after the noun phrase. Again, this reordering can lead to long distance movements of words. We include Hindi in our investigation since it has significantly different structure as compared to English.

4 Learning reordering rules

In this section we describe how we learn rules that transform source parse trees so the leaf word order is more like the target language. We restrict ourselves to reorderings that can be obtained by permuting child nodes at various interior nodes in a parse tree. With many reordering phenomena discussed in Section 3 this is a fairly strong assumption about pairs of languages, and there are examples in English→Hindi where such an assumption will not allow us to generate the right reordering. As an example consider the English sentence "*I do not want to play*". The sentence has a parse:



The correct word order of the translation in Hindi is "*I to play not want*" In this case, the word *not* breaks up the verb phrase *want to play* and hence the right Hindi word order cannot be obtained by the reordering allowed by our model. We found such examples to be rare in English→Hindi, and we impose this restriction for the simplicity of the model. Experimental results on several languages show benefits of reordering in spite of this simplifying assumption.

Consider a source sentence s and its corresponding constituency parse tree S[1]. We set up the problem in a probabilistic framework, i.e. we would like to build a probabilistic model $P(T|S)$ that assigns probabilities to trees such that the

[1] In this paper we work with constituency parse trees. Initial experiments, applying similar techniques to dependency parse trees did not yield improvements.

word order in trees T which are assigned higher probability match the order of words in the target language. A parse tree, S is a set of nodes. Interior nodes have an ordered list of children. Leaf nodes in the tree are the words in the sentence s, and interior nodes are labeled by the linguistic constituent that they represent. Each word has a parent node (with only one child) labeled by the part-of-speech tag of the word.

Our model assigns non-zero probabilities to trees that can be obtained by permuting the child nodes at various interior nodes of the tree S. We assume that children of a node are ordered independently of all other nodes in the tree. Thus

$$P(T|S) = \prod_{n \in I(S)} P(\pi(c_n)|S, n, c_n),$$

where $I(S)$ is the set of interior nodes in the tree S, c_n is the list of children of node n and π is a permutation. We further assume that the reordering at a particular node is dependent only on the labels of its children:

$$P(T|S) = \prod_{n \in I(S)} P(\pi(c_n)|c_n).$$

We parameterize our model using a log-linear model:

$$P(\pi(c_n)|c_n) = \frac{1}{Z(c_n)} \exp(\lambda^T f(\pi, c_n)). \quad (1)$$

We choose the simplest possible set of feature functions: for each observed sequence of non-terminals we have one boolean feature per permutation of the sequence of non-terminals, with the feature firing iff that particular sequence is observed. Assuming, we have a training corpus C of (T, S) tree pairs, we could optimize the parameters of our model to maximize : $\prod_{S \in C} P(T|S)$. With the simple choice of feature functions described above, this amounts to:

$$P(\pi(c_n)|c_n) = \frac{\mathrm{count}(\pi(c_n))}{\mathrm{count}(c_n)},$$

where $\mathrm{count}(c_n)$ is the number of times the sequences of nodes c_n is observed in the training data and $\mathrm{count}(\pi(c_n))$ is the number of times that c_n in S is permuted to $\pi(c_n)$ in T. In Section 6, we show considering more general feature functions and relaxing some of the independence might yield improvements on certain language pairs.

For each source sentence s with parse S we find the tree T that makes the given alignment for that sentence pair most monotone. For each node n in the source tree S let D_n be the set of words that are descendants of n. Let us denote by $tpos(n)$ the average position of words in the target sentence that are aligned to words in D_n. Then

$$tpos(n) = \frac{1}{|D_n|} \sum_{w \in D_n} a(w),$$

where $a(w)$ is the index of the word on the target side that w is aligned with. If a word w is not aligned to any target word, we leave it out from the mean position calculation above. If a word w is aligned to many words we let $a(w)$ be the mean position of the words that w is aligned to. For each node n in the tree we transform the tree by sorting the list of children of n according to $tpos$. The pairs of parse trees that we obtain (S, T) in this manner form our training corpus to estimate our parameters.

In using our model, we once again go for the simplest choice, we simply reorder the source side sentences by choosing $\arg\max_T P(T|S)$ both in training and in testing; this amounts to reordering each interior node based on the most frequent reordering of the constituents seen in training. To reduce the effect of noise in training alignments we apply the reordering, only if we have seen the constituent sequence often enough in our training data (a *count threshold* parameter) and if the most frequent reordering is sufficiently more frequent than the next most frequent reordering (a *significance threshold*).

5 Experiments

5.1 Results for French, Spanish, and German

In each language, the rule extraction was performed using approximately 1.2M sentence pairs aligned using a maxent aligner (Ittycheriah and Roukos, 2005) trained using a variety of domains (Europarl, computer manuals) and a maximum entropy parser for English (Ratnaparkhi, 1999). With a significance threshold of 1.2, we obtain about 1000 rules in the eventual reordering process.

Phrase-based systems were trained for each language pair using 11M sentence pairs spanning a variety of publicly available (e.g. Europarl, UN speeches) and internal corpora (IT technical and news domains). The system phrase blocks were extracted based on a union of HMM and max-ent alignments with corpus-selective count pruning. The lexicalized distortion model was used as described in (Al-Onaizan and Papineni, 2006) with a window width of up to 5 and a maximum number of skipped (not covered) words during decoding of 2. The distortion model assigns a probability to a particular word to be observed with a specific jump. The decoder uses a 5-gram interpolated language model spanning the various domains mentioned above. The baseline system without reordering and a system with reordering was trained and evaluated in contrastive experiments. The evaluation was performed utilizing the following (single-reference) test sets:

- *News*: 541 sentences from the news domain.
- *TechA:* 600 sentences from a computer-related technical domain, this has been used as a dev set.
- *TechB:* 1038 sentences from a similar domain as *TechA* used as a blind test.
- *Dev09:* 1026 sentences defined as the *news-dev2009b* development set of the Workshop on Statistical Machine Translation 2009 [2]. This set provides a reference measurement using a public data set. Previously published results on this set can be found, for example, in (Popovic et al., 2009).

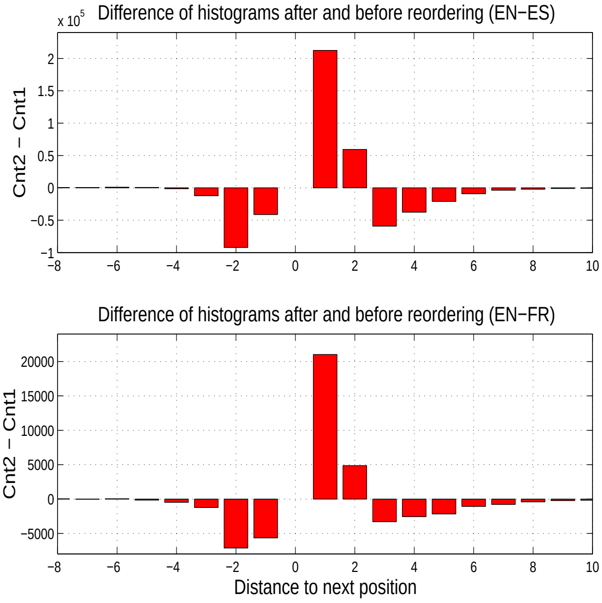


Figure 1: Difference-histogram of word order distortions for English→Spanish (upper), and English→French (lower).

In order to assess changes in word ordering patterns prior to and after an application of the reordering, we created histograms of word jumps in the alignments obtained in the baseline as well as in the reordered system. Given a source word s_i at index i and the target word t_j it is aligned to at index j, a jump of 1 would correspond to s_{i+1} aligning to target word t_{j+1}, while an alignment to t_{j-1} corresponds to a jump of -1, etc. A histogram over the jump values gives us a summary of word order distortion. If all of the jumps were one, then there is no reordering between the two languages. To gain insight into changes introduced by our reordering we look at differences of the two histograms i.e., counts after reordering minus counts before reordering. We would hope that after reordering most of the jumps are small and concentrated around one. Figure 1 shows such *difference*-histograms for the language pairs English→Spanish and English→French, respectively, on a sample of about 15k sentence pairs held out of the system training data. Here, a positive difference value indicates an increased number *after* reordering. In both cases a consistent trend toward monotonicity is observed, i.e more jumps of size one and two, and fewer large jumps. This confirms the intended reordering effect and indicates that the reordering rules extracted generalize well.

Table 1 shows the resulting uncased BLEU scores for English-Spanish and English-French.

In both cases the reordering has a consistent positive effect on the BLEU scores across test sets. In examining the sources of improvement, we noticed that word order in several noun phrases that

[2] http://statmt.org/wmt09/

	System	News	TechA	TechB	Dev09
Spanish	Baseline	0.3849	0.3371	0.3483	0.2244
	Reordered	0.4031	0.3582	0.3605	0.2320
French	Baseline	0.5140	0.2971	0.3035	0.2014
	Reordered	0.5242	0.3152	0.3154	0.2092
German	Baseline	0.2580	0.1582	0.1697	0.1281
	Reordered	0.2544	0.1606	0.1682	0.1271
Hindi	Baseline	20.0			
	Reordered	21.7			

Table 1: Uncased BLEU scores for phrase-based machine translation.

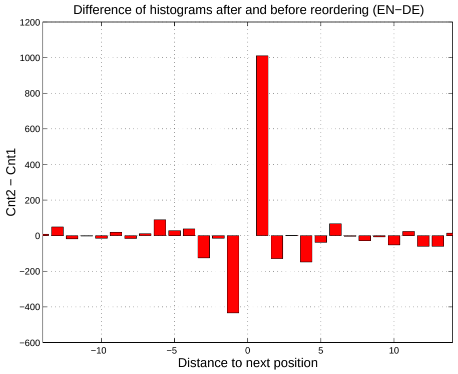


Figure 2: Difference-histogram of word order distortions for English→German.

were not common in the training data were fixed by use of the reordering rules.

Table 1 shows the BLEU scores for the English→German language pair, for which a mixed result is observed. The difference-histogram for English→German, shown in Figure 2, differs from those of the other languages with several increases in jumps of large magnitude, indicating failure of the extracted rules to generalize.

The failure of our simple method to gain consistent improvements comparable to Spanish and French, along with our preliminary finding that a relatively few manually crafted reordering rules (we describe these in Section 6.4) tend to outperform our method, leads us to believe that a more refined approach is needed in this case and will be subject of further discussion below.

5.2 Results for Hindi

Our Hindi-English experiments were run with an internal parallel corpus of roughly 250k sentence pairs (5.5M words) consisting of various domains (including news). To learn reordering rules we used HMM alignments and a maxent parser (Ratnaparkhi, 1999), with a count threshold of 100, and a significance threshold of 1.7 (these settings gave us roughly 200 rules). We also experimented with other values of these thresholds and found that the performance of our systems were not very sensitive to these thresholds. We trained Direct Translation Model 2 (DTM) systems (Ittycheriah and Roukos, 2007) with and without source reordering and evaluated on a test set of 357 sentences from the News domain. We note that the DTM baseline includes features (functions of target words and jump size) that allow it to model lexicalized reordering phenomena. The reordering window size was set to +/- 8 words for the baseline and system with reordered inputs. Table 1 shows the uncased BLEU scores for English-Hindi, showing a gain from using the reordering rules. For the reordered case, the HMM alignments are rederived, but the accuracy of these were no better than those of the unreordered input and experiments showed that the gains in performance were not due to the effect on the alignments.

Figure 3 shows *difference*-histograms for the language pair English→Hindi, on a sample of about 10k sentence pairs held out of the system training data. The histogram indicates that our reordering rules generalize and that the reordered English is far more monotonic with respect to the Hindi.

6 Analysis of errors and future directions

In this section, we analyze some of the sources of errors in reordering rules learned via our model, to better understand directions for further improvement.

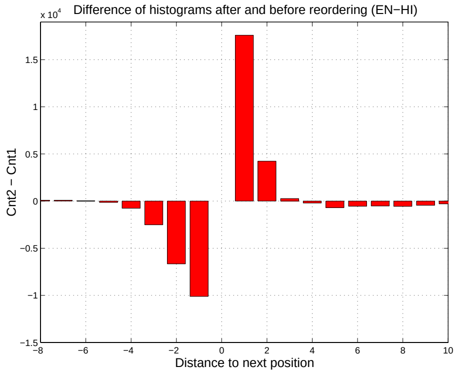


Figure 3: Difference-histogram of word order distortions for English→Hindi.

6.1 Model weakness

In our initial experiments, we noticed that for the most frequent reordering rules in English→Hindi (e.g that IN NP or NP PP flips in Hindi) the probability of a reordering was roughly 65%. This was concerning since it meant that on 35% of the data we would be making wrong reordering decisions by choosing the most likely reordering. To get a better feel for whether we needed a stronger model (e.g by lexicalization or by looking at larger context in the tree rather than just the children), we analyzed some of the cases in our training data where (IN,NP), (NP, PP) pairs were left unaltered in Hindi. In doing that analysis, we noticed examples involving negatives that our model does not currently handle. The first issue was mentioned in Section 4, where the assumption that we can achieve the right word order by reordering constituent phrases, is incorrect. The second issue is illustrated by the following sentences: *I have some/no books*, which have similar parse structures, the only difference being the determiner *some* vs the determiner *no*. In Hindi, the order of the fragments *some books* and the fragment *no books* are different (in the first case the words stay in order, in the second the flip). Handling this example would need our model to be lexicalized. These issue of negatives requiring special handling also came up in our analysis of German (Section 6.4). Other than the negatives (which require a lexicalized model), the major reason for the lack of sharpness of the reordering rule probability was alignment errors and parser issues. We look at these topics next.

Aligner	Number of Sentences	fMeasure	BLEU score
HMM	250k	62.4	21.7
MaxEnt	250k	76.6	21.4
Manual	5k	-	21.3

Table 2: Using different alignments

6.2 Alignment accuracy

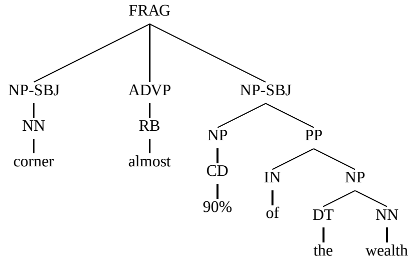
Since we rely on automatically generated alignments to learn the rules, low accuracy of the alignments could impact the quality of the rules learned. This is especially a concern for English→Hindi since the quality of HMM alignments are fairly low. To quantify this effect, we learn reordering rules using three sets of alignments: HMM alignments, alignments from a supervised MaxEnt aligner (Ittycheriah and Roukos, 2005), and hand alignments. Table 2 summarizes our results using aligners with differing alignment qualities for our English→Hindi task and shows that quality of alignments in learning the rules is not the driving factor in affecting rule quality.

6.3 Parser accuracy

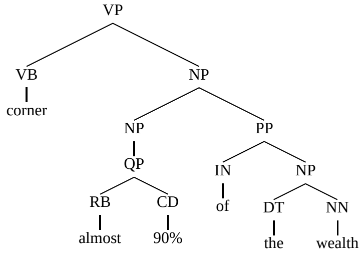
Accuracy of the parser in the source language is a key requirement for our reordering method, because we choose the single best reordering based on the most likely parse of the source sentence. This would especially be an issue in translating *from* languages other than English, where the parser would not be of quality comparable to the English parser.

In examining some of the errors in reordering we did observe a fair fraction attributable to issues in parsing, as seen in the example sentence: *The rich of this country , corner almost 90% of the wealth .*

The second half of the sentence is parsed by the Berkeley parser (Petrov et al., 2006) as:



and by IBM's maximum entropy parser parser (Ratnaparkhi, 1999) as:



With the first parse, we get the right Hindi order for the second part of the sentence which is: *the wealth of almost 90% corner* . To investigate the effect of choice of parser we compared using the Berkeley parser and the IBM parser for reordering, and we found the BLEU score essentially unchanged: 21.6 for the Berkeley parser and 21.7 for the IBM parser. A potential source of improvements might be to use alternative parses (via different parsers or n-best parses) to generate n-best reorderings both in training and at test.

6.4 Remarks on German reordering

Despite a common heritage, German word order is distinct from English, particularly regarding verb placement. This difference can be dramatic, if an auxiliary (e.g. modal) verb is used in conjunction with a full verb, or the sentence contains a subordinate clause. In addition to our experiments with automatically learned rules, a small set of handcrafted reordering rules was created and evaluated. Our preliminary results indicate that the latter rules tend to outperform the automatically derived ones by 0.5-1.0 BLEU points on average. These rules are summarized as follows:

1. In a VP immediately following an NP, move the negation particle to main verb.
2. Move a verb group away from a modal verb; to the end the of a VP. Negation also moves along with verb.
3. Move verb group to end of an embedded/relative clause.
4. In a VP following a subject, move negation to the end of VP (handling residual cases)

The above hand written rules show several weaknesses of our automatically learned rules for reordering. Since our model is not lexicalized, negations are not handled properly as they are tagged RB (along with other adverbs). Another limitation apparent from the first rule above (the movement of verbs in a verb phrase depends on the previous phrase being a noun phrase) is that the automatic reordering rule for a node's children depends only on the children of that node and not a larger context. For instance, a full verb following a modal verb is typically parsed as a VP child node of the modal VP node, hence the automatic rule, as currently considered, will not take the modal verb (being a sibling of the full-verb VP node) into account. We are currently investigating extensions of the automatic rule extraction alorithm to address these shortcomings.

6.5 Future directions

Based on our analysis of the errors and on the hand designed German rules we would like to extend our model with more general feature functions in Equation 1 by allowing features: that are dependent on the constituent words (or headwords), that examine a large context than just a nodes children (see the first German rule above) and that fire for all permutations when the constituent X is moved to the end (or start). This would allow us to generalize more easily to learn rules of the type "move X to the end of the phrase". Another direction that we feel should be explored, is the use of multiple parses to obtain multiple reorderings and combine these at a later stage.

7 Conclusions

In this paper we presented a simple method to automatically derive rules for reordering source sentences to make it look more like target language sentences. Experiments (on internal and public test sets) indicate performance gains for English→French, English→Spanish, and English→Hindi. For English→German we did not see improvements with automatically learned rules while a few hand designed rules did give improvements, which motivated a few directions to explore.

References

[Al-Onaizan and Papineni2006] Al-Onaizan, Yaser and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proceedings of ACL*.

[Badr et al.2009] Badr, Ibrahim, Rabih Zbib, and James Glass. 2009. Syntactic phrase reordering for english-to-arabic statistical machine translation. In *Proceedings of EACL*.

[Chiang2005] Chiang, David. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*.

[Collins et al.2005] Collins, Michael, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*.

[Habash2007] Habash, Nizar. 2007. Syntactic preprocessing for statistical machine translation. In *MT Summit*.

[Ittycheriah and Roukos2005] Ittycheriah, Abraham and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of HLT/EMNLP*.

[Ittycheriah and Roukos2007] Ittycheriah, Abraham and Salim Roukos. 2007. Direct translation model 2. In *Proceedings of HLT-NAACL*, pages 57–64.

[Koehn et al.2003] Koehn, Philipp, Franz Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*.

[Niessen and Ney2001] Niessen, Sonja and Hermann Ney. 2001. Morpho-syntactic analysis for reordering in statistical machine translation. In *Proc. MT Summit VIII*.

[Petrov et al.2006] Petrov, Slav, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *COLING-ACL*.

[Popovic et al.2009] Popovic, Maja, David Vilar, Daniel Stein, Evgeny Matusov, and Hermann Ney. 2009. The RWTH machine translation system for WMT 2009. In *Proceedings of WMT 2009*.

[Ramanathan et al.2008] Ramanathan, A., P. Bhattacharyya, J. Hegde, R. M. Shah, and M. Sasikumar. 2008. Simple syntactic and morphological processing can help english-hindi statistical machine translation. In *Proceedings of International Joint Conference on Natural Language Processing*.

[Ratnaparkhi1999] Ratnaparkhi, Adwait. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3).

[Tillman2004] Tillman, Christoph. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL*.

[Wang et al.2007] Wang, Chao, Michael Collins, and Philipp Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proceedings of EMNLP-CoNLL*.

[Xia and McCord2004] Xia, Fei and Michael McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *Proceedings of Coling*.

[Xu et al.2009] Xu, Peng, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve SMT for Subject-Object-Verb languages. In *Proceedings of NAACL-HLT*.

[Yamada and Knight2001] Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of ACL*.

[Zhang et al.2007] Zhang, Yuqi, Richard Zens, and Hermann Ney. 2007. Chunk-level reordering of source language sentences with automatically learned rules for statistical machine translation. In *NAACL-HLT AMTA Workshop on Syntax and Structure in Statistical Translation*.

[Zollmann and Venugopal2006] Zollmann, Andreas and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings on the Workshop on Statistical Machine Translation*.

[Zollmann et al.2008] Zollmann, Andreas, Ashish Venugopal, Franz Och, and Jay Ponte. 2008. A systematic comparison of phrase-based, hierarchical and syntax-augmented statistical MT. In *Proceedings of COLING*.

[Zwarts and Dras2007] Zwarts, Simon and Mark Dras. 2007. Syntax-based word reordering in phrase-based statistical machine translation: why does it work? In *Proc. MT Summit*.

Efficient Statement Identification for Automatic Market Forecasting

Henning Wachsmuth
Universität Paderborn
Software Quality Lab
hwachsmuth@slab.upb.de

Peter Prettenhofer and **Benno Stein**
Bauhaus-Universität Weimar
Web Technology & Information Systems
benno.stein@uni-weimar.de

Abstract

Strategic business decision making involves the analysis of market forecasts. Today, the identification and aggregation of relevant market statements is done by human experts, often by analyzing documents from the World Wide Web. We present an efficient information extraction chain to automate this complex natural language processing task and show results for the identification part. Based on time and money extraction, we identify sentences that represent statements on revenue using support vector classification. We provide a corpus with German online news articles, in which more than 2,000 such sentences are annotated by domain experts from the industry. On the test data, our statement identification algorithm achieves an overall precision and recall of 0.86 and 0.87 respectively.

1 Introduction

Touch screen market to hit $9B by 2015. 50 suppliers provide multi-touch screens, and that number is likely to rise.[1]

Strategic business decision making is a highly complex process that requires experience as well as an overall view of economics, politics, and technological developments. Clearly, for the time being this process cannot be done by a computer at the level of a human expert. However, important tasks may be automated such as market forecasting, which relies on identifying and aggregating relevant information from the World Wide Web (Berekoven et. al., 2001). An analyst who interprets the respective data can get a reasonable idea about the future market volume, for example. The problem is that a manually conducted Web search is time-consuming and usually far from being exhaustive. With our research we seek to develop an efficient system that finds and analyzes market forecast information with retrieval, extraction and natural language processing (NLP) techniques.

We contribute to the following situation. For a given product, technology, or industry sector we identify and aggregate statements on its market development found on relevant websites. In particular, we extract time information (“by 2015”) and money information (“$9B”) and use support vector classification to identify sentences that represent market statements. The statements’ subjects (“touch screen”) are found by relating recognized named entities to the time and money information, which we then normalize and aggregate. In this paper we report on results for the statement identification. To the best of our knowledge no data for the investigation of such market analysis tasks has been made publicly available until now. We provide such a corpus with statements on revenue annotated in news articles from the Web; the corpus was created in close collaboration with our industry partner *Resolto Informatik GmbH*.

We pursue two objectives, namely, to support human experts with respect to the effectiveness and completeness of their analysis, and to establish a technological basis upon which more intricate analysis tasks can be automated. To summarize, the main contributions of this paper are:

1. We show how to decompose the identification and aggregation of forecasts into retrieval, extraction, and normalization tasks.

2. We introduce a manually annotated German corpus for computational linguistics research on market information.

3. We offer empirical evidence that classification and extraction techniques can be com-

[1] Adapted from http://industry.bnet.com.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1128–1136,
Beijing, August 2010

bined to precisely identify statements on revenue.

1.1 Related Work

Stein et. al. (2005) were among the first to consider information extraction for automatic market forecasting. Unlike us, the authors put much emphasis on retrieval aspects and applied dependency grammar parsing to identify market statements. As a consequence their approach suffers from the limitation to a small number of predefined sentence structures.

While we obtain market forecasts by extracting expert statements from the Web, related approaches derive them from past market behavior and quantitative news data. Koppel and Shtrimberg (2004) studied the effect of news on financial markets. Lavrenko et al. (2000) used time-series analysis and language models to predict stock market prices and, similarly, Lerman et al. (2008) proposed a system for forecasting public opinion based on concurrent modeling of news articles and market history. Another related field is opinion mining in the sense that it relies on the aggregation of individual statements. Glance et al. (2005) inferred marketing intelligence from opinions in online discussions. Liu et al. (2007) examined the effect of Weblogs on box office revenues and combined time-series with sentiment analysis to predict the sales performance of movies.

The mentioned approaches are intended to reflect or to predict present developments and, therefore, primarily help for *operative* decision making. In contrast, we aim at predicting long-term market developments, which are essential for *strategic* decision making.

2 The Problem

Market forecasts depend on two parameters, the *topic* of interest and the *criterion* to look at. A topic is either an organization or a market. Under a market we unite branches, products, and technologies, because the distinction between these is not clear in general (e.g., for semiconductors). In contrast, we define a criterion to be a metric attribute that can be measured over time. Here we are interested in financial criteria such as revenue, profit, and the like. The ambitious overall task that we want to solve is as follows:

Task description: Given a topic τ and a financial criterion χ, find information for τ on the development of χ. Aggregate the found values on χ with respect to time.

We omit the limitation to forecasts because we could miss useful information otherwise:

(1) *In 2008*, the Egyptian automobile industry achieved *US$ 9.96bn* in sales.

(2) Egypt's automotive sales will rise *by 97% from 2008 to 2013*.

Both sentences have the same topic. In Particular, the 2008 amount of money from example (1) can be aggregated with the forecast in (2) to infer the predicted amount in 2013.

As in these examples, market information can often only be found in running text; the major source for this is the Web. Thus, we seek to find web pages with sentences that represent *statements on a financial criterion* χ and to make these statements processable. Conceptually, such a statement is a 5-tuple $\mathcal{S}_\chi = (S, g, T, M, t_d)$, where S is the topical subject, which may have a geographic scope g, T is a period of time, M consists of a growth rate and/or an amount of money to be achieved during T with respect to χ, and t_d is the statement time, i.e., the point in time when the statement was made.

3 Approach

Our goal is to find and aggregate statements on a criterion χ for a topic τ. In close collaboration with two companies from the semantic technology field, we identified eight high-level subtasks in the overall process as explained in the following. An overview is given in Table 1.

3.1 Find Candidate Documents

To find web pages that are likely to contain statements on χ and τ, we propose to perform a meta-search by starting from a set of characteristic terms of the domain and then using query expansion techniques such as local context analysis (Xu and Croft, 2000). As Stein et. al. (2005) describe,

Subtask	Applied technologies
1 Find candidate documents	meta-search, query expansion, genre analysis
2 Preprocess content	content extraction, sentence splitting, tokenization, POS tagging and chunking
3 Extract entities	time and money extraction, named entity recognition of organizations and markets
4 Identify statements	statistical classification based on lexical and distance features
5 Determine statement type	relation extraction based on dependency parse trees, matching of word lists
6 Fill statement templates	template filling, anaphora resolution, matching of word lists
7 Normalize values	time and money normalization, coreference resolution
8 Aggregate information	chronological merging and averaging, inference from subtopic to topic

Table 1: Subtasks of the identification and aggregation of market statements for a specified topic. Experiments in this paper cover the subtasks written in black.

a genre analysis, which classifies a document with respect to its form, style, and targeted audience, may be deployed afterwards to further improve the quality of the result list efficiently. In this way, we only maintain candidate documents that look promising on the surface.

3.2 Preprocess Content

Preprocessing is needed for accurate access to the document text. Our overall task incorporates relating information from different document areas, so mixing up a web page's main frame and sidebars should be avoided. We choose Document Slope Curve (DSC) for content detection, which looks for plateaus in the HTML tag distribution. Gottron (2007) has offered evidence that DSC is currently the best algorithm in terms of precision. Afterwards, the sentences are split with rules that consider the specific characteristics of reports, press releases and the like, such as headlines between short paragraphs. In succeeding subtasks, tokens as well as their Part-of-Speech and chunk tags are also used, but we see no point in not relying on standard algorithms here.

3.3 Extract Entities

The key to identify a statement $\mathcal{S}_\chi$ on a financial criterion χ is the extraction of temporal and monetary entities. Recent works report that statistical approaches to this task can compete with hand-crafted rules (Ahn et. al., 2005; Cramer et. al., 2007). In the financial domain, however, the focus is only on dates and periods as time information, along with currency numbers, currency terms, or fractions as money information. We found that with regular expressions, which represent the complex but finite structures of such phrases, we can achieve nearly perfect recall in recognition (see Section 5).

We apply named entity recognition (NER) of organizations and markets in this stage, too, so we can relate statements to the appropriate subjects, later on. Note that market names do not follow a unique naming scheme, but we observed that they often involve similar phrase patterns that can be exploited as features. NER is usually done by sequence labeling, and we use heuristic beam search due to our effort to design a highly efficient overall system. Ratinov and Roth (2009) have shown for the CoNLL-2003 shared task that Greedy decoding (i.e., beam search of width 1) is competitive to the widely used Viterbi algorithm while being over 100 times faster at the same time.

3.4 Identify Statements

Based on time and money information, sentences that represent a statement $\mathcal{S}_\chi$ can be identified. Such a sentence gives us valuable hints on which temporal and monetary entity stick together and how to interpret them in relation. Additionally, it serves as evidence for the statement's correctness (or incorrectness). Every sentence with at least one temporal and one monetary entity is a candidate. Criteria such as revenue usually imply small core vocabularies $\mathcal{L}_{pos}$, which indicate that a sentence is on that criterion or which often appear close to it. On the contrary, there are sets of words $\mathcal{L}_{neg}$ that suggest a different criterion. For a given text collection with known statements on χ, both $\mathcal{L}_{pos}$ and $\mathcal{L}_{neg}$ can be found by computing the most discriminant terms with respect to χ. A reasonable first approach is then to filter sentences that contain terms from $\mathcal{L}_{pos}$ and lack terms from $\mathcal{L}_{neg}$, but problems arise when terms from different vocabularies co-occur or statements on different criteria are attached to one another.

Instead, we propose a statistical learning approach. Support Vector Machines (SVMs) have been proven to yield very good performance in both general classification and sentence extraction while being immune to overfitting (Steinwart and Christmann, 2008; Hirao et. al., 2001). For our candidates, we compute lexical and distance features based on $\mathcal{L}_{pos}$, $\mathcal{L}_{neg}$, and the time and money information. Then we let an SVM use these features to distinguish between sentences with statements on χ and others. At least for online news articles, this works reasonably well as we demonstrate in Section 5. Note that classification is not used to match the right entities, but to filter the small set of sentences on χ.

3.5 Determine Statement Type

The statement type implies what information we can process. If a sentence contains more than one temporal or monetary entity, we need to relate the correct T and M to each $\mathcal{S}_\chi$, now. The type of $\mathcal{S}_\chi$ then depends on the available money information, its *trend* and the *time direction*.

We consider four types of money information. χ refers to a period of time that results in a *new amount* A of money in contrast to its *preceding amount* A_p. The difference between A and A_p may be specified as an *incremental amount* Δ_A or as a *relative growth rate* r. M can span any combination of A, A_p, Δ_A and r, and at least A and r constitute a reasonable entity on their own. Sometimes the trend of r (i.e. decreasing or increasing) cannot be derived from the given values. However, this information can mostly be obtained from a nearby indicator word (e.g. “plus” or “decreased”) and, therefore, we address this problem with appropriate word lists. Once the trend is known, any two types imply the others.

Though we are predominantly interested in *forecasts*, statements also often represent a *declaration* on achieved results. This distinction is essential and can be based on time-directional indicators (e.g. “next”) and the tense of leading verbs. For this, we test both feature and kernel methods on dependency parse trees, thereby determining T and M at the same time. We only parse the identified sentences, though. Hence, we avoid running into efficiency problems.

3.6 Fill Statement Templates

The remaining subtasks are ongoing work, so we only present basic concepts here.

Besides T and M, the subject S and the statement time t_d have to be determined. S may be found within the previously extracted named entities using the dependency parse tree from Section 3.5 or by anaphora resolution. Possible limitations to a geographic scope g can be recognized with word lists. In market analysis, the approximate t_d suffices, and for most news articles t_d is similar to their release date. Thus, if no date is in the parse tree, we search the extracted temporal entities for the release date, which is often mentioned at the beginning or end of the document’s content. We fill one template (S, g, T, M, t_d) for each $\mathcal{S}_\chi$ where we have at least S, T, and M.

3.7 Normalize Values

Since we base the extraction on regular expressions, we can normalize most monetary entities with a predefined set of rules. Section 3.5 implies that $M^* = (A^*, r^*)$ is a reasonable normalized form where A^* is A specified in million US-$ and r^* is r as percentage with a fixed number of decimals.[2] Time normalization is more complex. Any period should be transformed to $T^* = (t_s^*, t_e^*)$ consisting of the start date t_s^* and end date t_e^*. Following Ahn et. al. (2005), we consider fully qualified, deictic and anaphoric periods. While normalization of fully qualified periods like “from Apr to Jun 1999” is straightforward, deictic (e.g. “since 2005”, “next year”) and anaphoric mentions (e.g. “in the reported time”) require a reference time. Approaches to resolve such references rely on dates or fully qualified periods in the preceding text (Saquete et. al., 2003; Mani and Wilson, 2000).[3]

[2] Translating the currency requires exchange rates at statement time. We need access to such information or omit the translation if only one currency is relevant.

[3] References to fiscal years even involve a whole search problem if no look-up table on such data is available.

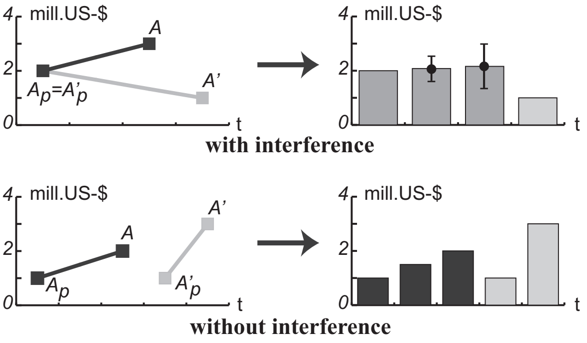


Figure 1: Example for merging monetary values.

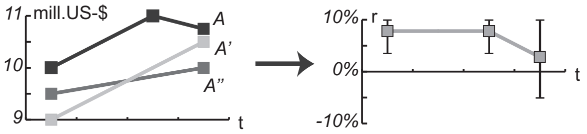


Figure 2: Example for the inference of relative information from absolute values.

If we cannot normalize M or T, we discard the corresponding statement templates. For the others, we have to resolve synonymous co-references (e.g. “Loewe AG” and “Loewe”) before we can proceed to the last step.

3.8 Aggregate Information

We can aggregate the normalized values in either two or three dimensions depending on whether to separate statements with respect to t_d. Aggregation then incorporates two challenges, namely, how to merge values and how to infer information on a topic from values of a subtopic.

We say that two statements on the same topic τ and criterion χ *interfere* if the contained periods of time intersect and the according monetary values do not coincide. In case of declarations, this means that we extracted incorrect values or extracted values incorrectly. For forecasts, on the contrary, we are exactly onto such information. In both cases, an intuitive solution is to compute the average (or median) and deviations. Figure 1 graphically illustrates such merging. The subtopic challenge is based on the assumption that a meaningful number of statements on a certain subtopic of τ implies relative information on τ, as shown in Figure 2. One of the most interesting relations are organizations as subtopics of markets they produce for, because it is quite usual to search for information on a market, but only receive statements on companies. Approaches to this relation may rely e.g. on the web page co-occurrence and term frequencies of the markets and companies.

Statements	Total	Forecasts	Declarations
Complete corpus	2075	523 (25.2%)	1552 (74.8%)
Training set	1366	306 (22.4%)	1060 (77.6%)
Validation set	362	113 (31.2%)	249 (68.8%)
Test set	347	104 (30.0%)	243 (70.0%)

Table 2: Statements on revenue in the corpus.

Altogether, we return the aggregated values linked to the sentences in which we found them. In this way, we make the results verifiable and, thereby, compensate for possible inaccuracies.

4 Corpus

To evaluate the given and related tasks, we built a manually annotated corpus with online news articles on the revenues of organizations and markets. The compilation aims at being representative for target documents, a search engine returns to queries on revenue. The purpose of the corpus is to investigate both the structure of sentences on financial criteria and the distribution of associated information over the text.

The corpus consists of 1,128 German news articles from the years 2003 to 2009, which were taken from 29 news websites like *www.spiegel.de* or *www.capital.de*. The content of each document comes as unicode plain text with appended URL for access to the HTML source code. Annotations are given in a standard XMI file preformatted for the *Unstructured Information Management Architecture* (Ferrucci and Lally, 2004). We created a split, in which 2/3 of the documents constitute the training set and each 1/6 refers to the validation and test set. To simulate real conditions, the training documents were randomly chosen from only the seven most represented websites, while the validation and test data both cover all 29 sources. Table 2 shows some corpus statistics, which give a hint that the validation and test set differ significantly from the training set. The corpus is free for scientific use and can be downloaded at http://infexba.upb.de.

Loewe AG: Vorläufige Neun-Monats-Zahlen
Kronach, [6. November 2007]$_{REF}$ — Das Ergebnis vor Zinsen und Steuern (EBIT) des Loewe Konzerns konnte in den ersten 9 Monaten 2007 um 41% gesteigert werden. Vor diesem Hintergrund hebt die [Loewe AG]$_{ORG}$ ihre EBIT-Prognose für das laufende Geschäftsjahr auf 20 Mio. Euro an. ***Beim Umsatz strebt Konzernchef [Rainer Hecker]$_{AUTH}$ [für das Gesamtjahr]$_{TIME}$ ein höher als ursprünglich geplantes [Wachstum]$_{TREND}$ [von 10% auf ca. 380 Mio. Euro]$_{MONEY}$ an.*** *(...)*

Figure 3: An annotated document in the corpus. The text is taken from *www.boerse-online.de*, but has been modified for clarification.

4.1 Annotations

In each document, every sentence that includes a temporal entity T and a monetary entity M and that represents a *forecast* or *declaration* on the revenue of an organization or market is marked as such. T and M are annotated themselves and linked to the sentence. Accordingly, the *subject* is tagged (and linked) within the sentence boundaries if available, otherwise its last mention in the preceding text. The same holds for optional entities, namely a *reference time*, a *trend indicator* and the *author* of a statement. Altogether, 2,075 statements are tagged in this way. As in Figure 3, only information that refers to a statement on revenue (typed in bold face) is annotated. These annotations may be spread across the text.

The source documents were manually selected and prepared by our industrial partners, and two of their employees annotated the plain document text. With respect to the statement annotations, a preceding pilot study yielded substantial inter-annotator agreement, as indicated by the value $\kappa = 0.79$ of the conservative measure *Cohen's Kappa* (Carletta, 1996). Additionally, we performed a manual correction process for each annotated document to improve consistency.

5 Experiments

We now present experiments for the statement identification, which were conducted on our corpus. The goal was to evaluate whether our combined extraction and classification approach succeeds in the precise identification of sentences that comprise a statement on revenue, while keeping recall high. Only exact matches of the annotated text spans were considered to be correct identifications. Unlike in Section 3, we only worked on plain text, though.

5.1 Experimental Setup

To find candidate sentences, we implemented a sentence splitter that can handle article elements such as subheadings, URLs, or bracketed sentences. We then constructed sophisticated, but efficient regular expressions for time and money. They do not represent correct language, in general, but model the structure of temporal and monetary entities, and use word lists provided by domain experts on the lowest level.[4] For feature computation, we assumed that the closest pair of temporal and monetary entity refers to the enclosing candidate sentence.[5] Since only positive instances I_P of statements on revenue are annotated in our corpus, we declared all candidates, which have no counterpart in the annotated data, to constitute the negative class I_N, and balanced I_P and I_N by "randomly" (seed 42) removing instances from I_N.[6]

For the vocabularies $\mathcal{L}_{pos} = \{P_1, P_2\}$ we first counted the frequencies of all words in the unbalanced sets I_P and I_N. From these, we deleted named entities, numbers and adjectives. If the prefix (e.g. "Umsatz") of a word ("Umsatzplus") occurred, we only kept the prefix. We then filtered all terms that appeared in at least 1.25% of the instances in I_P and more than 3.5 times as much in I_P as in I_N. The remaining words were manually partitioned into two lists:

P_1 = {umgesetzt, Umsatz, Umsätze, setzte} (all of these are terms for revenue)

P_2 = {Billionen, meldet, Mitarbeiter, Verband} (trillions, announce, employee, association)

$\mathcal{L}_{neg} = \{N_1, N_2\}$ was built accordingly. In addition, we set up a list G_1 with genitive pronouns

[4] More details are given at http://infexba.upb.de.

[5] 55% of the candidate sentences in the training set contain more than one temporal and/or monetary entity, so this assumption may lead to errors.

[6] We both tested undersampling and oversampling techniques but saw no effective differences in the results.

and determiners. Based on $\mathcal{L}_{pos}$, $\mathcal{L}_{neg}$ and G_1, we computed the following 43 features for every candidate sentence s:

- **1-8:** Number of terms from P_1 (N_1) in s as well as in the two preceding sentences and in the following sentence.
- **9-10:** Number of terms from P_2 (N_2) in s.
- **11:** Occurrence of term from G_1 next to the monetary entity.
- **12-19:** Forward (backward) distance in tokens between the monetary (temporal) entity in s and a term from P_1 (N_1).
- **20-27:** Forward (backward) distance in number of symbols from $O_1 = \{$'.','?','!'$\}$ between the monetary (temporal) entity in s and a term from P_1 (N_1).
- **28-43:** Same as 20-27 for $O_2 = \{$':',';'$\}$ and $O_3 = \{$','$\}$, respectively.

We trained a linear SVM with cost parameter $C = 0.3$ (selected during validation) on these features using the *Weka* integration of *LibSVM* (Hall et. al., 2009; Fan et. al., 2001). Further features were evaluated, e.g. occurrences of contrapositions or comparisons, but they did not improve the classifier. Instead, we noticed that we can avoid some complex cases when we apply two rules after entity extraction:

R_1: Delete temporal and monetary entities that are directly surrounded by brackets.

R_2: Delete temporal entities that contain the word "Vorjahr" ("preceding year").

Now, we evaluated the following five statement identification algorithms:

- **Naïve:** Simply return all candidate sentences (to estimate the relative frequency of statements on revenue in the corpus).
- **Baseline:** Return all candidate sentences that contain a term from the list P_1.
- **NEG:** Use the results from Baseline. Return all sentences that lack terms from N_1.
- **RB:** Filter candidates using R_1 and R_2. Then apply NEG.
- **SVM:** Filter candidates using R_1 and R_2. Then classify sentences with the SVM.

Recall	**Training**	**Validation**	**Test**
Sentences	0.98	0.98	0.96
Temporal entities	0.97 (0.95)	0.97 (0.94)	0.98 (0.96)
Monetary entities	0.96 (0.96)	0.96 (0.96)	0.95 (0.94)

Table 3: Recall of sentence and entity extraction. In brackets: Recall after applying R_1 and R_2.

5.2 Results

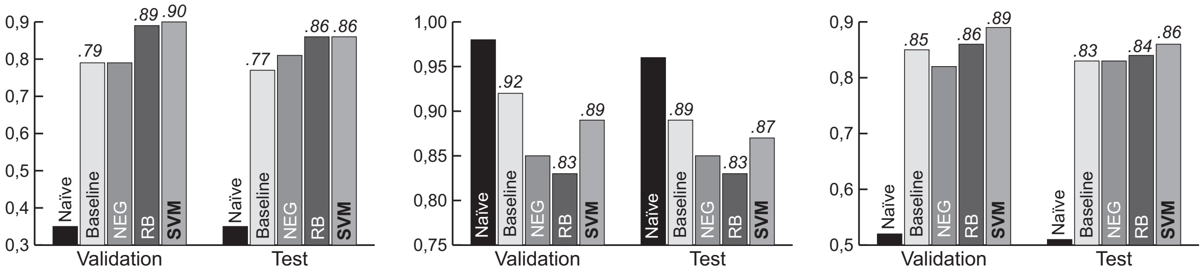
Table 3 shows that we found at least 95% of the sentences, time and money information, which refer to a statement on revenue, in all datasets.[7] We could not measure precision for these since not all sentences and entities are annotated in the corpus, as mentioned in Section 4.

Results for the statement identification are given in Figure 4. Generally, the test values are somewhat lower than the validation values, but analog in distribution. Nearly all statements were recognized by the Naïve algorithm, but only with a precision of 0.35. In contrast, both for Baseline and NEG already around 80% of the found statements were correct. The latter paid a small gain in precision with a significant loss in recall. While RB and SVM both achieved 86% precision on the test set, SVM tends to be a little more precise as suggested by the validation results. In terms of recall, SVM clearly outperformed RB with values of 89% and 87% and was only a little worse than the Baseline. Altogether, the F_1-Measure values show that SVM was the best performing algorithm in our evaluation.

5.3 Error Analysis

To assess the influence of the sentence, time and money extraction, we compared precision and recall of the classifier on the manually annotated and the extracted data, respectively. Table 4 shows

[7] We intentionally did not search for unusual entities like "am 1. Handelstag nach dem Erntedankfest" ("the 1st trading day after Thanksgiving") in order not to develop techniques that are tailored to individual cases. Also, money amounts that lack a currency term were not recognized.



Precision **Recall** **F$_1$-Measure**

Figure 4: Precision, recall and F_1-Measure of the five evaluated statement identification algorithms. SVM is best in precision both on validation and test data and outperforms RB in recall significantly.

that only recall differs significantly. We found that false statement identifications referred to the following noteworthy error cases.

False match: Most false positives result from matchings of temporal and monetary entities that actually do not refer to the same statement.

Missing criterion: Some texts describe the development of revenue without ever mentioning revenue. Surrogate words like "market" may be used, but they are not discriminative enough.

Multiple criteria: Though we aimed at discarding sentences, in which revenue is mentioned without comprising a statement on it, in some cases our features did not work out, mainly due to intricate sentence structure.

Traps: Some sentences contain numeric values on revenue, but not the ones looked for, as in "10% of the revenue". We tackled these cases, but had still some false classifications left.

Hidden boundaries: Finally, we did not find all correct sentence boundaries, which can lead to both false positives and false negatives. The predominant problem was to separate headlines from paragraph beginnings and is partly caused by the missing access to markup tags.

5.4 Efficiency

We ran the identification algorithm on the whole corpus using a *2 GHz Intel Core 2 Duo MacBook* with *4 GB RAM*. The 1,128 corpus documents contain 33,370 sentences as counted by our algorithm itself. Tokenization, sentence splitting, time and money extraction took only 55.2 seconds, i.e., more than 20 documents or 600 sentences each second. Since our feature computation is not optimized yet, the complete identification process is a little less efficient with 7.35 documents or 218 sentences per second. However, it is fast enough to be used in online applications, which was our goal in the end.

Candidates	Data	Precision	Recall
Annotated	validation data	0.91	0.94
	test data	0.87	0.93
Extracted	validation data	0.90	0.89
	test data	0.86	0.87

Table 4: Precision and recall of the statement identification on manually annotated data and on automatically extracted data, respectively.

6 Conclusion

We presented a multi-stage approach for the automatic identification and aggregation of market statements and introduced a manually annotated German corpus for related tasks. The approach has been influenced by industry and is oriented towards practical applications, but is, in general, not specific to the German language. It relies on efficient retrieval, extraction and NLP techniques. By now, we can precisely identify most sentences that represent statements on revenue. This already allows for the support of strategists, e.g. by highlighting such sentences in web pages, which we currently implement as a Firefox extension. The overall problem is complex, though, and we are aware that human experts can do better at present. Nevertheless, time-consuming tasks can be automated and, in this respect, the results on our corpus are very promising.

Acknowledgement: This work was funded by the project "InfexBA" of the German Federal Ministry of Education and Research (BMBF) under contract number 01IS08007A.

References

Ahn, David, Sisay F. Adafre, and Maarten de Rijke. 2005. Extracting Temporal Information from Open Domain Text: A Comparative Exploration. *Journal of Digital Information Management*, 3(1): 14–20.

Berekoven, Ludwig, Werner Eckert, and Peter Ellenrieder. 2001. *Marktforschung: Methodische Grundlagen und praktische Anwendung*, 9th Edition, Gabler, Wiesbaden, Germany.

Carletta, Jean. 1996. Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics*, 22: 249–254.

Cramer, Irene M., Stefan Schacht, and Andreas Merkel. 2007. Classifying Number Expressions in German Corpora. In *Proceedings of the 31st Annual Conference of the German Classification Society on Data Analysis, Machine Learning, and Applications*, pages 553–560.

Fan, Rong-En, Pai-Hsuen Chen, and Chih-Jen Lin. 2001. Working Set Selection Using Second Order Information for Training Support Vector Machines. *Journal of Machine Learning Research*, 6: 1889–1918.

Ferrucci, David and Adam Lally. 2004. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*, 10(3–4): pages 327–348.

Glance, Natalie, Matthew Hurst, Kamal Nigam, Matthew Siegler, Robert Stockton, and Takashi Tomokiyo. 2005. Deriving Marketing Intelligence from Online Discussion. In *Proceedings of the Eleventh International Conference on Knowledge Discovery in Data Mining*, pages 419–428.

Gottron, Thomas. 2007. Evaluating Content Extraction on HTML Documents. In *Proceedings of the 2nd International Conference on Internet Technologies and Applications*, pages 123–132.

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1).

Hirao, Tsutomu, Hideki Isozaki, Eisaku Maeda and Yuji Matsumoto. 2002. Extracting Important Sentences with Support Vector Machines. In *Proceedings of the 19th International Conference on Computational linguistics*, pages 342–348.

Koppel, Moshe and Itai Shtrimberg. 2004. Good News or Bad News? Let the Market Decide. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*, pages 86–88.

Lavrenko, Victor, Matt Schmill, Dawn Lawrie, Paul Ogilvie, David Jensen, and James Allan. 2000. Mining of Concurrent Text and Time Series. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Workshop on Text Mining*, pages 37–44.

Lerman, Kevin, Ari Gilder, Mark Dredze, and Fernando Pereira. 2008. Reading the Markets: Forecasting Public Opinion of Political Candidates by News Analysis. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 473–480.

Liu, Yang, Xiangji Huang, Aijun An, and Xiaohui Yu. 2007. Arsa: A Sentiment-Aware Model for Predicting Sales Performance Using Blogs. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 607–614.

Mani, Inderjeet and George Wilson. 2000. Robust Temporal Processing of News. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 69–76.

Ratinov, Lev and Dan Roth. 2009. Design Challenges and Misconceptions in Named Entity Recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155.

Saquete, Estela, Rafael Muñoz, and Patricio Martínez-Barco. 2003. TERSEO: Temporal Expression Resolution System Applied to Event Ordering. *Text, Speech and Dialogue*, Springer, Berlin / Heidelberg, Germany, pages 220–228.

Stein, Benno, Sven Meyer zu Eissen, Gernot Gräfe, and Frank Wissbrock. 2005. Automating Market Forecast Summarization from Internet Data. *Fourth International Conference on WWW/Internet*, pages 395–402.

Steinwart, Ingo and Andreas Christmann. 2008. *Support Vector Machines*, Springer, New York, NY.

Xu, Jinxi and Bruce W. Croft 2000. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems*, 18(1): 79-112.

Towards a Unified Approach to Simultaneous Single-Document and Multi-Document Summarizations

Xiaojun Wan
Institute of Compute Science and Technology
The MOE Key Laboratory of Computational Linguistics
Peking University
wanxiaojun@icst.pku.edu.cn

Abstract

Single-document summarization and multi-document summarization are very closely related tasks and they have been widely investigated independently. This paper examines the mutual influences between the two tasks and proposes a novel unified approach to simultaneous single-document and multi-document summarizations. The mutual influences between the two tasks are incorporated into a graph model and the ranking scores of a sentence for the two tasks can be obtained in a unified ranking process. Experimental results on the benchmark DUC datasets demonstrate the effectiveness of the proposed approach for both single-document and multi-document summarizations.

1 Introduction

Single-document summarization aims to produce a concise and fluent summary for a single document, and multi-document summarization aims to produce a concise and fluent summary for a document set consisting of multiple related documents. The two tasks are very closely related in both task definition and solution method. Moreover, both of them are very important in many information systems and applications. For example, given a cluster of news articles, a multi-document summary can be used to help users to understand the whole cluster, and a single summary for each article can be used to help users to know the content of the specified article.

To date, single-document and multi-document summarizations have been investigated extensively and independently in the NLP and IR fields. A series of special conferences or workshops on automatic text summarization (e.g. SUMMAC, DUC, NTCIR and TAC) have advanced the technology and produced a couple of experimental online systems. However, the two summarization tasks have not yet been simultaneously investigated in a unified framework.

Inspired by the fact that the two tasks are very closely related and they can be used simultaneously in many applications, we believe that the two tasks may have mutual influences on each other. In this study, we propose a unified approach to simultaneous single-document and multi-document summarizations. The mutual influences between the two tasks are incorporated into a graph-based model. The ranking scores of sentences for single-document summarization and the ranking scores of sentences for multi-document summarization can boost each other, and they can be obtained simultaneously in a unified graph-based ranking process. To the best of our knowledge, this study is the first attempt for simultaneously addressing the two summarization tasks in a unified graph-based framework. Moreover, the proposed approach can be easily adapted for topic-focused summarizations.

Experiments have been performed on both the single-document and multi-document summarization tasks of DUC2001 and DUC2002. The results demonstrate that the proposed approach can outperform baseline independent methods for both the two summarization tasks. The two tasks are validated to have mutual influences on each other.

The rest of this paper is organized as follows: Section 2 introduces related work. The details of the proposed approach are described in Section 3. Section 4 presents and discusses the evaluation results. Lastly we conclude our paper in Section 5.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1137–1145,
Beijing, August 2010

2 Related Work

Document summarization methods can be either extraction-based or abstraction-based. In this section, we focus on extraction-based methods.

Extraction-based methods for single-document summarization usually assign a saliency score to each sentence in a document and then rank and select the sentences. The score is usually computed based on a combination of statistical and linguistic features, such as term frequency, sentence position, cue words and stigma words (Luhn, 1969; Edmundson, 1969; Hovy and Lin, 1997). Machine learning techniques have also been used for sentence extraction (Kupiec et al., 1995; Conroy and O'Leary, 2001; Shen et al., 2007; Li et al., 2009). The mutual reinforcement principle has been exploited to iteratively extract key phrases and sentences from a document (Zha, 2002; Wan et al, 2007a). Wan et al. (2007b) propose the CollabSum algorithm to use additional knowledge in a cluster of documents to improve single document summarization in the cluster.

In recent years, graph-based ranking methods have been investigated for document summarization, such as TextRank (Mihalcea and Tarau, 2004; Mihalcea and Tarau, 2005) and LexPageRank (ErKan and Radev, 2004). Similar to PageRank (Page et al., 1998), these methods first build a graph based on the similarity relationships between the sentences in a document and then the saliency of a sentence is determined by making use of the global information on the graph recursively. The basic idea underlying the graph-based ranking algorithm is that of "voting" or "recommendation" between sentences.

Similar methods have been used for generic multi-document summarization. A typical method is the centroid-based method (Radev et al., 2004). For each sentence, the method computes a score based on each single feature (e.g. cluster centroids, position and TFIDF) and then linearly combines all the scores into an overall sentence score. Topic signature is used as a novel feature for selecting important content in NeATS (Lin and Hovy, 2002). Various sentence features have been combined by using machine learning techniques (Wong et al., 2008). A popular way for removing redundancy between summary sentences is the MMR algorithm (Carbonell and Goldstein, 1998). Themes (or topics, clusters) in documents have been discovered and used for sentence selection (Harabagiu and Lacatusu, 2005). Hachey (2009) investigates the effect of various source document representations on the accuracy of the sentence extraction phase of a multi-document summarization task. Graph-based methods have also been used to rank sentences in a document set. The methods first construct a graph to reflect sentence relationships at different granularities, and then compute sentence scores based on graph-based learning algorithms. For example, Wan (2008) proposes to use only cross-document relationships for graph building and sentence ranking. Cluster-level information has been incorporated in the graph model to better evaluate sentences (Wan and Yang, 2008).

For topic-focused multi-document summarization, many methods are extensions of generic summarization methods by incorporating the information of the given topic or query into generic summarizers. In recent years, a few novel methods have been proposed for topic-focused summarization (Daumé and Marcu, 2006; Wan et al., 2007c; Nastase 2008; Li et al., 2008; Schilder and Kondadadi, 2008; Wei et al., 2008).

The above previous graph-based summarization methods aim to address either single-document summarization or multi-document summarization, and the two summarization tasks have not yet been addressed in a unified graph-based framework.

3 The Unified Summarization Approach

3.1 Overview

Given a document set, in which the whole document set and each single document in the set are required to be summarized, we use *local saliency* to indicate the importance of a sentence in a particular document, and use *global saliency* to indicate the importance of a sentence in the whole document set.

In previous work, the following two assumptions are widely made for graph-based summarization models:

Assumption 1: A sentence is locally important in a particular document if it is heavily linked with many locally important sentences in the same document.

Assumption 2: A sentence is globally important in the document set if it is heavily linked with many globally important sentences in the document set.

The above assumptions are the basis for PageRank-like algorithms for single document summarization and multi-document summarization, respectively. In addition to the above two assumptions, we make the following two assumptions to consider the mutual influences between the two summarization tasks:

Assumption 3: A sentence is locally important in a particular document, if it is heavily linked with many globally important sentences in the document set.

The above assumption is reasonable because the documents in the set are relevant and the globally important information in the document set will be expressed in many single documents. Therefore, if a sentence is salient in the whole document set, the sentence may be salient in a particular document in the set.

Assumption 4: A sentence is globally important in the document set, if it is heavily linked with many locally important sentences.

The above assumption is reasonable because the documents in the set are relevant and the globally important information in the whole set is the aggregation of the locally important information in each single document. Therefore, if a sentence is salient in a particular document, the sentence has the potential to be salient in the whole document set.

In brief, the local saliency and global saliency of a sentence can mutually influence and boost each other: high local saliency will lead to high global saliency, and high global saliency will lead to high local saliency.

Based on the above assumptions, our proposed approach first builds affinity graphs (each graph is represented by an affinity matrix) to reflect the different kinds of relationships between sentences, respectively, and then iteratively computes the local saliency scores and the global saliency scores of the sentences based on the graphs. Finally, the algorithm converges and the local saliency score and global saliency score of each sentence are obtained. The sentences with high local saliency scores in a particular document are chosen into the summary of the single document, and the sentences with high global saliency scores in the set are chosen into the summary of the document set.

Note that for both summarization tasks, after the saliency scores of sentences have been obtained, the greedy algorithm used in (Wan et al., 2007c) is applied to remove redundancy and finally choose both informative and novel sentences into the summary.

3.2 Algorithm Details

Formally, the given document set is denoted as $D=\{d_i|1\leq i\leq m\}$, and the whole sentence set is denoted as $S=\{s_i|1\leq i\leq n\}$. We let $Info_{single}(s_i)$ denote the local saliency score of sentence s_i in a particular document $d(s_i)\in D$, and it is used to select summary sentences for the single document $d(s_i)$. And we let $Info_{multi}(s_i)$ denote the global saliency score of sentence s_i in the whole document set D, and it is used to select summary sentences for the document set D.

The four assumptions in Section 3.1 can be rendered as follows:

$$Info_{\sin gle}(s_i) \propto \sum_j (W_A)_{ji}\, Info_{\sin gle}(s_j) \quad (1)$$

$$Info_{multi}(s_i) \propto \sum_j (W_B)_{ji}\, Info_{multi}(s_j) \quad (2)$$

$$Info_{\sin gle}(s_i) \propto \sum_j (W_C)_{ji}\, Info_{multi}(s_j) \quad (3)$$

$$Info_{multi}(s_i) \propto \sum_j (W_D)_{ji}\, Info_{\sin gle}(s_j) \quad (4)$$

where W_A, W_B, W_C, W_D are $n\times n$ affinity matrices reflecting the different kinds of relationships between sentences in the document set, where n is the number of all sentences in the document set. The detailed derivation of the matrices will be presented later.

After fusing the above equations, we can obtain the following unified forms:

$$\begin{aligned} Info_{\sin gle}(s_i) &= \mu \sum_j (W_A)_{ji}\, Info_{\sin gle}(s_j) \\ &\quad + (1-\mu)\sum_j (W_C)_{ji}\, Info_{multi}(s_j) \end{aligned} \quad (5)$$

$$\begin{aligned} Info_{multi}(s_i) &= \mu \sum_j (W_B)_{ji}\, Info_{multi}(s_j) \\ &\quad + (1-\mu)\sum_j (W_D)_{ji}\, Info_{\sin gle}(s_j) \end{aligned} \quad (6)$$

However, the above summarization method ignores the feature of sentence position, which has been validated to be very important for document summarizations. In order to incorporate this important feature, we add one prior score to each computation as follows:

$$\begin{aligned} Info_{\sin gle}(s_i) &= \alpha \sum_j (W_A)_{ji}\, Info_{\sin gle}(s_j) \\ &+ \beta \sum_j (W_C)_{ji}\, Info_{multi}(s_j) + \gamma \cdot prior_{\sin gle}(s_i) \end{aligned} \quad (7)$$

$$Info_{multi}(s_i) = \alpha \sum_j (W_B)_{ji} Info_{multi}(s_j) + \beta \sum_j (W_D)_{ji} Info_{single}(s_j) + \gamma \cdot prior_{multi}(s_i) \quad (8)$$

where *α*, *β*, *γ*∈[0,1] specify the relative contributions to the final saliency scores from the different factors, and we have *α*+*β*+*γ*=1. $prior_{single}(s_i)$ is the prior score for the local saliency of sentence s_i, and here $prior_{single}(s_i)$ is computed based on sentence position of s_i in the particular document $d(s_i)$. $prior_{multi}(s_i)$ is the prior score for the global saliency of sentence s_i, and we also compute $prior_{multi}(s_i)$ based on sentence position of s_i.

We use two column vectors $\vec{\boldsymbol{u}} = [Info_{single}(s_i)]_{n\times 1}$ and $\vec{\boldsymbol{v}} = [Info_{multi}(s_i)]_{n\times 1}$ to denote the local and global saliency scores of all the sentences in the set, respectively. And the matrix forms of the above equations are as follows:

$$\vec{\boldsymbol{u}} = \alpha \boldsymbol{W}_A^T \vec{\boldsymbol{u}} + \beta \boldsymbol{W}_C^T \vec{\boldsymbol{v}} + \gamma \vec{\boldsymbol{p}}_{single} \quad (9)$$

$$\vec{\boldsymbol{v}} = \alpha \boldsymbol{W}_B^T \vec{\boldsymbol{v}} + \beta \boldsymbol{W}_D^T \vec{\boldsymbol{u}} + \gamma \vec{\boldsymbol{p}}_{multi} \quad (10)$$

where $\vec{\boldsymbol{p}}_{single} = [prior_{single}(s_i)]_{n\times 1}$ and $\vec{\boldsymbol{p}}_{multi} = [prior_{multi}(s_i)]_{n\times 1}$ are the prior column vectors.

The above matrices and prior vectors are constructed as follows, respectively:

W_A: This affinity matrix aims to reflect the local relationships between sentences in each single document, which is defined as follows:

$$(W_A)_{ij} = \begin{cases} sim_{cosine}(s_i, s_j), \text{ if } \mathrm{d}(s_i) = \mathrm{d}(s_j) \\ \qquad\qquad\qquad \text{and } \mathrm{i} \neq \mathrm{j} \\ 0, \text{ Otherwise} \end{cases} \quad (11)$$

where $d(s_i)$ refers to the document containing sentence s_i. $sim_{cosine}(s_i,s_j)$ is the cosine similarity between sentences s_i and s_j.

$$sim_{cosine}(s_i, s_j) = \frac{\vec{s}_i \cdot \vec{s}_j}{|\vec{s}_i| \times |\vec{s}_j|} \quad (12)$$

where $\vec{s}_i$ and $\vec{s}_j$ are the corresponding term vectors of s_i and s_j. Note that we have $(W_A)_{ij} = (W_A)_{ji}$, and we have $(W_A)_{ii}$ =0 to avoid self loops.

We can see that the matrix contains only the within-document relationships between sentences.

W_B: This affinity matrix aims to reflect the global relationships between sentences in the document set, which is defined as follows:

$$(W_B)_{ij} = \begin{cases} sim_{cosine}(s_i, s_j), \text{ if } \mathrm{d}(s_i) \neq \mathrm{d}(s_j) \\ 0, \text{ Otherwise} \end{cases} \quad (13)$$

We can see that the matrix contains only the cross-document relationships between sentences. We do not include the within-document sentence relationships in the matrix because it has been shown that the cross-document relationships are more appropriate to reflect the global mutual influences between sentences than the within-document relationships in (Wan, 2008).

W_C: This affinity matrix aims to reflect the cross-document relationships between sentences in the document set. However, the relationships in this matrix are used for carrying the influences of the sentences in other documents on the local saliency of the sentences in a particular document. If we directly use Equation (13) to compute the matrix, the mutual influences would be overly used. Because other documents might not be sampled from the same generative model as the specified document, we probably do not want to trust them so much as the specified document. Thus a confidence value is used to reflect out belief that the document is sampled from the same underlying model as the specified document. Heuristically, we use the cosine similarity between documents as the confidence value. And we use the confidence value as the decay factor in the matrix computation as follows:

$$(W_c)_{ij} = \begin{cases} sim_{cosine}(s_i, s_j) \times sim_{cosine}(d(s_i), d(s_j)), \\ \qquad\qquad\qquad \text{if } \mathrm{d}(s_i) \neq \mathrm{d}(s_j) \\ 0, \text{ Otherwise} \end{cases} \quad (14)$$

W_D: This affinity matrix aims to reflect the within-document relationships between sentences. Thus we have ***W_D***=***W_A***, which means that the global saliency score of a sentence is influenced only by the local saliency scores of the sentences in the same document, without considering the sentences in other documents.

Note that the above four matrices are symmetric and we can replace $\boldsymbol{W}_A^T$, $\boldsymbol{W}_B^T$, $\boldsymbol{W}_C^T$ and $\boldsymbol{W}_D^T$ by ***W_A***, ***W_B***, ***W_C*** and ***W_D*** in Equations (9) and (10), respectively.

prior_single(s_i): It is computed under the assumption that the first sentences in a document are usually more important than other sentences.

$$prior_{single}(s_i) = 0.5 + \frac{1}{position(s_i) + 1} \quad (15)$$

where $position(s_i)$ returns the position number of sentence s_i in its document $d(s_i)$. For example, if

s_i is the first sentence in its document, $position(s_i)$ is 1.

The prior weight is then normalized by:

$$prior_{\sin gle}(s_i) = \frac{prior_{\sin gle}(s_i)}{\sum_i prior_{\sin gle}(s_i)} \qquad (16)$$

prior$_{multi}$(s$_i$): We also let the prior weight reflect the influence of sentence position.

$$prior_{multi}(s_i) = prior_{\sin gle}(s_i) \qquad (17)$$

And then the prior weight is normalized in the same way.

The above definitions are for generic document summarizations and the above algorithm can be easily adapted for topic-focused summarizations. Given a topic q, the only change for the above computation is $prior_{multi}(s_i)$. The topic relevance is incorporated into the prior weight as follows:

$$prior_{multi}(s_i) = sim_{\cos ine}(s_i, q) \qquad (18)$$

$$prior_{multi}(s_i) = \frac{prior_{multi}(s_i)}{\sum_i prior_{multi}(s_i)} \qquad (19)$$

In order to solve the iterative problem defined in Equations (9) and (10), we let $\vec{\boldsymbol{r}} = [\vec{\boldsymbol{u}}^T \ \ \vec{\boldsymbol{v}}^{\mathrm{T}}]^T$, $\vec{\boldsymbol{p}} = [\gamma\vec{\boldsymbol{p}}^{\mathrm{T}}_{\text{single}} \ \ \gamma\vec{\boldsymbol{p}}^{\mathrm{T}}_{\text{multi}}]^T$, $\boldsymbol{W} = \begin{bmatrix} \alpha\boldsymbol{W}_A^T & \beta\boldsymbol{W}_C^T \\ \beta\boldsymbol{W}_D^T & \alpha\boldsymbol{W}_B^T \end{bmatrix}$, and then the iterative equations correspond to the following linear system:

$$\vec{\boldsymbol{r}} = \boldsymbol{W}\vec{\boldsymbol{r}} + \vec{\boldsymbol{p}} \qquad (20)$$

$$(\boldsymbol{I} - \boldsymbol{W})\vec{\boldsymbol{r}} = \vec{\boldsymbol{p}} \qquad (21)$$

To guarantee the solution of the above linear system, ***W*** is normalized by columns. If all the elements of a column are zero, we replace the elements with 1/(2*n*), where 2*n* equals to the element number of the column. We then multiply ***W*** by a decay factor θ ($0<\theta<1$) to scale down each element in ***W***, but remain the meaning of ***W***. Here, θ is empirically set to 0.6[1]. Finally, Equation (21) is rewritten as follows:

$$(\boldsymbol{I} - \theta \cdot \boldsymbol{W})\vec{\boldsymbol{r}} = \vec{\boldsymbol{p}} \qquad (22)$$

Thus, the matrix (*I-θW*) is a strictly diagonally dominant matrix and the solution of the linear system exists and we can apply the Gauss-Seidel method used in (Li et al., 2008) to solve the linear system. The GS method is a well-know method for numeric computation in mathematics and the details of the method is omitted here.

4 Empirical Evaluation

4.1 Dataset and Evaluation Metric

Generic single-document and multi-document summarizations have been the fundamental tasks in DUC 2001 and DUC 2002 (i.e. tasks 1 and 2 in DUC 2001 and tasks 1 and 2 in DUC 2002), and we used the two datasets for evaluation. DUC2001 provided 309 articles, which were grouped into 30 document sets. Generic summary of each article was required to be created for task 1, and generic summary of each document set was required to be created for task 2. The summary length was 100 words or less. DUC 2002 provided 59 document sets consisting of 567 articles (D088 is excluded from the original 60 document sets by NIST) and generic summaries for each article and each document set with a length of approximately 100 words were required to be created. The sentences in each article have been separated and the sentence information has been stored into files. The summary of the two datasets are shown in Table 1.

	DUC 2001	DUC 2002
Task	Tasks 1, 2	Tasks 1, 2
Number of documents	309	567
Number of clusters	30	59
Data source	TREC-9	TREC-9
summary length	100 words	100 words

Table 1. Summary of datasets

We used the ROUGE toolkit[2] (Lin and Hovy, 2003) for evaluation, which has been widely adopted by DUC for automatic summarization evaluation. It measured summary quality by counting overlapping units such as the n-gram, word sequences and word pairs between the candidate summary and the reference summary.

The ROUGE toolkit reported separate recall-oriented scores for 1, 2, 3 and 4-gram, and also for longest common subsequence co-occurrences. We showed three of the ROUGE metrics in the experimental results: ROUGE-1 (unigram-based), ROUGE-2 (bigram-based), and ROUGE-W (based on weighted longest common subsequence, weight=1.2). In order to truncate summaries longer than the length limit,

[1] In our pilot study, we can observe good performance when θ is in a wide range of [0.4, 0.8].

[2] We used ROUGEeval-1.4.2 in this study.

we used the “-l 100” option in ROUGE toolkit. We also used the “-m” option for word stemming.

4.2 Evaluation Results

4.2.1 System Comparison

In the experiments, the combination weight γ for the prior score is fixed at 0.15, as in the PageRank algorithm. Therefore, we have $\alpha+\beta=0.85$. Here, we use $\alpha/(\alpha+\beta)$ to indicate the relative contributions of the first two parts in Equations (9) and (10). We empirically set $\alpha/(\alpha+\beta)=0.4$ in the experiments. The proposed unified approach (i.e. UnifiedRank) is compared with a few baseline approaches and the top three participating systems.

The graph-based baselines for single-document summarization are described as follows:

BasicRank: This baseline approach adopts the basic PageRank algorithm to rank sentences based on all sentence relationships in a single document, similar to previous work (Mihalcea and Tarau, 2004).

PositionRank: This baseline approach improves the basic PageRank algorithm by using the position weight of a sentence as the prior score for the sentence. The position weight of a sentence is computed by using Equation (15).

CollabRank1: This baseline approach is the “UniformLink(Gold)” approach proposed in (Wan et al. 2007b). It uses a cluster of multiple documents to improve single document summarization by constructing a global affinity graph.

CollabRank2: This baseline approach is the “UnionLink(Gold)” approach proposed in (Wan et al. 2007b).

The graph-based baselines for multi-document summarization are described as follows:

BasicRank: This baseline approach adopts the basic PageRank algorithm to rank sentences based on all sentence relationships in document set. Both within-document and cross-document sentence relationships are used for constructing the affinity graph.

PositionRank: Similarly, this baseline approach improves the basic PageRank algorithm by using the position weight of a sentence as the prior score for the sentence.

TwoStageRank: This baseline approach leverages the results of single document summarization for multi-document summarization. It first computes the score of each sentence within each single document by using the PositionRank method, and then computes the final score of each sentence within the document set by considering the document-level sentence score as the prior score in the improved PageRank algorithm.

The top three systems are the systems with highest ROUGE scores, chosen from the participating systems on each task, respectively. Tables 2 and 3 show the comparison results for single-document summarization on DUC2001 and DUC2002, respectively. Tables 4 and 5 show the comparison results for multi-document summarization on DUC2001 and DUC2002, respectively. In the tables, SystemX (e.g. System28, SystemN) represents one of the top performing systems. The systems are sorted by decreasing order of the ROUGE-1 scores.

For single-document summarization, the proposed UnifiedRank approach always outperforms the four graph-based baselines over all three metrics on both two datasets. The performance differences are all statistically significant by using t-test (p-value<0.05). The ROUGE-1 score of UnifiedRank is higher than that of the best participating systems and the ROUGE-2 and ROUGE-W scores of UnifiedRank are comparable to that of the best participating systems.

For multi-document summarization, the proposed UnifiedRank approach outperforms all the three graph-based baselines over all three metrics on the DUC2001 dataset, and it outperforms the three baselines over ROUGE-1 and ROUGE-W on the DUC2002 dataset. In particular, UnifiedRank can significantly outperform BasicRank and TwoStageRank over all three metrics on the DUC2001 dataset (t-test, p-value<0.05). Moreover, the ROUGE-1 and ROUGE-W scores of UnifiedRank are higher than that of the best participating systems and the ROUGE-2 score of UnifiedRank is comparable to that of the best participating systems.

The results demonstrate that the single-document and multi-document summarizations can benefit each other by making use of the mutual influences between the local saliency and global saliency of the sentences. Overall, the proposed unified graph-based approach is effective for both single document summarization and multi-document summarization. However, the performance improvement for single-document summarization is more significant than that for multi-document summarization, which shows that the global information in a document set is very beneficial to summarization of each single document in the document set.

System	ROUGE-1	ROUGE-2	ROUGE-W
UnifiedRank	**0.45377**	**0.17649**	**0.14328**
CollabRank2	0.44038	0.16229	0.13678
CollabRank1	0.43890	0.16213	0.13676
PositionRank	0.43596	0.15936	0.13684
BasicRank	0.43407	0.15696	0.13629

Table 2. Comparison results for single-document summarization on DUC2001[3]

System	ROUGE-1	ROUGE-2	ROUGE-W
UnifiedRank	**0.48478**	0.21462	0.16877
System28	0.48049	**0.22832**	**0.17073**
System21	0.47754	0.22273	0.16814
CollabRank1	0.47187	0.20102	0.16318
CollabRank2	0.47028	0.20046	0.16260
PositionRank	0.46618	0.19853	0.16180
System31	0.46506	0.20392	0.16162
BasicRank	0.46261	0.19457	0.16018

Table 3. Comparison results for single-document summarization on DUC2002

System	ROUGE-1	ROUGE-2	ROUGE-W
UnifiedRank	**0.36360**	0.06496	**0.10950**
PositionRank	0.35733	0.06092	0.10798
BasicRank	0.35527	0.05608	0.10641
TwoStageRank	0.35221	0.05500	0.10515
SystemN	0.33910	0.06853	0.10240
SystemP	0.33332	0.06651	0.10068
SystemT	0.33029	**0.07862**	0.10215

Table 4. Comparison results for multi-document summarization on DUC2001

System	ROUGE-1	ROUGE-2	ROUGE-W
UnifiedRank	**0.38343**	0.07855	**0.12341**
PositionRank	0.38056	0.08238	0.12292
TwoStageRank	0.37972	0.08166	0.12261
BasicRank	0.37595	**0.08304**	0.12173
System26	0.35151	0.07642	0.11448
System19	0.34504	0.07936	0.11332
System28	0.34355	0.07521	0.10956

Table 5. Comparison results for multi-document summarization on DUC2002

4.2.2 *Influences of Combination Weight*

In the above experiments, the relative contributions from the first two parts in Equations (9) and (10) are empirically set as $\alpha/(\alpha+\beta)$=0.4. In this section, we investigate how the relative contributions influence the summarization performance by varying $\alpha/(\alpha+\beta)$ from 0 to 1. A small value of $\alpha/(\alpha+\beta)$ indicates that the contribution from the same kind of saliency scores of the sentences is less important than the contribution from the different kind of saliency scores of the sentences, and vice versa. Figures 1-8 show the ROUGE-1 and ROUGE-W curves for single-document summarization and multi-document summarization on DUC2001 and DUC2002, respectively.

For single document summarization, very small value or very large value for $\alpha/(\alpha+\beta)$ will lower the summarization performance values on the two datasets. The results demonstrate that both the two kinds of contributions are important to the final performance of single document summarization.

For multi-document summarization, a relatively large value (≥0.4) for $\alpha/(\alpha+\beta)$ will lead to relatively high performance values on the DUC2001 dataset, but a very large value for $\alpha/(\alpha+\beta)$ will decrease the performance values. On the DUC2002 dataset, a relatively small value (≤0.4) will lead to relatively high performance values, but a very small value for $\alpha/(\alpha+\beta)$ will decrease the performance values. Though the trends of the curves on the DUC2001 and DUC2002 datasets are not very consistent with each other, the results show that both the two kinds of contributions are beneficial to the final performance of multi-document summarization.

5 Conclusion and Future Work

In this study, we propose a novel unified approach to simultaneous single-document and multi-document summarization by making using of the mutual influences between the two tasks. Experimental results on the benchmark DUC datasets show the effectiveness of the proposed approach.

In future work, we will perform comprehensive experiments for topic-focused document

[3] The summarization results for participating systems on DUC2001 are incomplete.

summarizations to show the robustness of the proposed approach.

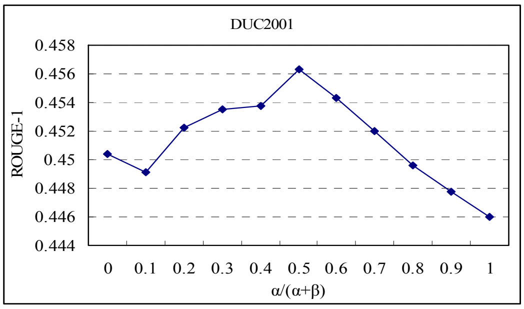


Figure 1. ROUGE-1 vs. combination weight for single-document summarization on DUC2001

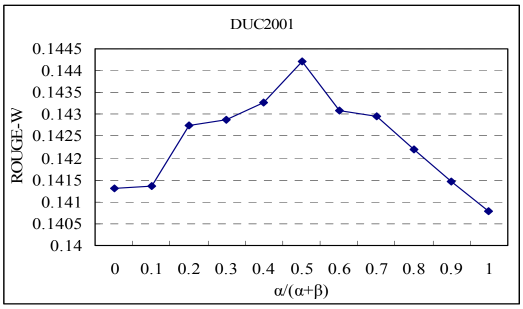


Figure 2. ROUGE-W vs. combination weight for single-document summarization on DUC2001

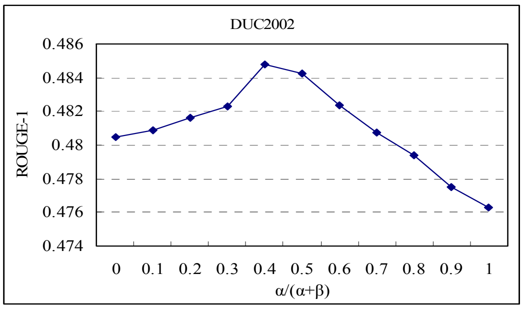


Figure 3. ROUGE-1 vs. combination weight for single-document summarization on DUC2002

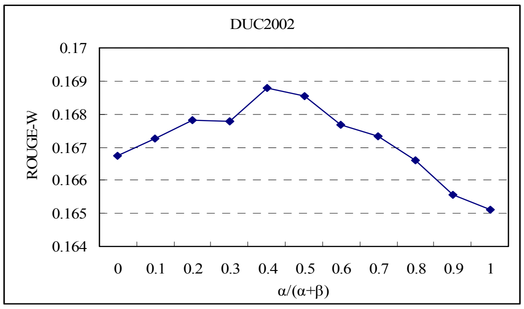


Figure 4. ROUGE-W vs. combination weight for single-document summarization on DUC2002

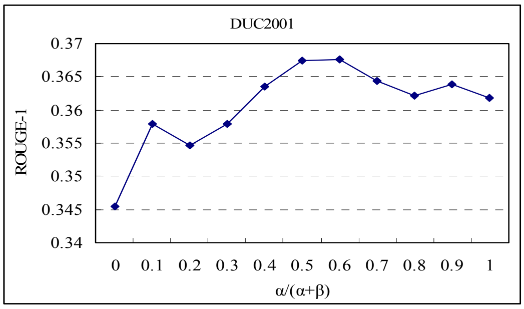


Figure 5. ROUGE-1 vs. combination weight for multi-document summarization on DUC2001

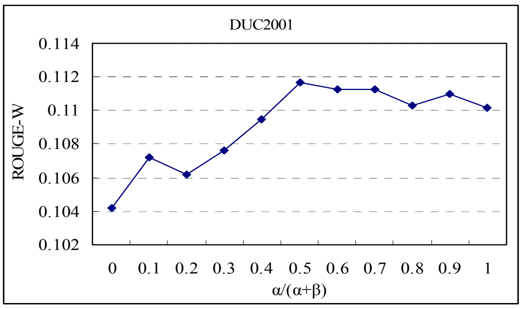


Figure 6. ROUGE-W vs. combination weight for multi-document summarization on DUC2001

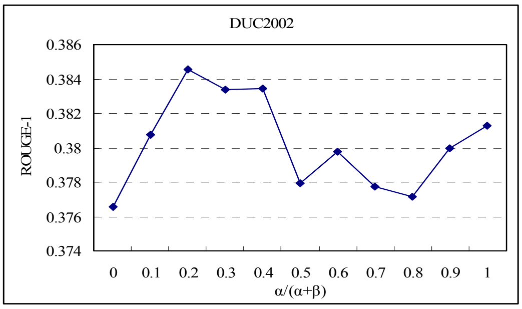


Figure 7. ROUGE-1 vs. combination weight for multi-document summarization on DUC2002

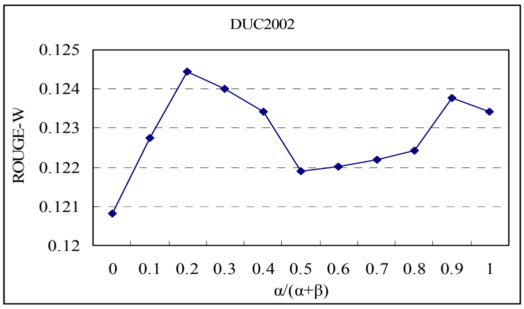


Figure 8. ROUGE-W vs. combination weight for multi-document summarization on DUC2002

Acknowledgments

This work was supported by NSFC (60873155), Beijing Nova Program (2008B03) and NCET (NCET-08-0006).

References

J. Carbonell, J. Goldstein. 1998. The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of SIGIR1998*, 335-336.

J. M. Conroy, D. P. O'Leary. 2001. Text Summarization via Hidden Markov Models. In *Proceedings of SIGIR2001*, 406-407.

H. Daumé and D. Marcu. 2006. Bayesian query-focused summarization. In *Proceedings of ACL-06*.

H. P. Edmundson. 1969. New Methods in Automatic Abstracting. *Journal of the Association for computing Machinery*, 16(2): 264-285.

G. ErKan, D. R. Radev. 2004. LexPageRank: Prestige in Multi-Document Text Summarization. In *Proceedings of EMNLP2004*.

B. Hachey. 2009. Multi-document summarisation using generic relation extraction. *In Proceedings of EMNLP2009*.

S. Harabagiu and F. Lacatusu. 2005. Topic themes for multi-document summarization. In *Proceedings of SIGIR-05*.

E. Hovy, C. Y. Lin. 1997. Automated Text Summarization in SUMMARIST. In *Proceeding of ACL'1997/EACL'1997 Worshop on Intelligent Scalable Text Summarization*.

J. Kupiec, J. Pedersen, F. Chen. 1995. A.Trainable Document Summarizer. In *Proceedings of SIGIR1995*, 68-73.

W. Li, F. Wei, Q. Lu and Y. He. 2008. PNR2: ranking sentences with positive and negative reinforcement for query-oriented update summarization. In *Proceedings of COLING-08*.

L. Li, K. Zhou, G.-R. Xue, H. Zha, Y. Yu. 2009. Enhancing diversity, coverage and balance for summarization through structure learning. In *Proceedings of WWW-09*.

C..-Y. Lin and E.. H. Hovy. 2002. From Single to Multi-document Summarization: A Prototype System and its Evaluation. In *Proceedings of ACL-02*.

C.-Y. Lin and E.H. Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *Proceedings of HLT-NAACL -03*.

H. P. Luhn. 1969. The Automatic Creation of literature Abstracts. *IBM Journal of Research and Development*, 2(2).

R. Mihalcea, P. Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of EMNLP2004*.

R. Mihalcea and P. Tarau. 2005. A language independent algorithm for single and multiple document summarization. In *Proceedings of IJCNLP-05*.

V. Nastase. 2008. Topic-driven multi-document summarization with encyclopedic knowledge and spreading activation. In *Proceedings of EMNLP-08*.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1998. The pagerank citation ranking: Bringing order to the web. *Technical report*, Stanford Digital Libraries.

D. R. Radev, H. Y. Jing, M. Stys and D. Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing and Management*, 40: 919-938.

F. Schilder and R. Kondadadi. 2008. FastSum: fast and accurate query-based multi-document summarization. In *Proceedings of ACL-08: HLT*.

D. Shen, J.-T. Sun, H. Li, Q. Yang, and Z. Chen. 2007. Document Summarization using Conditional Random Fields. In *Proceedings of IJCAI2007*.

X. Wan. 2008. Using Only Cross-Document Relationships for Both Generic and Topic-Focused Multi-Document Summarizations. *Information Retrieva*l, 11(1): 25-49.

X. Wan and J. Yang. 2008. Multi-document summarization using cluster-based link analysis. In *Proceedings of SIGIR-08*.

X. Wan, J. Yang and J. Xiao. 2007a. Towards an Iterative Reinforcement Approach for Simultaneous Document Summarization and Keyword Extraction. In *Proceedings of ACL2007*.

X. Wan, J. Yang and J. Xiao. 2007b. CollabSum: Exploiting Multiple Document Clustering for Collaborative Single Document Summarizations. In *Proceedings of SIGIR2007*.

X. Wan, J. Yang and J. Xiao. 2007c. Manifold-ranking based topic-focused multi-document summarization. In *Proceedings of IJCAI-07*.

F. Wei, W. Li, Q. Lu and Y. He. 2008. Query-sensitive mutual reinforcement chain and its application in query-oriented multi-document summarization. In *Proceedings of SIGIR-08*.

K.-F. Wong, M. Wu and W. Li. 2008. Extractive summarization using supervised and semi-supervised learning. In *Proceedings of COLING-08*.

H. Y. Zha. 2002. Generic Summarization and Keyphrase Extraction Using Mutual Reinforcement Principle and Sentence Clustering. In *Proceedings of SIGIR2002*, 113-120.

"Got You!": Automatic Vandalism Detection in Wikipedia with Web-based Shallow Syntactic-Semantic Modeling

William Yang Wang and Kathleen R. McKeown
Department of Computer Science
Columbia University
yw2347@columbia.edu kathy@cs.columbia.edu

Abstract

Discriminating vandalism edits from non-vandalism edits in Wikipedia is a challenging task, as ill-intentioned edits can include a variety of content and be expressed in many different forms and styles. Previous studies are limited to rule-based methods and learning based on lexical features, lacking in linguistic analysis. In this paper, we propose a novel Web-based shallow syntactic-semantic modeling method, which utilizes Web search results as resource and trains topic-specific n-tag and syntactic n-gram language models to detect vandalism. By combining basic task-specific and lexical features, we have achieved high F-measures using logistic boosting and logistic model trees classifiers, surpassing the results reported by major Wikipedia vandalism detection systems.

1 Introduction

Online open collaboration systems are becoming a major means of information sharing on the Web. With millions of articles from millions of resources edited by millions of people, Wikipedia is a pioneer in the fast growing, online knowledge collaboration era. Anyone who has Internet access can visit, edit and delete Wikipedia articles without authentication.

A primary threat to this convenience, however, is vandalism, which has become one of Wikipedia's biggest concerns (Geiger, 2010). To date, automatic countermeasures mainly involve rule-based approaches and these are not very effective. Therefore, Wikipedia volunteers have to spend a large amount of time identifying vandalized articles manually, rather than spending time contributing content to the articles. Hence, there is a need for more effective approaches to automatic vandalism detection.

In contrast to spam detection tasks, where a full spam message, which is typically 4K Bytes (Rigoutsos and Huynh, 2004), can be sampled and analyzed (Itakura and Clarke, 2009), Wikipedia vandals typically change only a small number of words or sentences in the targeted article. In our preliminary corpus (Potthast et al., 2007), we find the average size of 201 vandalized texts to be only 1K Byte. This leaves very few clues for vandalism modeling. The question we address in this paper is: given such limited information, how can we better understand and model Wikipedia vandalism?

Our proposed approach establishes a novel classification framework, aiming at capturing vandalism through an emphasis on shallow syntactic and semantic modeling. In contrast to previous work, we recognize the significance of natural language modeling techniques for Wikipedia vandalism detection and utilize Web search results to construct our shallow syntactic and semantic models. We first construct a baseline model that captures task-specific clues and lexical features that have been used in earlier work (Potthast et al., 2008; Smets et al., 2008) augmenting these with shallow syntactic and semantic features. Our main contributions are:

- Improvement over previous modeling methods with three novel lexical features
- Using Web search results as training data for syntactic and semantic modeling
- Building topic-specific n-tag syntax models and syntactic n-gram models for shallow syntactic and semantic modeling

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1146–1154,
Beijing, August 2010

2 Related Work

So far, the primary method for automatic vandalism detection in Wikipedia relies on rule-based bots. In recent years, however, with the rise of statistical machine learning, researchers have begun to treat Wikipedia vandalism detection task as a classification task. To the best of our knowledge, we are among the first to consider the shallow syntactic and semantic modeling using Natural Language Processing (NLP) techniques, utilizing the Web as corpus to detect vandalism.

ClueBot (Carter, 2007) is one of the most active bots fighting vandalism in Wikipedia. It keeps track of the IP of blocked users and uses simple regular expressions to keep Wikipedia vandalism free. A distinct advantage of rule-based bots is that they have very high precision. However they suffer from fixed-size knowledge bases and use only rigid rules. Therefore, their average recall is not very high and they can be easily fooled by unseen vandalism patterns. According to Smets et al., (2008) and Potthast et al., (2008), rule-based bots have a perfect precision of 1 and a recall of around 0.3.

The Wikipedia vandalism detection research community began to concentrate on the machine learning approaches in the past two years. Smets et al. (2008) wrapped all the content in *diff* text into a bag of words, disregarding grammar and word order. They used Na ïve Bayes as the classification algorithm. Compared to rule-based methods, they show an average precision of 0.59 but are able to reach a recall of 0.37. Though they are among the first to try machine learning approaches, the features in their study are the most straightforward set of features. Clearly, there is still room for improvement.

More recently, Itakura and Clarke (2009) have proposed a novel method using Dynamic Markov Compression (DMC). They model their approach after the successful use of DMC in Web and Mail Spam detection (Bratko et al., 2006). The reported average precision is 0.75 and average recall is 0.73.

To the best of our knowledge, Potthast et al., (2008) report the best result so far for Wikipedia vandalism detection. They craft a feature set that consists of interesting task-specific features. For example, they monitor the number of previously submitted edits from the same author or IP, which is a good feature to model author contribution. Their other contributions are the use of a logistic regression classifier, as well as the use of lexical features. They successfully demonstrate the use of lexical features like vulgarism frequency. Using all features, they reach an average precision of 0.83 and recall of 0.77.

In addition to previous work on vandalism detection, there is also earlier work using the web for modeling. Biadsy et al. (2008) extract patterns in Wikipedia to generate biographies automatically. In their experiment, they show that when using Wikipedia as the only resource for extracting named entities and corresponding collocational patterns, although the precision is typically high, recall can be very low. For that reason, they choose to use Google to retrieve training data from the Web. In our approach, instead of using Wikipedia edits and historical revisions, we also select the Web as a resource to train our shallow syntactic and semantic models.

3 Analysis of Types of Vandalism

In order to better understand the characteristics of vandalism cases in Wikipedia, we manually analyzed 201 vandalism edits in the training set of our preliminary corpus. In order to concentrate on textual vandalism detection, we did not take into account the cases where vandals hack the image, audio or other multimedia resources contained in the Wikipedia edit.

We found three main types of vandalism, which are shown in Table 1 along with corresponding examples. These examples contain both the title of the edit and a snippet of the *diff*-ed content of vandalism, which is the textual difference between the old revision and the new revision, derived through the standard *diff* algorithm (Heckel, 1978).

- **Lexically ill-formed**
  This is the most common type of vandalism in Wikipedia. Like other online vandalism acts, many vandalism cases in Wikipedia involve ill-intentioned or ill-formed words such as vulgarisms, invalid letter sequences, punctuation misuse and Web slang. An interesting observation is that vandals almost never add emoticons in Wikipedia. For the first example in

Vandalism Types	**Examples**
Lexically ill-formed	**Edit Title**: *IPod* shit!!!!!!!!!!!!!!!!!!!!!!
Syntactically ill-formed	**Edit Title**: *Rock music* DOWN WITH SOCIETY MADDISON STREET RIOT FOREVER.
	Edit Title: *Vietnam War* Crabinarah sucks dont buy it
Lexically + syntactically well-formed, semantically ill-intentioned	**Edit Title**: *Global Warming* Another popular theory involving global warming is the concept that global warming is not caused by greenhouse gases. The theory is that Carlos Boozer is the one preventing the infrared heat from escaping the atmosphere. Therefore, the Golden State Warriors will win next season.
	Edit Title: *Harry Potter* Harry Potter is a teenage boy who likes to smoke crack with his buds. They also run an illegal smuggling business to their headmaster dumbledore. He is dumb!

Table 1: Vandalism Types and Examples

Table 1, vulgarism and punctuation misuse are observed.

- **Syntactically ill-formed**

Most vandalism cases that are lexically ill-intentioned tend to be syntactically ill-formed as well. It is not easy to capture these cases by solely relying on lexical knowledge or rule-based dictionaries and it is also very expensive to update dictionaries and rules manually. Therefore, we think that is crucial to incorporate more syntactic cues in the feature set in order to improve performance. Moreover, there are also some cases where an edit could be lexically well-intentioned, yet syntactically ill-formed. The first example of syntactic ill-formed in Table 1 is of this kind.

Feature Sets	**Features**

Task-specific	Number of Revisions;
	Revisions Size Ratio;

Lexical	Vulgarism; Web Slang;
	Punctuation Misuse;
	Comment Cue Words;

Syntactic	Normalized Topic-specific N-tag Log Likelihood and Perplexity
Semantic	Normalized Topic-specific Syntactic N-gram Log Likelihood and Perplexity

Table 2: Feature Sets and Corresponding Features of Our Vandalism Detection System

- **Lexically and syntactically well formed, but semantically ill-intentioned**

This is the trickiest type of vandalism to identify. Vandals of this kind might have good knowledge of the rule-based vandalism detecting bots. Usually, this type of vandalism involves off-topic comments, inserted biased opinions, unconfirmed information and lobbying using very subjective comments. However, a common characteristic of all vandalism in this category is that it is free of both lexical and syntactic errors. Consider the first example of semantic vandalism in Table 1 with edit title "Global Warming": while the first sentence for that edit seems to be fairly normal (the author tries to claim another explanation of the global warming effect), the second sentence makes a sudden transition from the previous topic to mention a basketball star and makes a ridiculous conclusion in the last sentence.

In this work, we realize the importance of incorporating NLP techniques to tackle all the above types of vandalism, and our focus is on the syntactically ill-formed and semantically ill-intentioned types that could not be detected by rule-based systems and straightforward lexical features.

4 Our System

We propose a shallow syntactic-semantic focused classification approach for vandalism detection (Table 2). In contrast to previous work, our approach concentrates on the aspect of using natural language techniques to model vandalism. Our shallow syntactic and semantic modeling approaches extend the traditional n-gram language modeling method with topic-specific n-tag (Collins et al., 2005) syntax models and topic-specific syntactic n-gram semantic models. Moreover, in the Wikipedia vandalism detection task, since we do not have a sufficient amount of training data to model the topic of each edit, we propose the idea of using the Web as corpus by retrieving search engine results to learn our topic-specific n-tag syntax and syntactic n-gram semantic models. The difference between our syntactic and semantic modeling is that n-tag syntax models only model the order of sentence constituents, disregarding the corresponding words. Conversely, for our syntactic n-gram models, we do keep track of words together with their POS tags and model both the word and syntactic compositions as a sequence. The detail of our shallow syntactic-semantic modeling method will be described in subsection 4.4.

We use our shallow syntactic-semantic model to augment our base model, which builds on early work. For example, when building one of our task-specific features, we extract the name of the author of this revision to query Wikipedia about the historical behavior of this author. This kind of task-specific global feature tends to be very informative and thus forms the basis of our system. For lexical level features, we count vulgarism frequencies and also introduce three new lexical features: Web slang, punctuation misuse and comment cue words, all of which will be described in detail in 4.2 and 4.3.

4.1 Problem Representation

The vandalism detection task can be formulated as the following problem. Let's assume we have a vandalism corpus *C*, which contains a set of Wikipedia edits *S*. A Wikipedia edit is denoted as $\boldsymbol{e_i}$. In our case, we have $S = \{\boldsymbol{e_1}, \boldsymbol{e_2}\ldots,\boldsymbol{e}_n\}$. Each edit ***e*** has two consecutive revisions (an old revision R_{old} and a new revision R_{new}) that are unique in the entire data set. We write that $\boldsymbol{e} = \{R_{old}, R_{new}\}$. With the use of the standard *diff* algorithm, we can produce a text R_{diff}, showing the difference between these two revisions, so that $\boldsymbol{e} = \{R_{old}, R_{new}, R_{diff}\}$. Our task is: given S, to extract features from edit $\boldsymbol{e} \in S$ and train a logistic boosting classifier. On receiving an edit ***e*** from the test set, the classifier needs to decide whether this ***e*** is a vandalism edit or a non-vandalism edit. $\boldsymbol{e} \rightarrow \{1,0\}$.

4.2 Basic Task-specific and Lexical Features

Task-specific features are domain-dependent and are therefore unique in this Wikipedia vandalism detection task. In this work, we pick two task-specific features and one lexical feature that proved effective in previous studies.

- **Number of Revisions**
  This is a very simple but effective feature that is used by many studies (Wilkinson and Huberman, 2007; Adler et al., 2008; Stein and Hess, 2007). By extracting the author name for the new revision R_{new}, we can easily query Wikipedia and count how many revisions the author has modified in the history.

- **Revision Size Ratio**
  Revision size ratio measures the size of the new revision versus the size of the old revision in an edit. This measure is an indication of how much information is gained or lost in the new revision R_{new}, compared to the old revision R_{old}, and can be expressed as:

$$\text{RevRatio}(\boldsymbol{e}) = \frac{\sum_{w \,\epsilon\, R\ new} \text{Count (w)}}{\sum_{w \,\epsilon\, R\ old} \text{Count (w)}}$$

  where W represents any word token of a revision.

- **Vulgarism Frequency**
  Revision size ratio measures the size of the new revision versus the Vulgarism frequency was first introduced by Potthast et al. (2008). However, note that not all vulgarism words should be considered as vandalism and sometime even the Wikipedia edit's title and content themselves contain vulgarism words.

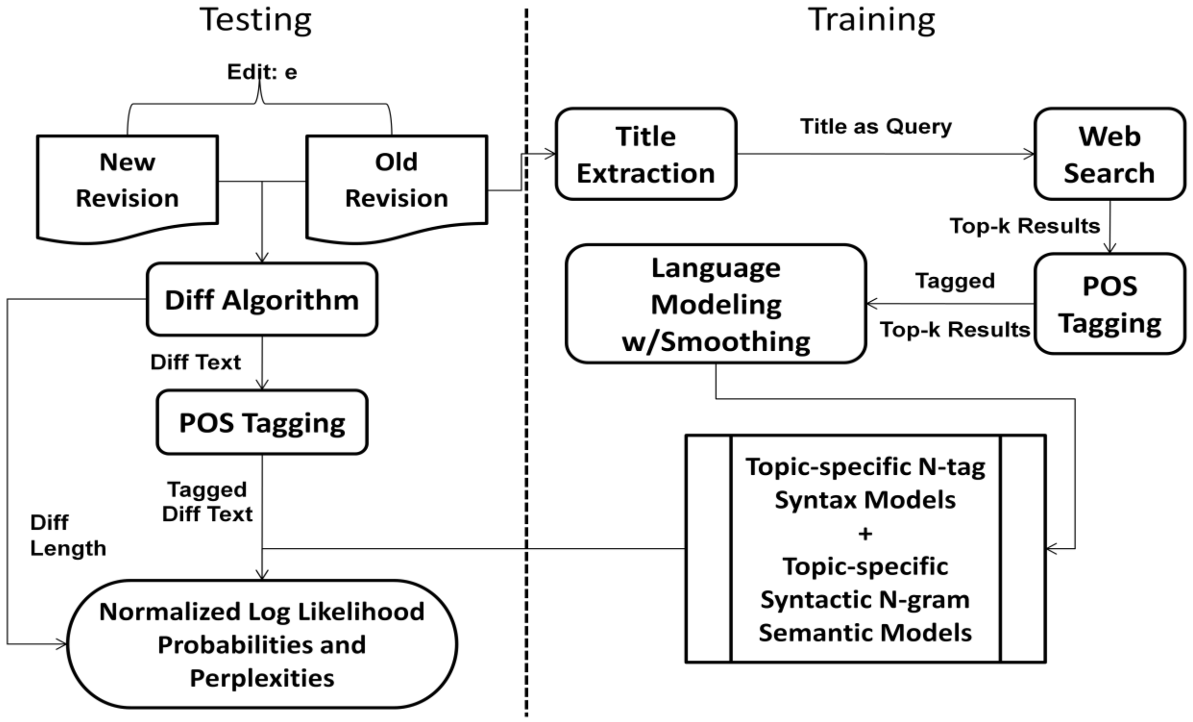


Figure 1. Topic-specific N-tag Syntax Models and Syntactical N-gram for Syntactical and Semantic Modeling

For each *diff* text in an edit ***e***, we count the total number of appearances of vulgarism words v where v is in our vulgarism dictionary[1].

$$\text{VulFreq}(\boldsymbol{e}) = \sum_{v \in \text{Rdiff}} \text{Count}(v)$$

4.3 Novel Lexical Features

In addition to previous lexical features, we propose three novel lexical features in this paper: Web slang frequency, punctuation misuse, and comment cue words frequency.

- **Web Slang and Punctuation Misuse**

  Since Wikipedia is an open Web application, vandalism also contains a fair amount of Web slang, such as, “haha”, “LOL” and “OMG”. We use the same method as above to calculate Web slang frequency, using a Web slang dictionary[2]. In vandalism edits, many vandalism edits also contain punctuation misuse, for example, “!!!” and “???”. However, we have not observed a significant amount of emoticons in the vandalism edits. Based on this, we only keep track of Web slang frequency and the occurrence of punctuation misuse.

- **Comment Cue Words**

  Upon committing each new revision in Wikipedia, the author is required to enter some comments describing the change. Well-intentioned Wikipedia contributors consistently use these comments to explain the motivation for their changes. For example, common non-vandalism edits may contain cue words and phrases like “edit revised, page changed, item cleaned up, link repaired or delinked”. In contrast, vandals almost never take their time to add these kinds of comments. We can measure this phenomenon by counting the frequency of comment cue words.

[1] http://www.noswearing.com/dictionary
[2] http://www.noslang.com/dictionary/full

(a)
Rock/NNP and/CC roll/NN -LRB-/-LRB- also/RB spelled/VBD Rock/NNP 'n'/CC Roll/NNP
(b)
NNP CC NN -LRB- RB VBD NNP CC NNP
(c)
Rock/NNP !/. !/. !/. and/CC roll/VB you/PRP !/. !/. !/.
(d)
NNP . . . CC VB PRP . . .

Figure 2. Topic-specific N-tag and Syntactic N-gram modeling for the edit "Rock and Roll" in Wikipedia (a) The Web-derived POS tagged sequence (b) The Web-derived POS tag-only sequence (c) A POS tagged vandalism *diff* text R_{diff} (d) A POS tag-only vandalism R_{diff}

4.4 Topic-specific N-tag Syntax Models and Syntactic N-grams for Shallow Syntactic and Semantic Modeling

In Figure 1, we present the overview of our approach, which uses Web-trained topic-specific training for both: (1) n-tag syntax models for shallow syntactic modeling and (2) syntactic n-gram models for shallow semantic modeling.

For each Wikipedia edit, we consider its title as an approximate semantic representation, using it as a query to build topic-specific models. In addition, we also use the title information to model the syntax of this topic.

Given R_{diff}, we produce the syntactic version of the *diff*-ed text using a probabilistic POS tagger (Toutanova and Manning, 2000; Toutanova et al., 2003). The edit title is extracted from the corpus (either R_{new} or R_{old}) and is used to query multiple Web search engines in order to collect the n-tag and n-gram training data from the top-*k* results. Before we start training language models, we tag the top-*k* results using the POS tagger. Note that when modeling n-tag syntax models, it is necessary to remove all the words. With the POS-only sequences, we train topic-specific n-tag models to describe the syntax of normal text on the same topic associated with this edit. With the original tagged sequences, we train syntactic n-gram models to represent the semantics of the normal text of this edit.

After completing the training stage, we send the test segment (i.e. the *diff*-ed text sequence) to both the learned n-tag syntax models and the learned syntactic n-gram models. For the n-tag syntax model, we submit the POS tag-only version of the segment. For the syntactic n-gram model, we submit a version of the segment where each original word is associated with its POS-tag. In both cases we compute the log-likelihood and the perplexity of the segment.

Finally, we normalize the log likelihood and perplexity scores by dividing them by the length of R_{diff}, as this length varies substantially from one edit to another. [3] We expect an edit that has low log likelihood probability and perplexity to be vandalism, and it is very likely to be unrelated to the syntax and semantic of the normal text of this Wikipedia edit. In the end, the normalized log probability and perplexity scores will be incorporated into our back-end classifier with all task-specific and lexical features.

Web as Corpus: In this work, we leverage Web search results to train the syntax and semantic models. This is based on the assumption that the Web itself is a large corpus and Web search results can be a good training set to approximate the semantics and syntax of the query.

Topic-specific Modeling: We introduce a topic-specific modeling method that treats every edit in Wikipedia as a unique topic. We think that the title of each Wikipedia edit is an approximation of the topic of the edit, so we extract the title of each edit and use it as keywords to retrieve training data for our shallow syntactic and semantic modeling.

Topic-specific N-tag and Syntactic N-gram: In our novel approach, we tag all the top-*k* query results and *diff* text with a probabilistic POS tagger in both the training and test set of the vandalism corpus. Figure 2(a) is an example of a POS-tagged sequence in a top-*k* query result.

For shallow syntactic modeling, we use an n-tag modeling method (Collins et al., 2005). Given a tagged sequence, we remove all the words and only keep track of its POS tags: tag_{i-2} tag_{i-1}

[3] Although we have experimented with using the length of R_{diff} as a potential feature, it does not appear to be a good indicator of vandalism.

tag_i. This is similar to n-gram language modeling, but instead, we model the syntax using POS tags, rather than its words. In this example, we can use the system in Figure 2 (b) to train an n-tag syntactic model and use the one in Figure 2 (d) to test. As we see, for this test segment, it belongs to the vandalism class and has very different syntax from the n-tag model. Therefore, the normalized log likelihood outcome from the n-tag model is very low.

In order to model semantics, we use an improved version of the n-gram language modeling method. Instead of only counting $word_{i-2}$ $word_{i-1}$ $word_i$, we model composite tag/word feature, e.g. $tag_{i-2}word_{i-2}$ $tag_{i-1}word_{i-1}$ tag_iword_i. This syntactic n-gram modeling method has been successfully applied to the task of automatic speech recognition (Collins et al., 2005). In the example in Figure 2, the vandalism *diff* text will probably score low, because although it shares an overlap bigram "*and roll*" with the phrase "*rock and roll*" in training text, once we apply the shallow syntactic n-gram modeling method, the POS tag bigram "*and/CC roll/VB*" in *diff* text will be distinguished from the "*and/CC roll/NN*" or "*and/CC roll/NNP*" in the training data.

5 Experiments

To evaluate the effectiveness of our approach, we first run experiments on a preliminary corpus that is also used by previous studies and compare the results. Then, we conduct a second experiment on a larger corpus and analyze in detail the features of our system.

5.1 Experiment Setup

In our experiments, we use a Wikipedia vandalism detection corpus (Potthast et al., 2007) as a preliminary corpus. The preliminary corpus contains 940 human-assessed edits from which 301 edits are classified as vandalism. We split the corpus and keep a held-out 100 edits for each class in testing and use the rest for training. In the second experiment, we adopt a larger corpus (Potthast et al., 2010) that contains 15,000 edits with 944 marked as vandalism. The split is 300 edits for each class in held-out testing and the rest used for training. In the description of the second corpus, each edit has been reviewed by at least 3 and up to 15 annotators. If more than 2/3 of the annotators agree on a given edit, then the edit is tagged as one of our target classes. Only 11 cases are reported where annotators fail to form a majority inter-labeler agreement and in those cases, the class is decided by corpus authors arbitrarily.

In our implementation, the Yahoo![4] search engine and Bing[5] search engine are the source for collecting top-*k* results for topic-specific n-gram training data, because Google has a daily query limit. We retrieve top-100 results from Yahoo!, and combine them with the top-50 results from Bing.

For POS tagging, we use the Stanford POS Tagger (Toutanova and Manning, 2000; Toutanova et al., 2003) with its attached wsj3t0-18-bidirectional model trained from the Wall Street Journal corpus. For both shallow syntactic and semantic modeling, we train topic-specific trigram language models on each edit using the SRILM toolkit (Stolcke, 2002).

In this classification task, we used two logistic classification methods that haven't been used before in vandalism detection. Logistic model trees (Landwehr et al., 2005) combine tree induction with linear modeling. The idea is to use the logistic regression to select attributes and build logistic regression at the leaves by incrementally refining those constructed at higher levels in the tree. The second method we used, logistic boosting (Friedman et al., 2000), improves logistic regression with boosting. It works by applying the classification algorithm to reweighted versions of the data and then taking a weighted majority vote of the sequence of classifiers thus produced.

5.2 Preliminary Experiment

In the preliminary experiment, we tried logistic boosting classifiers and logistic model trees as classifiers with 10-fold cross validation. The rule-based method, ClueBot, is our baseline.

We also implemented another baseline system, using the bag of words (BoW) and Naive Bayes method (Smets et al., 2008) and the same toolkit (McCallum, 1996) that Smets et al. used. Then, we compare our result with Potthast et al. (2008), who used the same corpus as us.

[4] http://www.yahoo.com

[5] http://www.bing.com

Systems	**Recall**	**Precision**	**F1**
ClueBot	0.27	**1**	0.43
BoW + Na ïve Bayes	0.75	0.74	0.75
Potthast et. al., 2008	0.77	0.83	0.80
Task-specific +Lexical (LMT)	0.87	0.87	0.87
Task-specific +Lexical (LB)	0.92	0.91	0.91
Our System (LMT)	0.89	0.89	0.89
Our System (LB)	**0.95**	0.95	**0.95**

Table 3: Preliminary Experiment Results; The acronyms: BoW: Bag of Words, LMT: Logistic Model Trees, LB: Logistic Boosting, Task-specific + Lexical: features in section 4.1 and 4.2

As we can see in Table 3, the ClueBot has a F-score (F1) of 0.43. The BoW + Na ïve Bayes approach improved the result and reached an F1 of 0.75. Compared to these results, the system of Potthast et al. (2008) is still better and has a F1 of 0.80.

For the results of our system, LMT gives us a 0.89 F1 and LogitBoost (LB) gives a 0.95 F1. A significant F1 improvement of 15% was achieved in comparison to the previous study (Potthast et al., 2008). Another finding is that we find our shallow syntactic-semantic modeling method improves 2-4% over our task-specific and lexical features.

5.3 Results and Analysis

In the second experiment, a notable difference from the preliminary evaluation is that we have an unbalanced data problem. So, we use random down-sampling method to resample the majority class into balanced classes in the training stage. Then, we also use the two classifiers with 10-fold cross validation.

The F1 result reported by our BoW + Na ïve Bayes baseline is 0.68. Next, we test our task-specific and lexical features that specified in section 4.1 and 4.2. The best result is a F1 of 0.82, using logistic boosting. Finally, with our topic-specific shallow syntactic and semantic modeling features, we have a precision of 0.86, a recall of 0.85 and F1 of 0.85.

Features	**Recall**	**Precision**	**F1**
BoW + Na ïve Bayes	0.68	0.68	0.68
Task-specific (LMT)	0.81	0.80	0.80
Task-specific +Lexical(LMT)	0.81	0.81	0.81
Our System (LMT)	0.84	0.83	0.83
Task-specific (LB)	0.81	0.80	0.80
Task-specific + Lexical (LB)	0.83	0.82	0.82
Our System (LB)	**0.86**	**0.85**	**0.85**

Table 4: Second Experiment Results

Though we are surprised to see the overall F1 for the second experiment are not as high as the first one, we do see that the topic-specific shallow syntactic and semantic modeling methods play an important role in improving the result.

Looking back at the related work we mentioned in section 2, though we use newer data sets, our overall results still seem to surpass major vandalism detection systems.

6 Conclusion and Future Works

We have described a practical classification framework for detecting Wikipedia vandalism using NLP techniques and shown that it outperforms rule-based methods and other major machine learning approaches that are previously applied in the task.

In future work, we would like to investigate deeper syntactic and semantic cues to vandalism. We hope to improve our models using shallow parsing and full parse trees. We may also try lexical chaining to model the internal semantic links within each edit.

Acknowledgements

The authors are grateful to Julia Hirschberg, Yves Petinot, Fadi Biadsy, Mukund Jha, Weiyun Ma, and the anonymous reviewers for useful feedback. We thank Potthast et al. for the Wikipedia vandalism detection corpora.

References

Adler, B. Thomas, Luca de Alfaro, Ian Pye and Vishwanath Raman. 2008. Measuring Author Contributions to the Wikipedia. In *Proc. of the ACM 2008 International Symposium on Wikis*.

Biadsy, Fadi, Julia Hirschberg, and Elena Filatova. 2008. An Unsupervised Approach to Biography Production using Wikipedia. In *Proc. of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies,* pages 807–815.

Bratko, Andrej, Gordon V. Cormack, Bogdan Filipic,Thomas R. Lynam and Blaz Zupan. 2006. Spam Filtering Using Statistical Data Compression Models. *Journal of Machine Learning Research*, pages 7:2673-2698.

Collins, Michael, Brian Roark and Murat Saraclar. 2005. Discriminative Syntactic Language Modeling for Speech Recognition. In *Proc. of the 43rd Annual Meeting of the Association for Computational Linguistics*. pages 507–514.

Friedman, Jerome, Trevor Hastie and Robert Tibshirani. 2000. Additive Logistic Regression: a Statistical View of Boosting. *Annals of Statistics* 28(2), pages 337-407.

Geiger, R. Stuart. 2010. The Work of Sustaining Order in Wikipedia: The Banning of a Vandal. In *Proc. of the 2010 ACM Conference on Computer Supported Cooperative Work,* pages 117-126.

Heckel, Paul. 1978. A Technique for Isolating Differences Between Files. *Communications of the ACM,* pages 264–268

Itakura, Kelly Y. and Charles L. A. Clarke. 2009. Using Dynamic Markov Compression to Detect Vandalism in the Wikipedia. In *Proc. of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 822-823.

Landwehr, Niels, Mark Hall and Eibe Frank. 2005. Logistic Model Trees. *Machine Learning*, 59(1-2), pages 161–205.

McCallum, Andrew. 1996. Bow: a Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering.

Potthast, Martin, Benno Stein, and Robert Gerling. 2008. Automatic Vandalism Detection in Wikipedia. In *Proc. of the 30th European Conference on Information Retrieval, Lecture Notes in Computer Science,* pages 663-668.

Potthast, Martin and Robert Gerling. 2007. Wikipedia Vandalism Corpus WEBIS-VC07-11. *Web Technology & Information Systems Group, Bauhaus University Weimar.*

Potthast, Martin, Benno Stein and Teresa Holfeld. 2010. PAN Wikipedia Vandalism Training Corpus PAN-WVC-10. *Web Technology & Information Systems Group, Bauhaus University Weimar*.

Rigoutsos, Isidore and Tien Huynh. 2004. Chung-Kwei: a pattern-discovery-based system for the automatic identification of unsolicited e-mail messages (SPAM). In *Proc. of the First Conference on E-mail and Anti-Spam.*

Smets, Koen, Bart Goethals and Brigitte Verdonk. 2008. Automatic Vandalism Detection in Wikipedia: Towards a Machine Learning Approach In *Proc. of AAAI '08, Workshop on Wikipedia and Artificial Intelligence*, pages 43-48.

Stein, Klaus and Claudia Hess. 2007. Does It Matter Who Contributes: a Study on Featured Articles in the German Wikipedia. In *Proc. of the ACM 18th Conference on Hypertext and Hypermedia*, pages 171–174.

Stolcke, Andreas. 2002. SRILM – an extensible language modeling toolkit. In *Proc. of the International Conference on Spoken Language Processing*, volume 2, pages 901–904.

Toutanova, Kristina and Christopher D. Manning. 2000. Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In *Proc. of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora,* pages 63-70.

Toutanova, Kristina, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of Human Language Technology Conference and the North American Chapter of the Association of Computational Linguistics Series,* pages 252-259.

Wilkinson,Dennis and Bernardo Huberman. 2007. Cooperation and Quality in Wikipedia. In *Proc. of the ACM 2007 International Symposium on Wikis*, pages 157–164.

Exploiting Salient Patterns for Question Detection and Question Retrieval in Community-based Question Answering

Kai Wang
Department of Computer Science
School of Computing
National University of Singapore
kwang@comp.nus.edu.sg

Tat-Seng Chua
Department of Computer Science
School of Computing
National University of Singapore
chuats@comp.nus.edu.sg

Abstract

Question detection serves great purposes in the cQA question retrieval task. While detecting questions in standard language data corpus is relatively easy, it becomes a great challenge for online content. Online questions are usually long and informal, and standard features such as question mark or 5W1H words are likely to be absent. In this paper, we explore question characteristics in cQA services, and propose an automated approach to detect question sentences based on lexical and syntactic features. Our model is capable of handling informal online languages. The empirical evaluation results further demonstrate that our model significantly outperforms traditional methods in detecting online question sentences, and it considerably boosts the question retrieval performance in cQA.

1 Introduction

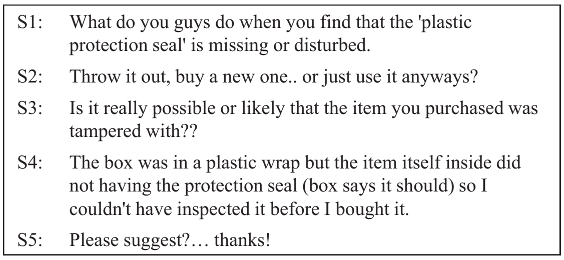
Community-based Question Answering services (cQA) such as Yahoo! Answers have emerged as popular means of information exchange on the web. They not only connect a network of people to freely ask and answer questions, but also allow information seekers to search for relevant historical questions in the cQA archive (Agichtein et al., 2008; Xue et al., 2008; Wang et al., 2009).

Many research works have been proposed to find similar questions in cQA. The state-of-the-art retrieval models include the vector space model (Duan et al., 2008), language model (Duan et al., 2008; Jeon et al., 2005), Okapi model (Jeon et al., 2005), translation model (Jeon et al., 2005; Riezler et al., 2007; Xue et al., 2008), and syntactic tree matching model(Wang et al., 2009). Although experimental studies in these works show that the proposed models are capable of improving question retrieval, they did not give clear explanation on which portion of the question that the user query is actually matched against. A question thread from cQA usually comprises several sub-questions conveying different information needs, and it is highly desirable to identify individual sub-questions and match each of them to the user query. Getting sub-questions clearly identified not only helps the retrieval system to match user query to the most desirable content but also improves the retrieval efficiency.

However, the detection of sub-question is non-trivial. Question sentences in cQA are usually mixed with various description sentences, and they usually employ informal languages, where standard features such as question mark or utterance are likely to be absent. As such, simple heuristics using question mark or 5W1H words (*who, what, where, why, how*) may become inadequate. The demand of special techniques in detecting question sentences online arises due to three particular reasons. First, the question mark could be missing at the end of a question[1], or might be used in cases other than questions such as "*Really bad toothache?*". Second, some questions such as "*I'd like to know the expense of removing wisdom teeth*" are expressed in a declarative form, which neither contains 5W1H words nor is neccessarily ended with "?". Third, some question-like sentences do not carry any actual information need, such as "*Please help me?*". Figure 1 illustrates an example of a question thread

[1] It is reported (Cong et al., 2008) that 30% of online questions do not end with question marks.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1155–1163,
Beijing, August 2010



S1: What do you guys do when you find that the 'plastic protection seal' is missing or disturbed.

S2: Throw it out, buy a new one.. or just use it anyways?

S3: Is it really possible or likely that the item you purchased was tampered with??

S4: The box was in a plastic wrap but the item itself inside did not having the protection seal (box says it should) so I couldn't have inspected it before I bought it.

S5: Please suggest?… thanks!

Figure 1: An example of a question thread extracted from Yahoo! Answers

from Yahoo! Answers, where sub-questions S1 and S2 are posted in non-standard forms, and S5 is merely a question-like simple sentence. To the best of our knowledge, none of the existing question retrieval systems are equipped with a comprehensive question detector module to handle various question forms online, and limited effort has been devoted to this direction.

In this paper, we extensively explore characteristics of questions in cQA, and propose a fully automated approach to detecting question sentences. In particular, we complement lexical patterns with syntactic patterns, and use them as features to train a classification model that is capable of handling informal online languages. To save human annotations, we further propose to employ one-class SVM algorithm for model learning, in which only positive examples are used as opposed to requiring both positive and negative examples.

The rest of the paper is organized as follows: Section 2 presents the lexical and syntactic patterns as used for question detection. Section 3 describes the learning algorithm for the classification model. Section 4 shows our experimental results. Section 5 reviews some related work and Section 6 concludes this paper.

2 Pattern Mining for Question Detection

As has been discussed, human generated content on the Web are usually not well formatted, and naive methods such as the use of question mark and 5W1H words are not adequate to correctly detect or capture all online questions. Methods based on hand-crafted rules also fail to cope with various question forms as randomly appeared on the Web. To overcome the shortcomings of these traditional methods, we propose to extract a set of salient patterns from online questions and use them as features to detect question sentences.

In this study, we mainly focus on two kinds of patterns – *sequential pattern* at the lexical level and *syntactic shallow pattern* at the syntactic level. Sequential patterns have been well discussed in many literature, including the identification of comparative sentences (Jindal and Liu, 2006), the detection of erroneous sentences (Sun et al., 2007) and question sentences (Cong et al., 2008) etc. However, works on syntactic patterns have only been partially explored (Zaki and Aggarwal, 2003; Sun et al., 2007; Wang et al., 2009). Grounded on these previous works, we next explain our mining approach of the sequential and syntactic shallow patterns.

2.1 Sequential Pattern Mining

Sequential Pattern is also referred to as *Labeled Sequential Pattern (LSP)* in the literature. It is in the form of $S{\Rightarrow}C$, where S is a sequence $\{t_1, \ldots, t_n\}$, and C is the class label that the sequence S is classified to. In the problem of question detection, a sequence is defined to be a series of tokens from questions, and the class labels are $\{Q, NQ\}$, which stand for question and non-question respectively.

The purpose of sequential pattern mining is to extract a set of frequent subsequence of words that are indicative of questions. For example, the word subsequence "*anyone know what* … *to*" could be a good indication to characterize the question sentence "*anyone know what I can do to make me less tired.*". Note that the mined sequential tokens need not to be contiguous as appeared in the original text.

There is a handful of algorithms available for frequent subsequence extraction. Pei et al. (2001) observed that all occurrences of a frequent pattern can be classified into groups (approximated pattern) and proposed a Prefixspan algorithm. The Prefixspan algorithm quickly finds out all relative frequent subsequences by a pattern growth method, and determines the approximated patterns from those subsequences. We adopt this algorithm in our work due to its high reported efficiency. We impose the following additional constraints for better control over the significance of the mined patterns:

1. Maximum Pattern Length: It limits the maximum number of tokens in a mined sequence.
2. Maximum Token Distance: The two adjacent tokens t_n and t_{n+1} in the pattern need to be within a threshold window in the original text.
3. Minimum Support: The minimum percentage of sentences in Q containing the pattern p.
4. Minimum Confidence: The probability of a pattern $p \Rightarrow Q$ being true in the whole database.

To overcome the word sparseness problem, we generalize each sentence by applying the Part-of-Speech (POS) tags to all tokens except some indicative keywords such as 5W1H words, modal words, stopwords etc. For instance, the question sentence *"How can I quickly tell if my wisdom teeth are coming"* is converted to *"How can I RB VBP if my NN NNS VBP VBG"*, on top of which the pattern mining is conducted. To further capture online language patterns, we mine a set of frequent tokens that are unique to cQA such as *"any1"*, *"im"* and *"whats"*, and keep them from being generalized. The reason to hold back this set of tokens is twofold. First, conventional POS taggers are trained from standard English corpus, and they could mis-tag these non-standard words. Second, the special online tokens are analogue to standard stopwords, and having them properly excluded could help reflect the online users' textual questioning patterns.

It is expected that the converted patterns preserve the most representative features of online questions. Each discovered pattern makes up a binary feature for the classification model that we will introduce in Section 3.

2.2 Syntactic Shallow Pattern Mining

The sequential patterns represent features at the lexical level, but we found that lexical patterns might not always be adequate to categorize questions. For example, the pattern {*when, do*} could presume the non-question "*Levator scapulae is used when you do the traps workout*" to be a question, whereas the question "*know someone with an eating disorder?*" could be overlooked due to the lack of indicative lexical patterns.

These limitations, however, could be alleviated by syntactic features. The syntactic pattern *(SBAR(WHADVP(WRB))(S(NP)(VP)))* extracted from the former example has the order of *NP* and *VP* being switched, which could indicate the sentence to be a non-question, whereas the pattern *(VP(VB)(NP(NP)(PP)))* may be evidence that the latter example is indeed a question, because this pattern is commonly witnessed in the archived questions. Figure 2 shows an example that two questions bear very different wordings but share the same questioning pattern *(S(NP(NN))(VP(VPB)(NP)))* at the syntactic level. In view of the above, we argue that patterns at the syntactic level could complement lexical patterns in identifying question sentences.

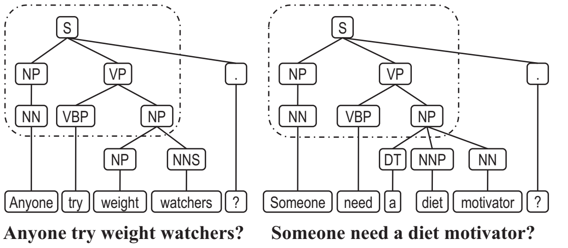


Figure 2: An example of common syntactic patterns observed in two different question sentences

To our knowledge, the mining of salient patterns at the syntactic level was limited to a few tasks. Zaki and Aggarwal (2003) employed tree patterns to classify XML data, Sun et al. (2007) extracted all frequent sub-tree structures for erroneous sentences detection, and Wang et al. (2009) decomposed the parsing tree into fragments and used them to match similar questions. Our work differs from these previous works in that: (1) we also utilize syntactic patterns for the question detection; and (2) we do not blindly extract all possible sub-tree structures, but focus only on certain portions of the parsing tree for better pattern representation and extraction efficiency.

Given a syntactic tree T, we define *syntactic pattern* as a part of sub-structures of T such that the production rule for each non-leaf node in the patterns is intact. For example, the pattern *(S(NP(NN))(VP(VPB)(NP)))* in Figure 2 is considered to be a valid syntactic pattern, whereas *(S(NP(NN))(VP(VPB)))* is not, since the production rule *VP→VPB·NP* is not strictly complied.

We take the following measures to mine salient syntactic patterns: First, we limit the depth of each syntactic pattern to be within a certain range.

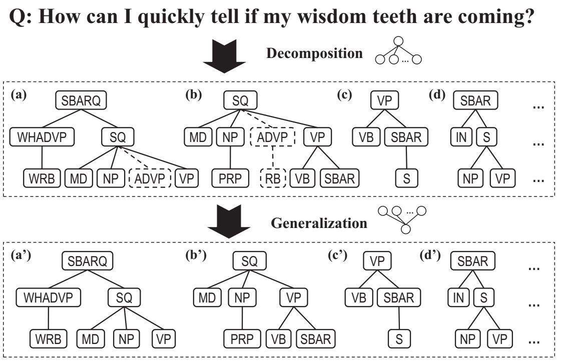


Figure 3: Illustration of syntactic pattern extraction and generalization process

It is believed that the syntax structure will become too specific if it is extended to a deeper level or too general if the depth is too shallow, neither of which produces good representative patterns. We therefore set the depth D of each syntactic pattern to be within a reasonable range ($2 \leq D \leq 4$). Second, we prune away all leaf nodes as well as the production rules at the POS tag level. We believe that nodes at the bottom levels do not carry much useful structural information favored by question detector. For example, the simple grammar rule *NP→DT·NN* does not give any insight to useful question structures. Third, we relax the definition of syntactic pattern by allowing the removal of some nodes denoting modifiers, preposition phrases, conjunctions etc. The reason is that these nodes are not essential in representing the syntactic patterns and are better excluded for generalization purpose. Figure 3 gives an illustration of the process for pattern extraction and generalization. In this example, several syntactic patterns are generated from the question sentence "*How can I quickly tell if my wisdom teeth are coming?*", and the tree patterns (a) and (b) are generalized into (a') and (b'), in which the redundant branch *(ADVP(RB))* that represents the adverb "*quickly*" is detached.

Contents on the Web are prone to noise, and most off-the-shelf parsers are not well-trained to parse online questions. For example, the parsing tree of the question "*whats the matter with it?*" will be very different from that of the question "*what is the matter with it?*". It would certainly be nice to know that "*whats*" is a widely used short form of the phrase "*what is*" on the Web, but we are lack of this kind of thesaurus. Nevertheless, we argue that the parsing errors would not hurt the question detector performance much as long as the mining database is large enough. The reason is that if certain irregular forms frequently occur on the Web, there will be statistical evidences that the syntactic patterns derived from it, though not desired, will commonly occur as well. In other words, we take the wrong patterns and utilize them to detect questions in the irregular forms. Our approach differs from other systems in that we do not intentionally try to rectify the grammatical errors, but leave the errors as they are and use the statistical based approach to capture those informal patterns.

The pattern extraction process is outlined in Algorithm 1. The overall mining strategy is analogous to the mining of sequential patterns, where *support* and *confidence* measures are taken into account to control the significance of the mined patterns. All mined syntactic patterns together with the lexical patterns will be used as features for learning the classification model.

Algorithm 1 *ExtractPattern(S, D)*

```
Input: A set of syntactic trees for sentences ($S$); the depth range ($D$)
Output: A set of sub-tree patterns extracted from $S$
$Patterns$ = {}
for all Syntactic tree $T \in S$ do
    Nodes ← Top-down level order traversal of $T$
    for all node $n \in Nodes$ do
        Extract subtree $p$ rooted under node $n$, with depth within the range $D$
        $p$ ← generalize($p$)
        $Patterns$.add($p$)
    end for
end for
return $Patterns$
```

3 Learning the Classification Model

Although Conditional Random Fields (CRF) is good sequential learning algorithm and has been used in other related work (Cong et al., 2008), here we select Support Vector Machines (SVM) as an alternative learner. The reason is that our task not only deals with sequential patterns but also involves syntactic patterns that possess no sequential criteria. Additionally, SVM has been widely shown to provide superior results compared to other classifiers.

The input to a SVM binary classifier normally consists of both positive and negative examples. While it is easy to discover certain patterns from questions, it is unnatural to identify characteristics for non-questions, as they usually do not share such common lexical and syntactic patterns. The lack of good negative examples leads traditional SVM to perform poorly. To adapt the imbalanced input data, we proposed to employ a one-class SVM method (Manevitz and Yousef, 2002) for learning. The basic idea of one-class SVM is to transform features from only positive examples via a kernel to a hyper-plane and treats the origin as the only member of the second class. It uses relaxation parameters to separate the positive examples from the origin, and finally applies the standard two-class SVM techniques to learn a decision boundary. As a result, anything outside the boundary are considered to be outliers (*i.e.* non-questions in this problem).

More formally, given n training samples $x_1, \ldots, x_n$ of one class, the hyperplane separating them from the origin is constructed by solving

$$\min \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho \qquad (1)$$

subject to: $w \cdot \Phi(x_i) \geq \rho - \xi_i$, where Φ is a kernel function, ξ_i is the slack variable, and ν is the parameter controlling the upper bound percentage of outliers. If w and ρ solve this problem, the decision function $f(x) = sign(w \cdot \Phi(x) - \rho)$ will be positive for most examples x_i in the training set.

Supervised learning methods usually require training data to be manually annotated. To save labeling efforts, we take a shortcut by treating all sentences ending with question marks as an initial positive examples. This assumption is acceptable, as Cong et al. (2008) reported that the rule-based method using only question mark achieves a very high precision of over 97% in detecting questions. It in turn indicates that questions ending with "?" are highly reliable to be real questions.

However, the initial training data still contain many sentences ending with "?" but are not true questions. These possible outliers will shift the decision boundary away from the optimal one, and we need to remove them from the training dataset for better classification. Many preprocessing strategies are available for training data refinement, including bootstrapping, condensing, and editing etc. In this work, we employ a SVM-based data editing and classification method proposed by Song et al. (2008), which iteratively sets a small value to the parameter ν of the one-class SVM so as to continuously refine the decision boundary. The algorithm could be better visualized with Figure 4. In each iteration, a new decision boundary will be determined based on the existing set of data points, and a portion of possible outliers will be removed from the training set. It is expected that the learned hyperplane will eventually be very close to the optimal one.

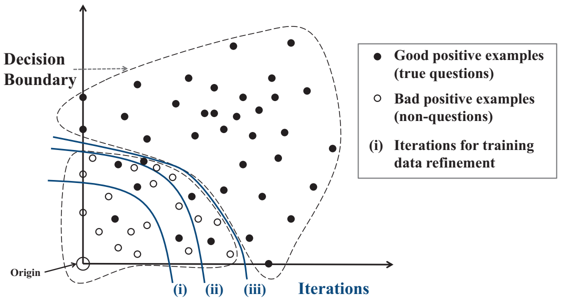

Decision Boundary
● Good positive examples (true questions)
○ Bad positive examples (non-questions)
(i) Iterations for training data refinement
Origin
(i)
(ii)
(iii)
Iterations

Figure 4: Illustration of one-class SVM classification with training data refinement (conceptual only). Three iterations (i) (ii) (iii) are presented.

We use the freely available software LIBSVM[2] to conduct the one-class SVM training and testing. A linear kernel is used, as it is shown to be superior in our experiments. In each refinement iteration, the parameter ν is conservatively set to 0.02. The number of iteration is dynamically determined according to the algorithm depicted in (Song et al., 2008). Other parameters are all set to default. The refined decision boundary from the training dataset will be applied to classify questions from non-questions. The question detector model learned will serve as a component for the cQA question retrieval system in our experiments.

4 Experiments

In this section, we present empirical evaluation results to assess the effectiveness of our question detection model. In particular, we first examine the effects of the number of patterns on question detection performance. We further conduct experiments to show that our question de-

[2] Available at: http://www.csie.ntu.edu.tw/˜cjlin/libsvm

# of Lexical Patterns		Confidence					# of Syntactic Patterns		Confidence				
		60%	65%	70%	75%	80%			60%	65%	70%	75%	80%
Support	0.40%	1685	1639	1609	1585	1545	Support	0.03%	916	758	638	530	453
	0.45%	1375	1338	1314	1294	1277		0.04%	707	580	488	402	341
	0.50%	1184	1151	1130	1113	1110		0.05%	546	450	375	308	261
	0.55%	1037	1007	989	975	964		0.06%	468	379	314	260	218

Table 1: Number of lexical and syntactic patterns mined over different *support* and *confidence* values

Lexical Patterns		Confidence									Syntactic Patterns		Confidence								
		65%			70%			75%					60%			65%			70%		
		P	R	F_1	P	R	F_1	P	R	F_1			P	R	F_1	P	R	F_1	P	R	F_1
Support	0.40%	85.7	90.7	88.1	86.9	88.6	87.7	87.8	86.6	87.2	Support	0.03%	80.4	83.3	81.9	85.1	77.5	81.1	90.7	70.2	79.1
	0.45%	86.6	90.2	88.4	88.9	88.5	88.7	89.6	86.7	88.2		0.04%	79.0	86.1	82.4	90.1	78.2	83.7	90.8	70.8	79.6
	0.50%	88.5	91.6	88.4	86.4	89.0	87.7	86.2	87.9	87.0		0.05%	80.3	82.5	81.4	88.8	78.4	83.3	89.9	69.0	78.1
	0.55%	86.5	89.9	88.1	88.1	87.5	87.8	88.0	89.2	88.6		0.06%	83.0	83.2	83.1	88.5	77.2	82.4	86.7	75.8	80.9

Table 2: Question detection performance over different sets of lexical patterns and syntactic patterns

tection model combining both lexical and syntactic features outperforms traditional rule-based or lexical-based methods. We finally demonstrate that our question detection model gives additional performance boosting to question matching.

4.1 Performance Variation over Different Pattern Sets

The performance of the question detection model can be sensitive to the number of features used for learning. To find the optimal number of features used for model training, we examine the performance variation over different amount of lexical and syntactic patterns undertaken for training.

Dataset: We collected a total of around 800k question threads from Yahoo! Answers Healthcare domain. From the collected data, we generated the following three datasets:

- Pattern Mining Set: Comprising around 350k sentences from 60k question threads, where those ending with "?" are treated as questions and others as non-questions.
- Training Set: Positive examples comprising around 130k sentences ending with "?" from another 60k question threads for the one-class SVM learning algorithm.
- Testing Set: Two annotators are asked to tag randomly picked sentences from the remaining set. A total of 2,004 question sentences and 2,039 non-question sentences are annotated.

Methods & Results: We use different combinations of *support* and *confidence* values to generate different set of patterns. The *support* value ranges from 0.40% to 0.55% for lexical patterns with a step size of 0.05%, and ranges from 0.03% to 0.06% for syntactic patterns with a step size of 0.01%. The *confidence* value for both patterns ranges from 60% to 80% with a step size of 5%. These value ranges are empirically determined. Table 1 presents the number of lexical and syntactic patterns mined against different *support* and *confidence* value combinations.

For each set of lexical or syntactic patterns mined, we use them as features for model training. We convert the training sentences into a set of feature vectors and employ the one-class SVM algorithm to train a classifier. The classifier will then be applied to predict the question sentences in the testing set. To evaluate each question detection model, we employ Precision (P), Recall (R), and F_1 as performance metrics, and Table 2 presents the results[3].

We observe from Table 2 that given a fixed support level, the precision generally increases with the confidence level for both lexical and syntactic patterns, but the recall drops. The lexical feature set comprising 1,314 sequential patterns as generated with $\{$*sup=0.45%*, *conf=70%*$\}$ gives the best F_1 score of 88.7%, and the syntactic feature set comprising 580 syntactic patterns generated from $\{$*sup=0.04%*, *conf=65%*$\}$ gives the best F_1 score of 83.7%. It is noted that the sequential patterns give relatively high recall while the syntactic patterns give relatively high precision. Our reading is that the sequential patterns are capable of capturing most questions, but it may also give wrong predictions to non-questions such as "*Lev-*

[3] The results for certain *confidence* levels are not very promising and are not shown in the table due to lack of space.

ator scapulae is used when you do the traps workout" that bears the sequential pattern $\{$*when, do*$\}$. On the other hand, the syntactic patterns could give reliable predictions, but its coverage could suffer due to the limited number of syntactic patterns. We conjecture that a combination of both features could further improve the performance.

4.2 Performance Comparison with Traditional Question Detection Methods

We next conduct experiments to compare the performance of our question detection model to traditional rule-based or lexical-based methods.

Methods & Results: We set up five different systems for meaningful comparisons:

1. 5W1H (baseline1): a rule-based method using 5W1H to determine a question sentence.
2. Question Mark (baseline2): a method using the question mark "?" to judge a question.
3. SeqPattern: Using only the set of 1,314 sequential patterns as features.
4. SynPattern: Using only the set of 580 syntactic patterns as features.
5. SeqPattern+SynPattern: Merging both lexical and syntactic patterns and use them as a set of features for question detection.

We again employ Precision (P), Recall (R), and F_1 as performance metrics to evaluate each question detection system, and tabulate the comparison results in Table 3. From the Table, we observe that 5W1H performs poorly in both precision and recall, and question mark based method gives relatively low recall although the precision is the highest amongst all the methods evaluated. This is in line with the results as observed in (Cong et al., 2008). SeqPattern outperforms the two baseline systems in both R and F_1 scores, and its combination with SynPattern augments the performance in both precision and recall by a lot. It also achieves statistically significant improved results (t-test, p-value<0.05) as compared to other four systems. These results are consistent with our intuition that syntactic patterns can leverage sequential patterns in improving the question detection performance.

It is noted that SeqPattern+SynPattern exhibits the highest recall (R) amongst all the systems. The significance test further suggests that many question sentences miss-detected by 5W1H or Question Mark method could be properly captured by our model. This improvement is meaningful, as the question coverage is also an important factor in the cQA question retrieval task, where high recall implies that more similar questions could be matched and returned, hence improving the question retrieval performance.

System Combination	$P(\%)$	$R(\%)$	$F_1(\%)$
(1) 5W1H	75.37	49.50	59.76
(2) Question Mark	**94.12**	77.50	85.00
(3) SeqPattern	88.92	88.47	88.69
(4) SynPattern	90.06	78.19	83.71
(5) SeqPattern+SynPattern	92.11	**89.67**	**90.87**

Table 3: Performance comparisons for question detection on different system combinations

4.3 Performance Evaluation on Question Retrieval with Question Detection Model

To further demonstrate that our question detection model can improve question retrieval, we incorporate it into different question retrieval systems.

Methods: We select a simple bag-of-word (BoW) system retrieving questions at the lexical level, and a syntactic tree matching (STM) model matching questions at the syntactic level (Wang et al., 2009) as two baselines. For each baseline, we further set up two different combinations:

- Baseline+QM: Using question mark to detect question sentences, and perform question retrieval on top of the detected questions.
- Baseline+QD: Using our proposed model to detect question sentences, and perform question retrieval on top of the detected questions.

This gives rise to additional 4 different system combinations for comparison.

Dataset: We divide the dataset from Yahoo! Answers into a question repository set (750k) and a test set (50k). For the baseline systems, all the repository sentences containing both questions and non-questions are indexed, whereas for systems equipped with QM or QD, only the detected question sentences are indexed for retrieval. We randomly select 250 single-sentence questions from the test set as queries, and for each query, the retrieval system will return a list of top 10 question matches. We combine the retrieved results from different systems and ask two annotators to label each result to be either "relevant" or "irrel-

System Combination	BoW	BoW +QM	BoW +QD	STM	STM +QM	STM +QD
MAP (%)	58.07	59.89	60.68	66.53	68.41	69.85
% improvement of MAP over:						
Baseline	N.A.	+3.13	+4.49	N.A.	+2.83	+4.99
Baseline+QM	N.A.	N.A.	+1.32	N.A.	N.A.	+2.10
P@1 (%)	59.81	61.21	63.55	63.08	64.02	65.42

Table 4: Question retrieval performance on different system combinations measured by MAP and P@1 (Baseline is either BoW or STM)

evant” without telling them which system the result is generated from. By eliminating some query questions that have no relevant matches, the final testing set contains 214 query questions.

Metrics & Results: We evaluate the question retrieval performance using two metrics: Mean Average Precision (MAP) and Top One Precision (P@1). The results are presented in Table 4.

We can see from Table 4 that STM outperforms BoW. Applying QM or QD over BoW and STM boosts the system performance in terms of both MAP and P@1. They also achieve statistical significance as judged by paired t-test (p-value$<$0.05). More specifically, the MAP on QM coupled systems improves by 3.13% and 2.83% respectively over BoW and STM. This is evidence that having question sentences clearly identified could help to retrieve relevant questions more precisely, as without question detection, the user query is likely to be matched to irrelevant description sentences. Our question detection model (QD) further improves the MAP by 1.32% and 2.1% respectively over BoW+QM and STM+QM, and it also yields better top one precision by correctly retrieving questions at the first position on 136 and 140 questions respectively, out of a total of 214 questions. These improvements are in line with our expectation that our model incorporating salient features at both the lexical and syntactic levels is comprehensive enough to capture various forms of questions online, and hence improve the performance of question matching.

5 Related Work

Research on detecting question sentences can generally be classified into two categories. The first category simply employs rule-based methods such as question mark, 5W1H words, or hand-crafted regular expressions to detect questions. As discussed, these conventional methods are not adequate to cope with online questions.

The second category uses machine learning approaches to detect question sentences. Shrestha and McKeown (2004) proposed a supervised rule induction method to detect interrogative questions in email conversations based on part-of-speech features. Yeh and Yuan (2003) used a statistical approach to extract a set of question-related words and derived some syntax and semantic rules to detect mandarin question sentences. Cong et al. (2008) extracted labeled sequential patterns and used them as features to learn a classifier for question detection in online forums.

Question pattern mining is also closely related to the learning of answer patterns. Work on answer patterns includes the web based pattern mining (Zhang and Lee, 2002; Du et al., 2005) and a combination of syntactic and semantic elements (Soubbotin and Soubbotin, 2002) etc.

In contrast to previous work, we do not only focus on standard language corpus, but extensively explore characteristics of online questions. Our approach exploits salient question patterns at both the lexical and syntactic levels for question detection. In particular, we employ the one-class SVM algorithm such that the learning process is weakly supervised and no human annotation is involved.

6 Conclusion

This paper proposed a new approach to detecting question sentences in cQA. We mined both lexical and syntactic question patterns, and used them as features to build classification models. The mining and leaning process is fully automated and requires no human intervention. Empirical evaluation on the cQA archive demonstrated the effectiveness of our model as well as its usefulness in improving question retrieval performance.

We are still investigating other features that are helpful to detect questions. One promising direction for future work is to also employ lexical and syntactic patterns to other related areas such as question type classification etc. It is also interesting to employ a hybrid of CRF and SVM learning methods to boost the accuracy and scalability of the classifier.

References

Agichtein, Eugene, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding high-quality content in social media. In *WSDM*, pages 183–194.

Cong, Gao, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun. 2008. Finding question-answer pairs from online forums. In *SIGIR*, pages 467–474.

Du, Yongping, Helen Meng, Xuanjing Huang, and Lide Wu. 2005. The use of metadata, web-derived answer patterns and passage context to improve reading comprehension performance. In *HLT*, pages 604–611.

Duan, Huizhong, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching questions by identifying question topic and question focus. In *HLT-ACL*, pages 156–164.

Jeon, Jiwoon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *CIKM*, pages 84–90.

Jindal, Nitin and Bing Liu. 2006. Identifying comparative sentences in text documents. In *SIGIR*, pages 244–251.

Manevitz, Larry M. and Malik Yousef. 2002. One-class svms for document classification. *J. Mach. Learn. Res.*, 2:139–154.

Pei, Jian, Jiawei Han, Behzad Mortazavi-asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei chun Hsu. 2001. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE*, pages 215–224.

Riezler, Stefan, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical machine translation for query expansion in answer retrieval. In *ACL*, pages 464–471.

Shrestha, Lokesh and Kathleen McKeown. 2004. Detection of question-answer pairs in email conversations. In *COLING*, page 889.

Song, Xiaomu, Guoliang Fan, and M. Rao. 2008. Svm-based data editing for enhanced one-class classification of remotely sensed imagery. *Geoscience and Remote Sensing Letters, IEEE*, 5(2):189–193.

Soubbotin, Martin M. and Sergei M. Soubbotin. 2002. Use of patterns for detection of likely answer strings: A systematic approach. In *TREC*.

Sun, Guihua, Gao Cong, Xiaohua Liu, Chin-Yew Lin, and Ming Zhou. 2007. Mining sequential patterns and tree patterns to detect erroneous sentences. In *AAAI*, pages 925–930.

Wang, Kai, Zhaoyan Ming, and Tat-Seng Chua. 2009. A syntactic tree matching approach to finding similar questions in community-based qa services. In *SIGIR*, pages 187–194.

Xue, Xiaobing, Jiwoon Jeon, and W. Bruce Croft. 2008. Retrieval models for question and answer archives. In *SIGIR*, pages 475–482.

Yeh, Ping-Jer and Shyan-Ming Yuan. 2003. Mandarin question sentence detection: A preliminary study. In *EPIA*, pages 466–478.

Zaki, Mohammed J. and Charu C. Aggarwal. 2003. Xrules: an effective structural classifier for xml data. In *KDD*, pages 316–325.

Zhang, Dell and Wee Sun Lee. 2002. Web based pattern mining and matching approach to question answering. In *TREC*.

Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering

Mengqiu Wang
Computer Science Department
Stanford University
mengqiu@cs.stanford.edu

Christopher D. Manning
Computer Science Department
Stanford University
manning@cs.stanford.edu

Abstract

A range of Natural Language Processing tasks involve making judgments about the semantic relatedness of a pair of sentences, such as Recognizing Textual Entailment (RTE) and answer selection for Question Answering (QA). A key challenge that these tasks face in common is the lack of explicit alignment annotation between a sentence pair. We capture the alignment by using a novel probabilistic model that models tree-edit operations on dependency parse trees. Unlike previous tree-edit models which require a separate alignment-finding phase and resort to ad-hoc distance metrics, our method treats alignments as structured latent variables, and offers a principled framework for incorporating complex linguistic features. We demonstrate the robustness of our model by conducting experiments for RTE and QA, and show that our model performs competitively on both tasks with the same set of general features.

1 Introduction

Many complex Natural Language Processing (NLP) applications can be broken down to a subtask of evaluating the semantic relationship of pairs of sentences (e.g., in Question Answering, answer selection involve comparing each answer candidate against the question). This means that research aiming at analyzing pairs of semantically related natural language sentences is promising because of its reusability: it is not tied to a particular internal representation of meanings, but it nevertheless serves as a first step towards full meaning understanding, which is applicable to a number of applications. At the same time, this paradigm clearly defines the input and output space, facilitating system comparison and standard evaluation. Tasks of this paradigm have drawn much of the focus in recent NLP research, including Recognizing Textual Entailment (RTE), answer selection for Question Answering (QA), Paraphrase Identification (PI), Machine Translation Evaluation (MTE), and many more.

In each of these tasks, inputs to the systems are pairs of sentences that may or may not convey the desired semantic property (e.g., in RTE, whether the hypothesis sentence can be entailed from the premise sentence; in QA, whether the answer candidate sentence correctly answers the question), and the output of the system is a binary classification decision (or a regression score,as in MTE).

Earlier studies in these domains have concluded that simple word overlap measures (e.g., bag of words, n-grams) have a surprising degree of utility (Papineni et al., 2002; Jijkoun and de Rijke, 2005b), but are nevertheless not sufficient for these tasks (Jijkoun and de Rijke, 2005a). A common problem identified in these earlier systems is the lack of understanding of the semantic relation between words and phrases. Later systems that include more linguistic features extracted from resources such as WordNet have enjoyed more success (MacCartney et al., 2006). Studies have also shown that certain prominent syntactic features are often found beneficial (Snow et al., 2006). More recent studies gained further leverage from systematic exploration of the syntactic feature space through analysis of parse trees (Wang et al.,

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1164–1172,
Beijing, August 2010

2007; Das and Smith, 2009).

There are two key challenges imposed by these tasks. The first challenge has to do with the hidden alignment structures embedded in the sentence pairs. It is straightforward to see that in order to extract word-matching and/or syntax-matching features, inevitably one has to consider the alignment between words and/or syntactic parts. These alignments are not given as inputs, and it is a non-trivial task to decide what the *correct* alignment is. Alignment-based approach have been proven effective by many RTE, QA and MTE systems (Haghighi et al., 2005; Wang et al., 2007; MacCartney et al., 2008; Das and Smith, 2009, *inter alia*). Although alignment is a commonly used approach, it is not the only one. Other studies have successfully applied theorem proving and logical induction techniques, translating both sentences to knowledge representations and then doing inference on these representations (Moldovan et al., 2003; Raina et al., 2005; de Salvo Braz et al., 2005; MacCartney and Manning, 2007, *inter alia*).

A second challenge arises when a system needs to combine various sources of evidence (i.e., surface text features, semantic features, and syntactic features) to make a global classification decision. Quite often these features are heavily overlapping and sometimes contradicting, and thus a robust learning scheme that knows when to activate what feature is desired. Traditional approaches employ a two-stage or multi-stage model where tasks are broken down into alignment finding, feature extraction, and feature learning subtasks (Haghighi et al., 2005; MacCartney et al., 2008). The alignment finding task is typically done by committing to a *one best* alignment, and subsequent features are extracted only according to this alignment. A large body of literature in joint learning has demonstrated that such an approach can suffer from cascaded errors at testing, and does not benefit from the potential for joint learning (Finkel et al., 2006).

In this paper, we present a novel undirected graphical model to address these challenges. A promising approach to these challenges is modeling the alignment as an edit operation sequence over parse tree representation, an approach pioneered by (Punyakanok et al., 2004; Kouylekov and Magnini, 2006; Harmeling, 2007; Mehdad, 2009). We improve upon this earlier work by showing how alignment structures can be inherently learned as structured latent variables in our model. Tree edits are represented internally as state transitions in a Finite-State Machine (FSM), and our model is parameterized as a Conditional Random Field (CRF) (Lafferty et al., 2001), which allows us to incorporate a diverse set of arbitrarily overlapping features.

In comparison to previous work that exploits various ad-hoc or heuristic ways of incorporating tree-edit operations, our model provides an elegant and much more principled way of describing tree-edit operations in a probabilistic setting.

2 Tree-edit CRF for Classification

A training instance consists of a pair of sentences and an associated binary judgment. In RTE, for example, the input sentence pairs is made up of a text sentence (e.g., *Gabriel Garcia Marquez is a novelist and winner of the Nobel prize for literature.*) and a hypothesis sentence (e.g., *Gabriel Garcia Marquez won the Nobel for Literature.*). The pair is judged to be *true* if the hypothesis can be entailed from the text (e.g., the answer is *true* for the example sentence pair).

Formally, we denote the text sentence as *txt* and the hypothesis sentence as *hyp*, and denote their labeled dependency parse trees as $\tau_\mathbf{t}$ and $\tau_\mathbf{h}$, respectively. We use the binary variable $z \in \{0, 1\}$ to denote the judgment.

The generative story behind our model is a parse tree transformation process. $\tau_\mathbf{t}$ is transformed into $\tau_\mathbf{h}$ through a sequence of *tree edits*. Examples of tree edits are *delete child*, *insert parent*, and *substitute current*. An edit sequence $\mathbf{e} = e_1 \ldots e_m$ is *valid* if $\tau_\mathbf{t}$ can be successfully turned into $\tau_\mathbf{h}$ according to $\mathbf{e}$. An example of a trivial valid edit sequence is one that first *deletes* all nodes in $\tau_\mathbf{t}$ then *inserts* all nodes in $\tau_\mathbf{h}$.

Delete, *insert* and *substitute* form the three basic edit operations. Each step in an edit sequence is also linked with current edit positions in both trees, denoted as $\mathbf{e}.\mathbf{p} = e_1.p \ldots e_m.p$. We index the tree nodes using a level-order tree traversal scheme (i.e., root is visited first and assigned index 0, then each one of the first level children of the root is visited in turn, and assigned an index number incremented by 1). It is worth noting that every valid edit sequence has a corresponding alignment mapping. Nodes that are inserted or deleted are aligned to *null*, and nodes that are substituted are aligned. One can find many edit sequence for the same alignment, by altering the order of edit operations.

We extend these basic edit operations into more elaborate edit operations based on the linguistic and syntactic properties of the current tree nodes that they fire on. For example, the following are all possible edit operations: *delete* a noun that is SUB of the root, *delete* a named-entity of type PERSON, *substitute* roots of the tree. In our experiments, we designed a set of 45 edit operations (12 *delete*, 12 *insert* and 21 *substitute*). More details of the edit operations are described in §4. Depending on the specific application domain, more sophisticated and verbose tree edit operations can be designed and easily incorporated into our model. In particular, tree edit operations involving deleting, inserting or substituting entire treelets seem interesting and promising, requiring merely a simple extension to the forward-backward dynamic programming.

Next, we design a Finite-State Machine (FSM) in which each edit operation is mapped to a unique state, and an edit sequence is mapped into a transition sequence among states (denoted as $\mathbf{e}.\mathbf{a} = e_1.a \ldots e_m.a$). In brief, an edit sequence is associated with a sequence of edit positions in the trees ($\mathbf{e}.\mathbf{p} = e_1.p \ldots e_m.p$), as well as a transition sequence among states ($\mathbf{e}.\mathbf{a} = e_1.a \ldots e_m.a$).

The probability of an edit sequence $\mathbf{e}$ given the parse trees is defined as:

$$P(\mathbf{e} \mid \tau_{\mathbf{t}}, \tau_{\mathbf{h}}) = \frac{1}{Z} \prod_{i=1}^{|\mathbf{e}|} \exp \theta \cdot \mathbf{f}(e_{i-1}, e_i, \tau_{\mathbf{t}}, \tau_{\mathbf{h}}) \quad (1)$$

where $\mathbf{f}$ are feature functions, θ are associated feature weights, and Z is the partition function to be defined next.

Recall that our training data is composed of not only positive examples but also negative examples. In order to take advantage of this label information, we adopt an interesting discriminative learning framework first introduced by McCallum et al. (2005). We call the FSM state set described above the *positive* state set ($\mathcal{S}_1$), and duplicate the exact same set of states, and call the new set *negative* state set ($\mathcal{S}_0$). We then add a starting state(S_s), and add non-deterministic transitions from S_s to every state in $\mathcal{S}_1$. We then add the same transitions for $\mathcal{S}_0$. We now arrive at a new FSM structure where upon arriving at the starting state, one makes a non-deterministic decision to enter either the positive set or the negative set and stay in that set until reaching the end of the edit sequence, since no transitions are allowed across the positive and negative set. Each edit operation sequence can now be associated with a sequence of positive states as well as a sequence of negative states. The intuitive idea is that during training, we want to maximize the weights of the positive examples in the positive state set and minimize their weights in the negative state set, and vice versa. In other words, we want the positive state set to attract positive examples but push away negative examples. Figure 1 illustrates two example valid edit sequences in the FSM, one in the positive state set and one in the negative state set.

Formally, the partition function Z in (1) is defined as the sum of weights of all valid edit sequences in both the positive set and negative set. Features extracted from positive states are disjoint from features extracted from negative states.

$$Z = \sum_{\mathbf{e}:\, \mathbf{e}.\mathbf{a} \subseteq S_s + \{\mathcal{S}_0 \bigcup \mathcal{S}_1\}^*} \prod_{i=1}^{|\mathbf{e}|} \exp \theta \cdot \mathbf{f}(e_{i-1}, e_i, \tau_{\mathbf{t}}, \tau_{\mathbf{h}})$$

Recall $z \in \{0,1\}$ is the binary judgment indicator variable. The conditional probability of z is obtained by marginalizing over all edit sequences that have state transitions in the state set corresponding to z:

$$P(z \mid \tau_{\mathbf{t}}, \tau_{\mathbf{h}}) = \sum_{\mathbf{e}:\, \mathbf{e}.\mathbf{a} \subseteq S_s + \mathcal{S}_z^*} P(\mathbf{e} \mid \tau_{\mathbf{t}}, \tau_{\mathbf{h}}) \quad (2)$$

The L_2-norm penalized log-likelihood over n training examples ($\mathcal{L}$) is our training objective function:

$$\mathcal{L} = \sum_{j=1}^{n} \log(P(z^{(j)} \mid \tau_{\mathbf{t}}^{(j)}, \tau_{\mathbf{h}}^{(j)})) - \frac{\|\theta\|^2}{2\sigma^2} \quad (3)$$

At test time, the z with higher probability is taken as our prediction outcome.

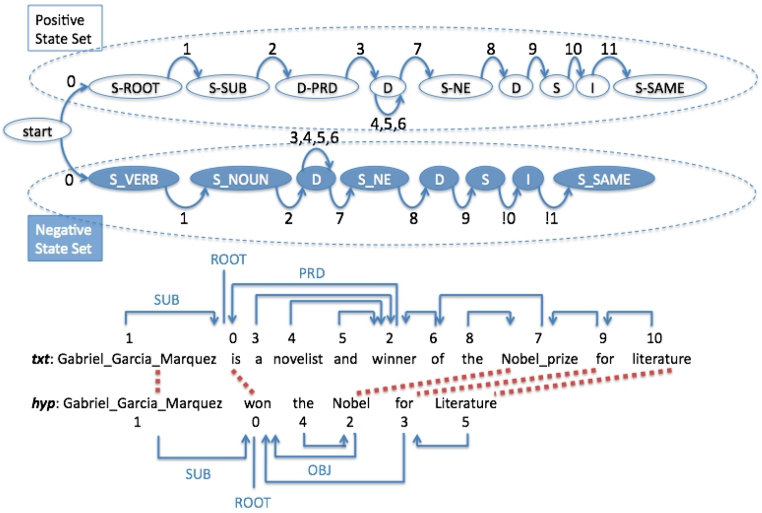


Figure 1: This diagram illustrates the FSM architecture. There is a single start state, and we can transit into either the positive state set (nodes that are not shaded), or the negative state set (shaded nodes). Here we show two examples of valid edit sequences. They result in the same alignment structure as show in the bottom half of the diagram (dotted lines across the two sentences are alignment links). Numbers over the arcs in the state diagram denote the edit sequence index, and numbers under each word in the parse tree diagram denote each node's level-order index number.

3 Parameter Estimation

We used Expectation Maximization method since the objective function given in (3) is non-convex. In the M-step, finding the optimal parameters under the current model expectation involves computing forward-backward style dynamic programming (DP) in a three-dimensional table (two for inputs and one for states) and optimization using L-BFGS method. In practice the resulting DP table can be quite large (for a sentence pair of length 100, and 2 sets of 45 states, we obtain 900,000 entries). We improved efficiency by pruning out partial sequences that do not lead to a complete valid sequence and pre-compute the state-transition table and features.

4 Edit Operations

Table 1 lists the groups of edit operations we designed and their descriptions. Not shown in the table are three default edits (*insert*, *delete* and *substitute*), which fire when none of the more specific edit operations match. Edit operations listed in the the top-left section capture basic matching, deletion and insertion of surface text, part-of-speech tags and named-entity tags. The top-right section capture alignments of semantically related words, based on relational information extracted from various linguistic resources, such as WordNet and NomBank. And the bottom section capture syntactic edits. Note that multiple edit operations can fire at the same edit position if conditions are matched (e.g., we can choose to delete if there are more words to edit in *txt*, or to insert if there are more words to edit in *hyp*).

5 Features

One of the most distinctive advantages of our model compared to previous tree-edit based models is the ability to include a wide range of non-independent, rich linguistic features. The features we employed can be broken down into two categories. The first category is *zero-order* features that model the current edit step. They consist of a conditioning property of the current edit, and the current state in the FSM. The second category is *first-order* features that capture state transitions, by concatenating the current FSM state with the previous FSM state. One simple form of zero-order feature is the current FSM state itself. The FSM states already carry a lot of information about the current edits. Conditioning properties are used to further describe the current edit. They are often more fine-grained and complex (e.g.,

Surface edits		**Semantic edits**	
{I,D,S}-{POS}	insert/delete/substitute words of a POS type, where POS is *noun*, *verb* or *proper noun*	S-SYNONYM	substitute two words that are synonyms
		S-HYPERNYM	substitute two words that are hypernyms
{I,D,S}-NE	insert/delete/substitute named-entity words	S-ANTONYM	substitute two words that are antonyms
{I,D,S}-LIKE	insert/delete/substitute words that expresses likelihood, e.g., *maybe*, *possibly*	S-ACRONYM	substitute two words in which one is an acronym of the other
{I,D,S}-MODAL	insert/delete/substitute modal verbs, e.g., *can*, *could*, *may*	S-NOMBANK	substitute two words that are related according to NomBank
S-{SAME/DIFF}	the words being substituted are the same or different	S-NUM-0,1	substitute two words that are both numerical values, and 1 if they match, 0 if they mismatch
Syntactic edits			
{I,D,S}-ROOT	insert/delete/substitute root of the trees		
{I,D,S}-{REL}	insert/delete/substitute a tree node of grammatical relation type, where REL is either SUB, OBJ, VC or PRD		

Table 1: List of edit operations. I for INSERT, D for DELETE, and S for SUBSTITUTE.

syntactic-matching conditions listed below). To give an example, in Figure 1, the second edit operation in the example sequence is S-NE. A matching condition feature that fires with this state could be *substitute_NE_type_PERSON*, which tells us exactly what type of named-entity is being substituted.

It is notable that in designing edit operations and features, there is a continuum of choice in terms of how much information to be encoded as features versus edit operations. To better illustrate the trade-off, consider the two extreme cases of this continuum. At one extreme, we can design a system where there are only three basic edit operations, and all extra information in our current set of edit operations can be encoded as features. For example, in this case edit operation S-NE would become S with feature *substitute_NE*. The other extreme is to encode every zero-order feature as a separate edit operation. The amount of information encoded in the zero-order features and edit operations is the same in both cases, but the difference lies in first-order features and efficiency. When encoding more information as edit operations (and thus more states in FSM), first-order features become much more expressive; whereas when encoding more information as features, computation becomes cheaper as the number of possible state transition sequences is reduced. In our experiments, we aim to keep a minimal set of edit operations that are meaningful but not overly verbose, and encode additional information as features. Each feature is a binary feature initialized with weight 0.

Due to space limitation, we list the most important zero-order features. Many of these features are inspired by MacCartney et al. (2006) and Snow et al. (2006), but not as sophisticated.

Word matching features. These features detect if a text word and a hypothesis word match the following conditions:

1. have the same lemma
2. one is a phrase and contains the other word
3. are multi-word phrases and parts match
4. have the same/different named-entity type(s) + the named-entity type(s)

Tree structure features. These features try to capture syntactic matching/mismatching information from the labeled dependency parse trees. 1. whether the roots of the two trees are aligned
2. parent-child pair match
3. (2.) and labels also match
4. (2.) and labels mismatch
5. (4.) and detailing the mismatching labels
6. parent+label match, child mismatch
7. child and label match, parents are {hyper/syno/anto}nym
8. looking for specific SUB/OBJ/PRD construct as in Snow et al. (2006).

6 Preprocessing

In all of our experiments, each input pair of text and hypothesis sentence is preprocessed as following: Sentences were first tokenized by the standard Penn TreeBank tokenization script, and then we used MXPOST tagger (Ratnaparkhi, 1996) for part-of-speech (POS) tagging. POS tagged sentences were then parsed by MST-Parser (McDonald et al., 2005) to produce labeled dependency parse trees. The parser was trained on the entire Penn TreeBank. The last step in the pipeline is named-entity tagging using Stanford NER Tagger (Finkel et al., 2005).

7 RTE Experiments

Given an input text sentence and a hypothesis sentence, the task of RTE is to make predictions about whether or not the hypothesis can be entailed from the text sentence. We use standard evaluation datasets RTE1-3 from the Pascal RTE Challenges (Dagan et al., 2006). For each RTE dataset, we train a tree-edit CRF model on the training portion and evaluate on the testing portion. We report accuracy of classification results, and precision and recall for the true entailment class. There is a balanced positive-negative sample distribution in each dataset, so a random baseline gives 50% classification accuracy. We used RTE1 for feature selection and tuning σ in the L_2 regularizer ($\sigma = 5$ was used). RTE2 and RTE3 were reserved for testing.

Our system is compared with four systems on RTE2 and three other systems on the RTE3 dataset.[1] We chose these systems for comparison because they make use of syntactic dependencies and lexical semantic information. Notably other systems that give state-of-the-art performance on RTE use non-comparable techniques such as theorem-proving and logical induction, and often involve significant manual engineering specifically for RTE, thus do not make meaningful comparison to our model.

For RTE2, Kouylekov and Magnini (2006) experimented with various TED cost functions and found a combination scheme to work the best for RTE. Vanderwende et al. (2006) used syntactic heuristic matching rules with a lexical-similarity back-off model. Nielsen et al. (2006) extracted features from dependency path, and combined them with word-alignment features in a mixture of experts classifier. Zanzotto et al. (2006) proposed a syntactic cross-pair similarity measure for RTE.

For RTE3, Harmeling (2007) took a similar classification-based approach with transformation sequence features. Marsi et al. (2007) described a system using dependency-based paraphrasing techniques for RTE. de Marneffe et al. (2006) described a system where best alignments between the sentence pairs were first found, then classification decisions were made based on these alignments.

RTE2	**Acc.%**	**Prec.%**	**Rec.%**
Vanderwende et al., 2006	60.2	59.0	67.0
K&M, 2006	60.5	58.9	70.0
Nielsen et al., 2006	61.1	59.0	73.3
Zanzotto et al., 2006	63.9	60.8	78.0
Tree-edit CRF	63.0	61.7	68.5
RTE3	**Acc.%**	**Prec.%**	**Rec.%**
Marsi et al., 2007	59.1	-	-
Harmeling, 2007	59.5	-	-
de Marneffe et al., 2006	60.5	61.8	60.2
Tree-edit CRF	61.1	61.3	65.3

Table 2: Results on RTE2 and RTE3 dataset. Results for de Marneffe et al. (2006) were reported by MacCartney and Manning (2008).

Table 2 presents RTE results. Our model performs competitively on both datasets. On RTE2, our model gives second best performance among the methods we compare against, and the difference in accuracy from the best system is quite small (7 out of 800 examples). We observe a larger gap in recall, suggesting our method tends to give higher precision, which is also commonly found in other syntax-based systems (Snow et al., 2006). It is worth noting that Zanzotto et al. (2006) achieved second place in the official RTE2 evaluation. On RTE3, our model outperforms the other syntax-based systems compared. In particular, out system gives the same precision level as the second best system (de Marneffe et al., 2006) without sacrificing as much recall, which is the most common drawback found in syntax-based systems.

8 QA Experiments

A second Tree-edit CRF model was trained for the task of answer selection for Question Answering. In this task, the input pair consists of a short factoid question (e.g., *Who beat Floyd Patterson to take the title away?*) and an answer candidate sentence (e.g., *He saw Ingemar Johansson knock down Floyd Patterson seven times there in winning the heavyweight title.*). The pair is judged positive if the answer candidate sentence correctly answers the question and provides sufficient con-

[1] Different systems are used for comparison because none of these systems reported performance on both datasets.

System	MAP	MRR
Punyakanok et al., 2004	0.4189	0.4939
Cui et al., 2005	0.4350	0.5569
Wang et al., 2007	0.6029	0.6852
H&S, 2010	**0.6091**	0.6917
Tree-edit CRF	0.5951	**0.6951**

Table 3: Results on QA task reported in Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR).

textual support (i.e., does not merely contain the answer key, for example, *"Ingemar Johansson was a world heavyweight champion"* would not be a correct answer). We followed the same experimental setup as Wang et al. (2007) and Heilman and Smith (2010). The training portion of the dataset consists of 5919 manually judged Q/A pairs from previous QA tracks at Text REtrieval Conference (TREC 8–12). There are also 1374 Q/A pairs for development and 1866 Q/A pairs for testing, both from the TREC 3 evaluation. The task is framed as a sentence retrieval task, and thus Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) are reported for the ranked list of most probable answer candidates. We compare out model with four other systems. Wang et al. (2007) proposed a Quasi-synchronous Grammar formulation of the problem which also models alignment as structured latent variables, but in a generative probabilistic model. Their method gives the current state-of-the-art performance on this task. Heilman and Smith (2010) presented a classification-based approach with tree-edit features extracted from a tree kernel. Cui et al. (2005) developed a dependency-tree based information discrepancy measure. Punyakanok et al. (2004) used a generalized Tree-edit Distance method to score mappings between dependency parse trees. All systems were evaluated against the same dataset as the one we used. Results of replicated systems for the last two were reported by Wang et al. (2007), with lexical-semantic augmentation from WordNet.

Results in Table 3 show that our model gives the same level of performance as Wang et al. (2007), with no statistically significant difference ($p > 5$ in sign test). Both systems out-perform the other two earlier systems significantly.

9 Discussion

Our experiments on RTE and QA applications demonstrated that Tree-edit CRF models provide results competitive with previous syntax-based methods. Even though the improvements were quite moderate in some cases, the important point is that our model provides a novel principled framework. It works across different problem domains with minimal domain knowledge and feature engineering, whereas previous methods are only engineered for a particular task and are hard to generalize to new problems.

While the current Tree-edit CRF model can model a large set of linguistic phenomenon and tree-transformations, it has some clear limitations. One of the biggest drawbacks is the lack of support for modeling phrasal re-ordering, which is a very common and important linguistic phenomena. It is not straightforward to implement re-ordering in the current model because it breaks the word-order constraint which admits tractable forward-backward style dynamic programming. However, this shortcoming can be addressed partially by extending the model to deal with constrained re-ordering per Zhang (1996).

10 Related Work

Tree Edit Distance (TED) have been studied extensively in theoretical and algorithmic research (Klein, 1989; Zhang and Shasha, 1989; Bille, 2005). In recent years we have seen many work on applying TED based methods for NLP-related tasks (Punyakanok et al., 2004; Kouylekov and Magnini, 2006; Harmeling, 2007; Mehdad, 2009). Mehdad (2009) proposed a method based on particle swarm optimization technique to automatically learn the TED cost function. Another work that also developed an interesting approach to stochastic tree edit distance is Bernard et al. (2008), but unfortunately experiments in the paper were limited to digit recognition and tasks on small artificial datasets.

Many different approaches to modeling sentence alignment have been proposed before (Haghighi et al., 2005; MacCartney et al., 2008). Haghighi et al. (2005) treated alignment finding in RTE as a graph matching problem between sentence parse trees. MacCartney et al. (2008) described a phrase-based alignment model for MT, trained by the Perceptron learning algorithm. A line of work that offers similar treatment of alignment to our model is the Quasi-synchronous Grammar (QG) (Smith and Eisner, 2006; Wang et al., 2007; Das and Smith, 2009). QG models alignments between two parse trees as structured latent variables. The generative story of QG describes one that builds the parse tree of one sentence, loosely conditioned on the parse tree of the other sentence. This formalism prefers but is not confined to tree isomorphism, therefore possesses more model flexibility than synchronous grammars.

The work of McCallum et al. (2005) inspired the discriminative training framework that we used in our experiments. They presented a String Edit Distance model that also learns alignments as hidden structures for simple tasks such as restaurant name matching.

Our work is also closely related to other recent work on learning probabilistic models involving structural latent variables (Clark and Curran, 2004; Petrov et al., 2007; Blunsom et al., 2008; Chang et al., 2010). The Tree-edit CRF model we present here is a new addition to this family of interesting models for discriminative learning with structural latent variables.

11 Conclusion

We described a Tree-edit CRF model for predicting semantic relatedness of pairs of sentences. Our approach generalizes TED in a principled probabilistic model that embeds alignments as structured latent variables. We demonstrate a wide-range of lexical-semantic and syntactic features can be easily incorporated into the model. Discriminatively trained, the Tree-edit CRF led to competitive performance on the task of Recognizing Textual Entailment and answer selection for Question Answering.

References

Bernard, M., L. Boyer, A. Habrard, and M. Sebban. 2008. Learning probabilistic models of tree edit distance. *Pattern Recognition*, 41(8):2611–2629.

Bille, P. 2005. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3):217–239.

Blunsom, P., T. Cohn, and M. Osborne. 2008. A discriminative latent variable model for statistical machine translation. In *Proceedings of ACL-HLT*.

Chang, Ming-Wei, Dan Goldwasser, Dan Roth, and Vivek Srikumar. 2010. Discriminative learning over constrained latent representations. In *Proceedings of NAACL-HLT*.

Clark, S. and J. R. Curran. 2004. Parsing the wsj using ccg and log-linear models. In *Proceedings of ACL*.

Cui, Hang, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of SIGIR*.

Dagan, I., O. Glickman, and B. Magnini. 2006. The pascal recognising textual entailment challenge. *Machine Learning Challenges, LNCS*, 3944:177–190.

Das, Dipanjan and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of ACL-IJCNLP*.

de Marneffe, M.-C., B. MacCartney, T. Grenager, D. Cer, A. Rafferty, and C. D. Manning. 2006. Learning to distinguish valid textual entailments. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

de Salvo Braz, R., R. Girju, V. Punyakanok, D. Roth, and M. Sammons. 2005. An inference model for semantic entailment and question-answering. In *Proceedings of AAAI*.

Finkel, J. R., T. Grenager, and C. D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*.

Finkel, J. R., C. D. Manning, and A. Y. Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of EMNLP*.

Haghighi, A., A. Y. Ng, and C. D. Manning. 2005. Robust textual inference via graph matching. In *Proceedings of EMNLP*.

Harmeling, S. 2007. An extensible probabilistic transformation-based approach to the third recognizing textual entailment challenge. In *Proceedings of ACL PASCAL Workshop on Textual Entailment and Paraphrasing*.

Heilman, M. and N. A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL-HLT*.

Jijkoun, V. and M. de Rijke. 2005a. Recognizing textual entailment: Is word similarity enough?. In *Machine Learning Challenge Workshop*, volume 3944 of *LNCS*, pages 449–460. Springer.

Jijkoun, V. and M. de Rijke. 2005b. Recognizing textual entailment using lexical similarity. In *Proceedings of the PASCAL Challenges Workshop on RTE*.

Klein, P. N. 1989. Computing the edit-distance between unrooted ordered trees. In *Proceedings of European Symposium on Algorithms*.

Kouylekov, M. and B. Magnini. 2006. Tree edit distance for recognizing textual entailment: Estimating the cost of insertion. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

Lafferty, J., A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

MacCartney, Bill and Christopher D. Manning. 2007. Natural logic for textual inference. In *Proceedings of Workshop on Textual Entailment and Paraphrasing at ACL 2007*.

MacCartney, B. and C. D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of COLING*.

MacCartney, B., T. Grenager, M.-C. de Marneffe, D. Cer, and C. D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of HLT-NAACL*.

MacCartney, B., M. Galley, and C. D. Manning. 2008. A phrase-based alignment model for natural language inference. In *Proceedings of EMNLP*.

Marsi, E., E. Krahmer, and W. Bosma. 2007. Dependency-based paraphrasing for recognizing textual entailment. In *Proceedings of ACL PASCAL Workshop on Textual Entailment and Paraphrasing*.

McCallum, A., K. Bellare, and F. Pereira. 2005. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of UAI*.

McDonald, R., K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*.

Mehdad, Yashar. 2009. Automatic cost estimation for tree edit distance using particle swarm optimization. In *Proceedings of ACL*.

Moldovan, D., C. Clark, S. Harabagiu, and S. Maiorano. 2003. Cogex: A logic prover for question answering. In *Proceedings of HLT-NAACL*.

Nielsen, R. D., W. Ward, and J. H. Martin. 2006. Toward dependency path based entailment. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

Papineni, K., S. Roukos, T. Ward, and W. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Petrov, S., A. Pauls, and D. Klein. 2007. Discriminative log-linear grammars with latent variables. In *Proceedings of NIPS*.

Punyakanok, V., D. Roth, and W. Yih. 2004. Mapping dependencies trees: An application to question answering. In *Proceedings of AI-Math*.

Raina, R., A. Y. Ng, , and C. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of AAAI*.

Ratnaparkhi, Adwait. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of EMNLP*.

Smith, D. A. and J. Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*.

Snow, R., L. Vanderwende, and A. Menezes. 2006. Effectively using syntax for recognizing false entailment. In *Proceedings of HLT-NAACL*.

Vanderwende, L., A. Menezes, and R. Snow. 2006. Microsoft research at rte-2: Syntactic contributions in the entailment task: an implementation. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

Wang, M., N. A. Smith, and T. Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for question answering. In *Proceedings of EMNLP-CoNLL*.

Zanzotto, F. M., A. Moschitti, M. Pennacchiotti, and M.T. Pazienza. 2006. Learning textual entailment from examples. In *Proceedings of the second PASCAL Challenges Workshop on RTE*.

Zhang, K. and D. Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal of Computing*, 18.

Zhang, K. 1996. A constrained edit distance between unordered labeled trees. *Algorithmica*, 15(3):205–222.

A Character-Based Joint Model for Chinese Word Segmentation

Kun Wang and **Chengqing Zong**
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Science
{kunwang,cqzong}@nlpr.ia.ac.cn

Keh-Yih Su
Behavior Design Corporation
Kysu@bdc.com.tw

Abstract

The character-based tagging approach is a dominant technique for Chinese word segmentation, and both discriminative and generative models can be adopted in that framework. However, generative and discriminative character-based approaches are significantly different and complement each other. A simple joint model combining the character-based generative model and the discriminative one is thus proposed in this paper to take advantage of both approaches. Experiments on the Second SIGHAN Bakeoff show that this joint approach achieves 21% relative error reduction over the discriminative model and 14% over the generative one. In addition, closed tests also show that the proposed joint model outperforms all the existing approaches reported in the literature and achieves the best F-score in four out of five corpora.

1 Introduction

Chinese word segmentation (CWS) plays an important role in most Chinese NLP applications such as machine translation, information retrieval and question answering. Many statistical methods for CWS have been proposed in the last two decades, which can be classified as either *word-based* or *character-based*. The word-based approach regards the word as the basic unit, and the desired segmentation result is the best word sequence found by the search process. On the other hand, the character-based approach treats the word segmentation task as a character tagging problem. The final segmentation result is thus indirectly generated according to the tag assigned to each associated character. Since the vocabulary size of possible character-tag-pairs is limited, the character-based models can tolerate *out-of-vocabulary* (OOV) words and have become the dominant technique for CWS in recent years.

On the other hand, statistical approaches can also be classified as either adopting a *generative model* or adopting a *discriminative model*. The generative model learns the joint probability of the given input and its associated label sequence, while the discriminative model learns the posterior probability directly. Generative models often do not perform well because they make strong independence assumptions between features and labels. However, (Toutanova, 2006) shows that generative models can also achieve very similar or better performance than the corresponding discriminative models if they have a structure that avoids unrealistic independence assumptions.

In terms of the above dimensions, methods for CWS can be classified as:

1) The word-based generative model (Gao et al., 2003; Zhang et al., 2003), which is a well-known approach and has been used in many successful applications;

2) The word-based discriminative model (Zhang and Clark, 2007), which generates word candidates with both word and character features and is the only word-based model that adopts the discriminative approach；

3) The character-based discriminative model (Xue, 2003; Peng et al., 2004; Tseng et al., 2005; Jiang et al., 2008), which has become the dominant method as it is robust on OOV words and is capable of handling a range of different features, and it has been adopted in many previous works;

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1173–1181,
Beijing, August 2010

4) The character-based generative model (Wang et al., 2009), which adopts a character-tag-pair-based *n*-gram model and achieves comparable results with the popular character-based discriminative model.

In general, character-based models are much more robust on OOV words than word-based approaches do, as the vocabulary size of characters is a closed set (versus the open set of that of words). Furthermore, among those character-based approaches, the generative model and the discriminative one complement each other in handling *in-vocabulary* (IV) words and OOV words. Therefore, a character-based joint model is proposed to combine them.

This proposed joint approach has achieved good balance between IV word recognition and OOV word identification. The experiments of closed tests on the second SIGHAN Bakeoff (Emerson, 2005) show that the joint model significantly outperforms the baseline models of both generative and discriminative approaches. Moreover, statistical significance tests also show that the joint model is significantly better than all those state-of-the-art systems reported in the literature and achieves the best F-score in four of the five corpora tested.

2 Character-Based Models for CWS

The goal of CWS is to find the corresponding word sequence for a given character sequence. Character-based model is to find out the corresponding tags for given character sequence.

2.1 Character-Based Discriminative Model

The character-based discriminative model (Xue, 2003) treats segmentation as a tagging problem, which assigns a corresponding tag to each character. The model is formulated as:

$$P(t_1^n \mid c_1^n) = \prod_{k=1}^{n} P(t_k \mid t_1^{k-1}, c_1^n) \approx \prod_{k=1}^{n} P(t_k \mid c_{k-2}^{k+2}) \qquad (1)$$

Where t_k is a member of {***B**egin*, ***M**iddle*, ***E**nd*, ***S**ingle*} (abbreviated as B, M, E and S from now on) to indicate the corresponding position of character c_k in its associated word. For example, the word “北京市 (Beijing City)” will be assigned with the corresponding tags as: “北/B (North) 京/M (Capital) 市/E (City)”.

Since this tagging approach treats characters as basic units, the vocabulary size of those possible character-tag-pairs is limited. Therefore, this method is robust to OOV words and could possess a high *recall of OOV words* ($\mathrm{R_{OOV}}$). Although the dependency between adjacent tags/labels can be addressed, the dependency between adjacent characters within a word cannot be directly modeled under this framework. Lower *recall of IV words* ($\mathrm{R_{IV}}$) is thus usually accompanied (Wang et al., 2009).

In this work, the character-based discriminative model is implemented by adopting the feature templates given by (Ng and Low, 2004), but excluding those ones that are forbidden by the closed test regulation of SIGHAN (e.g., $Pu(C_0)$: whether C_0 is a punctuation). Those feature templates adopted are listed below:

$(a)\, C_n\, (n = -2, -1, 0, 1, 2);$

$(b)\, C_n C_{n+1}\, (n = -2, -1, 0, 1);$

$(c)\, C_{-1} C_1$

For example, when we consider the third character “奥” in the sequence “北京奥运会”, template (a) results in the features as following: C_{-2}=北, C_{-1}=京, C_0=奥, C_1=运, C_2=会, and template (b) generates the features as: $C_{-2}C_{-1}$=北京, $C_{-1}C_0$=京奥, C_0C_1=奥运, C_1C_2=运会, and template (c) gives the feature $C_{-1}C_1$=京运.

2.2 Character-Based Generative Model

To incorporate the dependency between adjacent characters in the character-based approach, (Wang et al., 2009) proposes a character-based generative model. In this approach, word w_i is first replaced with its corresponding sequence of [character, tag] (denoted as [*c*, *t*]), where tag is the same as that adopted in the above character-based discriminative model. With this representation, this model can be expressed as:

$$\begin{aligned} P(w_1^m \mid c_1^n) &\equiv P([c,t]_1^n \mid c_1^n) \\ &= P(c_1^n \mid [c,t]_1^n) \times P([c,t]_1^n) \big/ P(c_1^n) \end{aligned} \qquad (2)$$

Since $P(c_1^n \mid [c,t]_1^n) \equiv 1$ and $P(c_1^n)$ is the same for various candidates, only $P([c,t]_1^n)$ should be considered. It can be further simplified with Markov Chain assumption as:

$$P([c,t]_1^n) \approx \prod_{i=1}^{n} P([c,t]_i \mid [c,t]_{i-k}^{i-1}). \qquad (3)$$

Compared with the character-based discriminative model, this generative model keeps the capability to handle OOV words because it also regards the character as basic unit. In addition, the dependency between adjacent

宿	Gold and Discriminative Tag: M				Generative Trigram Tag: E					
Tag probability:	B/0.0333		E/0.2236		M/0.7401		S/0.0030			
Feature Tag	C_{-2}	C_{-1}	C_0	$\mathbf{C_1}$	$\mathbf{C_2}$	$C_{-2}C_{-1}$	$C_{-1}C_0$	C_0C_1	C_1C_2	$C_{-1}C_1$
B	-1.4375	0.1572	0.0800	0.2282	0.7709	0.2741	0.0000	0.0000	-0.6718	0.0000
E	1.3558	0.1910	0.7229	**-1.2696**	**-0.5970**	0.0049	0.0921	0.0000	0.8049	0.0000
M	1.1071	-0.5527	-0.3174	**2.9422**	**0.4636**	-0.1708	0.0000	0.0000	-0.9700	0.0000
S	-1.0254	0.2046	-0.4856	-1.9008	-0.6375	0.0000	0.0000	0.0000	0.8368	0.0000
者	Gold and Discriminative Tag: E				Generative Trigram Tag: S					
Tag probability:	B/0.0009		E/0.8138		M/0.0012		S/0.1841			
Feature Tag	C_{-2}	C_{-1}	$\mathbf{C_0}$	$\mathbf{C_1}$	C_2	$C_{-2}C_{-1}$	$C_{-1}C_0$	C_0C_1	C_1C_2	$C_{-1}C_1$
B	0.3586	0.4175	0.0000	-0.7207	0.4626	0.0085	0.0000	0.0000	0.0000	0.0000
E	0.3666	0.0687	**4.5381**	**2.8300**	-0.0846	0.0000	0.0000	-1.0279	0.6127	0.0000
M	-0.5657	-0.4330	1.8847	0.0000	-0.0918	0.0000	0.0000	0.0000	0.0000	0.0000
S	-0.1595	-0.0532	**2.7360**	**1.8223**	-0.2862	-0.0024	0.0000	1.0494	0.7113	0.0000

Table 1: The corresponding lambda weight of features for “露宿者” in the sentence “[該] [處] [的] [露宿者] [只] [有] [數] [人]”. In the Feature column and Tag row, the value is the corresponding lambda weight for the feature and tag under ME framework. The meanings of those features are explained in Section 2.1.

characters is now directly modeled. This will give sharper preference when the history of assignment is given. Therefore, this approach not only holds robust IV performance but also achieves comparable results with the discriminative model. However, the OOV performance of this approach is still lower than that of the discriminative model (see in Table 5), which would be discussed in the next section.

3 Problems with the Character-Based Generative Model

The character-based generative model can handle the dependency between adjacent characters and thus performs well on IV words. However, this generative trigram model is derived under the second order Markov Chain assumption. Future character context (i.e., C_1 and C_2) is thus not utilized in the model when the tag of the current character (i.e., t_0) is determined. Nevertheless, the future context would help to select the correct tag when the associated trigram has not been observed in the training-set, which is just the case for those OOV words. In contrast, the discriminative one could get help from the future context in this case. The example given in the next paragraph clearly shows the above situation.

At the sentence “該(that) 處(place) 的(of) 露宿者(street sleeper) 只(only) 有(have) 數(some) 人(person) (There are only some street sleepers in that place)” in the CITYU corpus, “露/B宿/M者/E(street sleeper)” is observed to be an OOV word, while “露/B宿/E(sleep on the street)” is an IV word, where the associated tag of each character is given after the slash symbol. The character-based generative model wrongly splits “露宿者” into two words “露/B宿/E” and “者/S (person)”, as the associated trigram for “露宿者” is not seen in the training set. However, the discriminative model gives the correct result for “宿/M” and the dominant features come from its future context “者” and “只”. Similarly, the future context “只” helps to give the correct tag to “者/E”. Table 1 gives the corresponding lambda feature weights (under the Maximum Entropy (ME) (Ratnaparkhi, 1998) framework) for “露宿者” in the discriminative model. It shows that in the column of “C_1” below “宿”, the lambda value associated with the correct tag “M” is 2.9422, which is the highest value in that column and is far greater than that of the wrong tag “E” (i.e., -1.2696) assigned by the generative model. Which indicates that the future feature “C_1” is the most useful feature for tagging “宿”.

The above example shows the character-based generative model fails to handle some OOV words such as “露宿者” because this approach cannot utilize future context when it is indeed required. However, the future context for the generative model scanning from left to right is just its past context when it scans from right to left. It is thus expected that this kind of errors will be fixed if we let the model scans from both directions, and then combine their results. Unfortunately, it is observed that these two scanning modes share over 90% of their errors. For example, in CITYU corpus, the left-to-right scan generates 1,958 wrong words and the right-to-left scan results 1,947 ones, while 1,795 of them are the same. Similar behavior can also be observed on other corpora.

To find out what are the problems, 10 errors that are similar to “露宿者” are selected to examine. Among those errors, only one of them is fixed, and “露宿者” still cannot be correctly segmented. Having analyzed the scores of the model scanning from both directions, we found that the original scores (from left-to-right scan) at the stages “者” and “宿” indeed get better if the model scans from right-to-left. However, the score at the stage “露” deteriorates because the useful feature “者” (a past non-adjacent character for “露” when scans form right-to-left) still cannot be utilized when the past context “宿者” as a whole is unseen, when the related probabilities are estimated via modified Kneser-Ney smoothing (Chen and Goodman, 1998) technique.

Two scanning modes seem not complementing each other, which is out of our original expectation. However, we found that the character-based generative model and the discriminative one complement each other much more than the two scanning modes do. It is observed that these two approaches share less than 50% of their errors. For example, in CITYU corpus, the generative approach generates 1,958 wrong words and the discriminative one results 2,338 ones, while only 835 of them are the same.

The statistics of the remaining errors resulted from the generative model and the discriminative model is shown in Table 2. As shown in the table, it can be seen that the generative model and the discriminative model complement each other on handling IV words and OOV words (In the “IV Errors” column, the number of “G+D-” is much more than the “G-D+”, while the behavior is reversed in the “OOV Errors” column).

IV Errors			OOV Errors		
G+D-	G-D+	G-D-	G+D-	G-D+	G-D-
12,027	4,723	7,481	2,384	6,139	3,975

Table 2: Statistics for remaining errors of the character-based generative model and the discriminative one on the second SIGHAN Bakeoff (“G+D-” in the “IV Errors” column means that the generative model segments the IV words correctly but the discriminative one gives wrong results. The meanings of other abbreviations are similar with this one.).

4 Proposed Joint Model

Since the performance of both IV words and OOV words are important for real applications, we need to combine the strength from both models. Among various combining methods, log-linear interpolation combination is a simple but effective one (Bishop, 2006). Therefore, the following character-based joint model is proposed, and a parameter α is used to weight the generative model in a cross-validation set.

$$\begin{aligned} Score(t_k) &= \alpha \times \log(P([c,t]_k \,\big|\, [c,t]_{k-2}^{k-1})) \\ &\quad + (1-\alpha) \times \log(P(t_k \,\big|\, c_{k-2}^{k+2})) \end{aligned} \tag{4}$$

Where t_k indicates the corresponding position of character c_k, and α $(0.0 \le \alpha \le 1.0)$ is the weight for the generative model. $Score(t_k)$ will be used during searching the best sequence. It can be seen that these two models are integrated naturally as both are character-based.

Generally speaking, if the “G(or D)+” has a strong preference on the desired candidate, but the “D(or G)-” has a weak preference on its top-1 incorrect candidate, then this combining method would correct most “G+D- (also G-D+)” errors. On the other hand, the advantage of combining two models would vanish if the “G(or D)+” has a weak preference while the “D(or G)-” has a strong preference over their top-1 candidates. In our observation, these two models meet this requirement quite well.

5 Weigh Various Features Differently

For a given observation, intuitively each feature should be trained only once under the ME framework and its associated weight will be automatically learned from the training corpus. However, when we repeat the work of (Jiang et al., 2008), which reports to achieve the state-of-art performance in the data-sets that we adopt, it has been found that some features (e.g., C_0) are unnoticeably trained several times in their model (which are implicitly generated from different feature templates used in the paper). For example, the feature C_0 actually

Corpus	Abbrev.	Encoding	Training Size (Words/Type)	Test Size (Words/Type)	OOV Rate
Academia Sinica (Taipei)	AS	Unicode/Big5	5.45M/141K	122K/19K	0.046
City University of Hong Kong	CITYU	Unicode/Big5	1.46M/69K	41K/9K	0.074
Microsoft Research (Beijing)	MSR	Unicode/CP936	2.37M/88K	107K/13K	0.026
Peking University	PKU(ucvt.)	Unicode/CP936	1.1M/55K	104K/13K	0.058
	PKU(cvt.)	Unicode/CP936	1.1M/55K	104K/13K	0.035

Table 3: Corpus statistics for the second SIGHAN Bakeoff

appears twice, which is generated from two different templates C_n (with n=0, generates C_0) and [C_0C_n] (used in (Jiang et al., 2008), with n=0, generates [C_0C_0]). The meanings of features are illustrated in Section 2.1. Those repetitive features also include [$C_{-1}C_0$] and [C_0C_1], which implicitly appear thrice. And it is surprising to discover that its better performance is mainly due to this implicit feature repetition but the authors do not point out this fact. As all the features adopted in (Jiang et al., 2008) possess binary values, if a binary feature is repeated *n* times, then it should behave like a real-valued feature with its value to be "*n*", at least in principle. Inspired by the above discovery, accordingly, we convert all the binary-value features into their corresponding real-valued features. After having transformed binary features into their corresponding real-valued ones, the original discriminative model is re-trained under the ME framework.

This new implementation, which would be named as the character-based discriminative-plus model, just weights various features differently before conducting ME training. Afterwards, it is further combined with the generative trigram model, and is called the character-based joint-plus model.

6 Experiments

The corpora provided by the second SIGHAN Bakeoff (Emerson, 2005) were used in our experiments. The statistics of those corpora are shown in Table 3.

Note that the PKU corpus is a little different from others. In the training set, Arabic numbers and English characters are in full-width form occupying two bytes. However, in the testing set, these characters are in half-width form occupying only one byte. Most researchers in the SIGHAN Bakeoff competition performed a conversion before segmentation (Xiong et al., 2009). In this work, we conduct the tests on both unconverted (ucvt.) case and converted (cvt.) case. After the conversion, the OOV rate of converted corpus is obviously lower than that of unconverted corpus.

To fairly compare the proposed approach with previous works, we only conduct *closed tests*[1]. The metrics *Precision* (**P**), *Recall* (**R**), *F-score* (**F**) (*F*=2*PR*/(*P*+*R*)), *Recall of OOV* ($\mathbf{R_{OOV}}$) and *Recall of IV* ($\mathbf{R_{IV}}$) are used to evaluate the results.

6.1 Character-Based Generative Model and Discriminative Model

As shown in (Wang et al., 2009), the character-based generative trigram model significantly exceeds its related bigram model and performs the same as its 4-gram model. Therefore, SRI Language Modeling Toolkit[2] (Stolcke, 2002) is used to train the trigram model with modified Kneser-Ney smoothing (Chen and Goodman, 1998). Afterwards, a beam search decoder is applied to find out the best sequence.

For the character-based discriminative model, the ME Package[3] given by Zhang Le is used to conduct the experiments. Training was done with Gaussian prior 1.0 and 300, 150 iterations for AS and other corpora respectively. Table 5 gives the segmentation results of both the character-based generative model and the discriminative model. From the results, it can be seen that the generative model achieves comparable results with the discriminative one and they outperform each other on different corpus. However, the generative model exceeds the discriminative one on R_{IV} (0.973 vs. 0.956) but loses on R_{OOV} (0.511 vs. 0.680). It illustrates that they complement each other.

[1] According to the second Sighan Bakeoff regulation, the closed test could only use the training data directly provided. Any other data or information is forbidden, including the knowledge of characters set, punctuation set, etc.

[2] http://www.speech.sri.com/projects/srilm/

[3] http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html

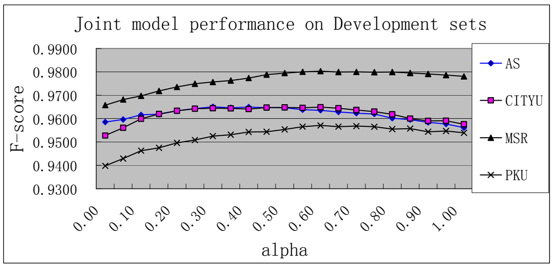


Figure 1: Development sets performance of Character-based joint model.

Corpus	Set	Words	OOV Num	OOV Rate
AS	Development	17,243	445	0.026
	Testing	122,610	5,308/5,311	0.043/0.043
MSR	Development	17,324	355	0.020
	Testing	106,873	2,829/2,833	0.026/0.027
CITYU	Development	12,075	537	0.044
	Testing	40,936	3,028/3,034	0.074/0.074
PKU	Development	13,576	532	0.039
	Testing (ucvt.)	104,372	6,006/6,054	0.058/0.058
	Testing (cvt.)	104,372	3,611/3,661	0.035/0.035

Table 4: Corpus statistics for Development sets and Testing sets. A "/" separates the OOV number (or OOV rate) with respect to the original training sets and the new training sets.

6.2 Character-Based Joint Model

For the character-based joint model, a development set is required to obtain the weight α for its associated generative model. A small portion of each original training corpus is thus extracted as the development set and the remaining data is regarded as the new training-set, which is used to train two new parameter-sets for both generative and discriminative models associated.

The last 2,000, 600, 400, and 300 sentences for AS, MSR, CITYU, and PKU are extracted from the original training corpora as their corresponding development sets. The statistics for new data sets are shown in Table 4. It can be seen that the variation of the OOV rate could be hardly noticed. The F-scores of the joint model, versus different α, evaluated on four development sets are shown in Figure 1. It can be seen that the curves are not sharp but flat near the top, which indicates that the character-based joint model is not sensitive to the α value selected. From those curves, the best suitable α for AS, CITYU, MSR and PKU are found to be 0.30, 0.60, 0.60 and 0.60, respectively. Those alpha values will then be adopted to conduct the experiments on the testing sets.

Corpus	Model	R	P	F	R_{OOV}	R_{IV}
AS	G	0.958	0.938	0.948	0.518	0.978
	D	0.955	0.946	0.951	0.707	0.967
	D-Plus	0.960	0.948	0.954	0.680	0.973
	J	0.962	0.950	**0.956**	0.679	0.975
	J-Plus	0.963	0.949	**0.956**	0.652	0.977
CITYU	G	0.951	0.937	0.944	0.609	0.978
	D	0.941	0.944	0.942	0.708	0.959
	D-Plus	0.951	0.952	0.952	0.720	0.970
	J	0.957	0.951	0.954	0.691	0.979
	J-Plus	0.959	0.952	**0.956**	0.700	0.980
MSR	G	0.974	0.967	0.970	0.561	0.985
	D	0.957	0.962	0.960	0.719	0.964
	D-Plus	0.965	0.967	0.966	0.675	0.973
	J	0.974	0.971	**0.972**	0.659	0.983
	J-Plus	0.975	0.970	**0.972**	0.632	0.984
PKU (ucvt.)	G	0.929	0.933	0.931	0.435	0.959
	D	0.922	0.941	0.932	0.620	0.941
	D-Plus	0.934	0.949	0.941	0.649	0.951
	J	0.935	0.946	0.941	0.561	0.958
	J-Plus	0.937	0.947	**0.942**	0.556	0.960
PKU (cvt.)	G	0.952	0.951	0.952	0.503	0.968
	D	0.940	0.951	0.946	0.685	0.949
	D-Plus	0.949	0.958	0.953	0.674	0.958
	J	0.954	0.958	0.956	0.616	0.966
	J-Plus	0.955	0.958	**0.957**	0.610	0.967
Overall	G	0.953	0.946	0.950	0.511	0.973
	D	0.944	0.950	0.947	0.680	0.956
	D-Plus	0.952	0.955	0.953	0.676	0.965
	J	0.957	0.955	0.956	0.633	0.971
	J-Plus	0.958	0.955	**0.957**	0.621	0.973

Table 5: Segmentation results of various character-based models on the second SIGHAN Bakeoff, the generative trigram model (G), the discriminative model (D), the discriminative-plus model (D-Plus), the joint model (J) and the joint-plus model (J-Plus).

As shown in Table 5, the joint model significantly outperforms both the character-based generative model and the discriminative one in F-score on all the testing corpora. Compared with the generative approach, the joint model increases the overall R_{OOV} from 0.510 to 0.633, with the cost of slightly degrading the overall R_{IV} from 0.973 to 0.971. This shows that the joint model holds the advantage of the generative model on IV words. Compared with the discriminative model, the proposed joint model improves the overall R_{IV} from 0.956 to 0.971, with the cost of degrading the overall R_{OOV} from 0.680 to 0.633. It clearly shows that the joint model achieves a good balance between IV words and OOV words and achieves the best F-scores obtained so far (21% relative error reduction over the discriminative model and 14% over the generative model).

6.3 Weigh Various Features Differently

Inspired by (Jiang et al., 2008), we set the real-value of C_0 to be 2.0, the value of $C_{-1}C_0$ and C_0C_1 to be 3.0, and the values of all other features to be 1.0 for the character-based discriminative-plus model. Although it seems reasonable to weight those closely relevant features more (C_0 should be the most relevant feature for assigning tag t_0), both implementations seem to be equal if their corresponding lambda-values are also updated accordingly. However, Table 5 shows that this new discriminative-plus implementation (D-Plus) significantly outperforms the original one (overall F-score is raised from 0.947 to 0.953) when both of them adopt real-valued features. It is not clear how this change makes the difference.

Similar improvements can be observed with two other ME packages. One anonymous reviewer pointed out that the duplicated features should not make difference if there is no regularization. However, we found that the duplicated features would improve the performance whether we give Gaussian penalty or not.

Afterwards, this new implementation and the generative trigram model are further combined (named as the joint-plus model). Table 5 shows that this joint-plus model also achieves better results compared with the discriminative-plus model, which illustrates that our joint approach is an effective and robust method for CWS. However, compared with the original joint model, the new joint-plus approach does not show much improvement, regardless of the significant improvement made by the discriminative-plus model, as the additional benefit generated by the discriminative-plus model has already covered by the generative approach (Among the 6,965 error words corrected by the discriminative-plus model, 6,292 (90%) of them are covered by the generative model).

7 Statistical Significance Tests

Although Table 5 has shown that the proposed joint (joint-plus) model outperforms all the baselines mentioned above, we want to know if the difference is statistically significant enough to make such a claim. Since there is only one testing set for each training corpus, the bootstrapping technique (Zhang et al., 2004) is adopted to conduct the tests: Giving an testing-set T_0, additional *M-1* new testing-sets $T_0,\ldots,T_{M-1}$ (each with the same size of T_0) will be generated by repeatedly re-sampling data from T_0. Then, we will have a total of M testing-sets (M=2000 in our experiments).

Models		**AS**	**CITYU**	**MSR**	**PKU**	**PKU**
A	**B**				**(ucvt.)**	**(cvt.)**
G	D	<	~	>	~	>
D-Plus	G	>	>	<	>	>
D-Plus	D	>	>	>	>	>
J	G	>	>	>	>	>
J	D	>	>	>	>	>
J-Plus	G	>	>	>	>	>
J-Plus	D-Plus	>	>	>	~	>
J-Plus	J	~	>	~	>	>

Table 6: Statistical significance test of F-score among various character-based models.

7.1 Comparisons with Baselines

We then follow (Zhang et al., 2004) to measure the 95% confidence interval for the discrepancy between two models. If the confidence interval does not include the origin point, we then claim that system A is significantly different from system B. Table 6 gives the results of significant tests among various models mentioned above. In this table, “>” means that system A is significantly better than B, where as “<” denotes that system A is significantly worse than B, and “~” indicates that these two systems are not significantly different.

As shown in Table 6, the proposed joint model is significantly better than the two baseline models on all corpora. Similarly, the proposed joint-plus model also significantly outperforms the generative model and the discriminative-plus model on all corpora except on the PKU(ucvt.). The comparison shows that the proposed joint (also joint-plus) model indeed exceeds each of its component models.

7.2 Comparisons with Previous Works

The above comparison mainly shows the superiority of the proposed joint model among those approaches that have been implemented. However, it would be interesting to know if the joint (and joint-plus) model also outperforms those previous state-of-the-art systems.

The systems that performed best for at least one corpus in the second SIGHAN Bakeoff are first selected for comparison. This category includes (Asahara et al., 2005) (denoted as

Asahara05) and (Tseng et al., 2005) [4] (**Tseng05**). (Asahara et al., 2005) achieves the best result in the AS corpus, and (Tseng et al., 2005) performs best in the remaining three corpora. Besides, those systems that are reported to exceed the above two systems are also selected. This category includes (Zhang et al., 2006) (**Zhang06**), (Zhang and Clark, 2007) (**Z&C07**) and (Jiang et al., 2008) (**Jiang08**). They are briefly summarized as follows. (Zhang et al., 2006) is based on sub-word tagging and uses a confidence measure method to combine the sub-word CRF (Lafferty et al., 2001) and rule-based models. (Zhang and Clark, 2007) uses perceptron (Collins, 2002) to generate word candidates with both word and character features. Last, (Jiang et al., 2008)[5] adds repeated features implicitly based on (Ng and Low, 2004). All of the above models, except (Zhang and Clark, 2007), adopt the character-based discriminative approach.

All the results of the systems mentioned above are shown in Table 7. Since the systems are not re-implemented, we cannot generate paired samples from those *M* testing-sets. Instead, we calculate the 95% confidence interval of the joint (also joint-plus) model. Afterwards, those systems can be compared with our proposed models. If the F-score of system B does not fall within the 95% confidence interval of system A (joint or joint-plus), then they are statistically significantly different.

Table 8 gives the results of significant tests for those systems mentioned in this section. It shows that both our joint-plus model and joint model exceed (or are comparable to) almost all the state-of-the-art systems across all corpora, except (Zhang and Clark, 2007) at PKU(ucvt.). In that special case, (Zhang and Clark, 2007) outperforms the joint-plus model by 0.3% on F- score (0.4% for the joint model). However, our joint-plus model exceeds it more over AS and CITYU corpora by 1.0% and 0.5%, respectively (1.0% and 0.3% for the joint model). Thus, it is fair to say that both our joint model and joint-plus model are superior to the state-of-the-art systems reported in the literature.

Corpus **Participants**	**AS**	**CITYU**	**MSR**	**PKU (ucvt.)**	**PKU (cvt.)**
Asahara05	0.952	0.941	0.958	N/A	0.941
Tseng05	0.947	0.943	0.964	N/A	0.950
Zhang06	0.951	0.951	0.971	N/A	0.951
Z&C07	0.946	0.951	**0.972**	**0.945**	N/A
Jiang08	0.953	0.948	0.966	0.937	N/A
Our Joint	**0.956**	0.954	**0.972**	0.941	0.956
Our Joint-Plus	**0.956**	**0.956**	**0.972**	0.942	**0.957**

Table 7: Comparisons of F-score with previous state-of-the-art systems.

Systems		**AS**	**CITYU**	**MSR**	**PKU (ucvt.)**	**PKU (cvt.)**
A	**B**					
J	Asahara05	>	>	>	N/A	>
	Tseng05	>	>	>	N/A	>
	Zhang06	>	~	~	N/A	>
	Z&C07	>	>	~	<	N/A
	Jiang08	>	>	>	>	N/A
J-Plus	Asahara05	>	>	>	N/A	>
	Tseng05	>	>	>	N/A	>
	Zhang06	>	>	~	N/A	>
	Z&C07	>	>	~	<	N/A
	Jiang08	~	>	>	>	N/A

Table 8: Statistical significance test of F-score for previous state-of-the-art systems.

8 Conclusion

From the error analysis of the character-based generative model and the discriminative one, we found that these two models complement each other on handling IV words and OOV words. To take advantage of these two approaches, a joint model is thus proposed to combine them. Experiments on the Second SIGHAN Bakeoff show that the joint model achieves 21% error reduction over the discriminative model (14% over the generative model). Moreover, closed tests on the second SIGHAN Bakeoff corpora show that this joint model significantly outperforms all the state-of-the-art systems reported in the literature.

Last, it is found that weighting various features differently would give better result. However, further study is required to find out the true reason for this strange but interesting phenomenon.

[4] We are not sure whether (Asahara et al., 2005) and (Tseng et al., 2005) performed a conversion before segmentation in PKU corpus. In this paper, we followed previous works, which cited and compared with them.

[5] The data for (Jiang et al., 2008) given at Table 7 are different from what were reported at their paper. In the communication with the authors, it is found that the script for evaluating performance, provided by the SIGHAN Bakeoff, does not work correctly in their platform. After the problem is fixed, the re-evaluated real performances reported here deteriorate from their original version. Please see the announcement in Jiang's homepage (http://mtgroup.ict.ac.cn/~jiangwenbin/papers/error_correction.pdf).

Acknowledgement

The authors extend sincere thanks to Wenbing Jiang for his helps with our experiments. Also, we thank Behavior Design Corporation for using their Generic-Beam-Search code and show special thanks to Ms. Nanyan Kuo for her helps with the Generic-Beam-Search code.

The research work has been partially funded by the Natural Science Foundation of China under Grant No. 60975053, 90820303 and 60736014, the National Key Technology R&D Program under Grant No. 2006BAH03B02, and also the Hi-Tech Research and Development Program ("863" Program) of China under Grant No. 2006AA010108-4 as well.

References

Masayuki Asahara, Kenta Fukuoka, Ai Azuma, Chooi-Ling Goh, Yotaro Watanabe, Yuji Matsumoto and Takashi Tsuzuki, 2005. Combination of machine learning methods for optimum Chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 134–137, Jeju, Korea.

Christopher M. Bishop, 2006. *Pattern recognition and machine learning*. New York: Springer

Stanley F. Chen and Joshua Goodman, 1998. An empirical study of smoothing techniques for language modeling. *Technical Report TR-10-98, Harvard University Center for Research in Computing Technology*.

Michael Collins, 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1-8, Philadelphia.

Thomas Emerson, 2005. The second international Chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 123-133.

Jianfeng Gao, Mu Li and Chang-Ning Huang, 2003. Improved Source-Channel Models for Chinese Word Segmentation. In *Proceedings of ACL*, pages 272-279.

Wenbin Jiang, Liang Huang, Qun Liu and Yajuan Lu, 2008. A Cascaded Linear Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging. In *Proceedings of ACL*, pages 897-904.

John Lafferty, Andrew McCallum and Fernando Pereira, 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of ICML*, pages 282-289.

Hwee Tou Ng and Jin Kiat Low, 2004. Chinese part-of-speech tagging: one-at-a-time or all-at-once? word-based or character-based. In *Proceedings of EMNLP*, pages 277-284.

Fuchun Peng, Fangfang Feng and Andrew McCallum, 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of COLING*, pages 562–568.

Adwait Ratnaparkhi, 1998. Maximum entropy models for natural language ambiguity resolution. University of Pennsylvania.

Andreas Stolcke, 2002. SRILM-an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 311-318.

Kristina Toutanova, 2006. Competitive generative models with structure learning for NLP classification tasks. In *Proceedings of EMNLP*, pages 576-584, Sydney, Australia.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky and Christopher Manning, 2005. A Conditional Random Field Word Segmenter for Sighan Bakeoff 2005. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 168-171.

Kun Wang, Chengqing Zong and Keh-Yih Su, 2009. Which is more suitable for Chinese word segmentation, the generative model or the discriminative one? In *Proceedings of PACLIC*, pages 827-834, Hong Kong, China.

Ying Xiong, Jie Zhu, Hao Huang and Haihua Xu, 2009. Minimum tag error for discriminative training of conditional random fields. *Information Sciences*, 179 (1-2). pages 169-179.

Nianwen Xue, 2003. Chinese Word Segmentation as Character Tagging. *Computational Linguistics and Chinese Language Processing*, 8 (1). pages 29-48.

Huaping Zhang, Hongkui Yu, Deyi Xiong and Qun Liu, 2003. HHMM-based Chinese lexical analyzer ICTCLAS. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 184–187.

Ruiqiang Zhang, Genichiro Kikui and Eiichiro Sumita, 2006. Subword-based Tagging for Confidence-dependent Chinese Word Segmentation. In *Proceedings of the COLING/ACL*, pages 961-968, Sydney, Australia.

Ying Zhang, Stephan Vogel and Alex Waibel, 2004. Interpreting BLEU/NIST scores: How much improvement do we need to have a better system. In *Proceedings of LREC*, pages 2051–2054.

Yue Zhang and Stephen Clark, 2007. Chinese Segmentation with a Word-Based Perceptron Algorithm. In *Proceedings of ACL*, pages 840-847, Prague, Czech Republic.

Near-synonym Lexical Choice in Latent Semantic Space

Tong Wang
Department of Computer Science
University of Toronto
tong@cs.toronto.edu

Graeme Hirst
Department of Computer Science
University of Toronto
gh@cs.toronto.edu

Abstract

We explore the near-synonym lexical choice problem using a novel representation of near-synonyms and their contexts in the latent semantic space. In contrast to traditional latent semantic analysis (LSA), our model is built on the lexical level of co-occurrence, which has been empirically proven to be effective in providing higher dimensional information on the subtle differences among near-synonyms. By employing supervised learning on the latent features, our system achieves an accuracy of 74.5% in a "fill-in-the-blank" task. The improvement over the current state-of-the-art is statistically significant.

We also formalize the notion of *subtlety* through its relation to semantic space dimensionality. Using this formalization and our learning models, several of our intuitions about subtlety, dimensionality, and context are quantified and empirically tested.

1 Introduction

Lexical choice is the process of selecting content words in language generation. Consciously or not, people encounter the task of lexical choice on a daily basis — when speaking, writing, and perhaps even in inner monologues. Its application also extends to various domains of natural language processing, including Natural Language Generation (NLG, Inkpen and Hirst 2006), writers' assistant systems (Inkpen, 2007), and second language (L2) teaching and learning (Ouyang et al., 2009).

In the context of *near-synonymy*, the process of lexical choice becomes profoundly more complicated. This is partly because of the subtle nuances among near-synonyms, which can arguably differ along an infinite number of dimensions. Each dimension of variation carries differences in style, connotation, or even truth conditions into the discourse in question (Cruse, 1986), all making the seemingly intuitive problem of "choosing the right word for the right context" far from trivial even for native speakers of a language. In a widely-adopted "fill-in-the-blank" task, where the goal was to guess missing words (from a set of near-synonyms) in English sentences, two human judges achieved an accuracy of about 80% (Inkpen, 2007). The current state-of-the-art accuracy for an automated system is 69.9% (Islam and Inkpen, 2010).

When the goal is to make plausible or even elegant lexical choices that best suit the *context*, the representation of that context becomes a key issue. We approach this problem in the *latent semantic space*, where transformed local co-occurrence data is capable of implicitly inducing global knowledge (Landauer and Dumais, 1997). A latent semantic space is constructed by reducing the dimensionality of co-occurring linguistic units — typically words and documents as in *Latent Semantic Analysis* (LSA). We refer to this *level of association* (LoA) as *document LoA* hereafter. Although document LoA can benefit topical level classification (e.g., as in document retrieval, Deerwester et al. 1990), it is not necessarily suitable for lexical-level tasks which might require information on a more fine-grained level (Edmonds and Hirst, 2002). Our experimental results show

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1182–1190,
Beijing, August 2010

noticeable improvement when the co-occurrence matrix is built on a lexical LoA between words within a given context window.

One intuitive explanation for this improvement is that the lexical-level co-occurrence might have helped recover the high-dimensional subtle nuances between near-synonyms. This conjecture is, however, as imprecise as it is intuitive. The notion of *subtlety* has mostly been used qualitatively in the literature to describe the level of difficulty involved in near-synonym lexical choice. Hence, we endeavor to formalize the concept of subtlety computationally by using our observations regarding the relationship between "subtle" concepts and their lexical co-occurrence patterns.

We introduce related work on near-synonymy, lexical choice, and latent semantic space models in the next section. Section 3 elaborates on lexical and contextual representations in latent semantic space. In Section 4, we formulate near-synonym lexical choice as a learning problem and report our system performance. Section 5 formalizes the notion of subtlety and its relation to dimensionality and context. Conclusions and future work are presented in Section 6.

2 Related Work

2.1 Near-Synonymy and Nuances

Near-synonymy is a concept better explained by intuition than by definition — which it does not seem to have in the existing literature. We thus borrow Table 1 from Edmonds and Hirst (2002) to illustrate some basic ideas about near-synonymy. Cruse (1986) compared the notion of *plesionymy* to cognitive synonymy in terms of mutual entailment and semantic traits, which, to the best of our knowledge, is possibly the closest to a textbook account of near-synonymy.

There has been a substantial amount of interest in characterizing the nuances between near-synonyms for a computation-friendly representation of near-synonymy. DiMarco et al. (1993) discovered 38 dimensions for differentiating near-synonyms from dictionary usage notes and categorized them into semantic and stylistic variations. Stede (1993) focused on the latter and further decomposed them into seven scalable subcategories. By organizing near-synonym variations into a tree structure, Inkpen and Hirst (2006) combined stylistic and attitudinal variation into one class parallel to denotational differences. They also incorporated this knowledge of near-synonyms into a knowledge base and demonstrated its application in an NLG system.

Table 1: Examples of near-synonyms and dimension of variations (Edmonds and Hirst, 2002).

Types of variation	***Examples***
Continuous, intermittent	seep:drip
Emphasis	enemy:foe
Denotational, indirect	error:mistake
Denotational, fuzzy	woods:forest
Stylistic, formality	pissed:drunk:inebriated
Stylistic, force	ruin:annihilate
Expressed attitude	skinny:thin:slim:slender
Emotive	daddy:dad:father
Collocational	task:job
Selectional	pass away:die
Sub-categorization	give:donate

2.2 Lexical Choice Evaluation

Due to their symbolic nature, many of the early studies were only able to provide "demo runs" in NLG systems rather than any empirical evaluation. The study of near-synonym lexical choice had remained largely qualitative until a "fill-in-the-blank" (FITB) task was introduced by Edmonds (1997). The task is based on sentences collected from the 1987 *Wall Street Journal* (WSJ) that contain any of a given set of near-synonyms. Each occurrence of the near-synonyms is removed from the sentence to create a "lexical gap", and the goal is to guess which one of the near-synonyms is the missing word. Presuming that the 1987 WSJ authors have made high-quality lexical choices, the FITB test provides a fairly objective benchmark for empirical evaluation for near-synonym lexical choice. The same idea can be applied to virtually any corpus to provide a fair amount of gold-standard data at relatively low cost for lexical choice evaluation.

The FITB task has since been frequently adopted for evaluating the quality of lexical choice systems on a standard dataset of seven near-synonym sets (as shown in Table 2). Edmonds

(1997) constructed a second-order lexical co-occurrence network on a training corpus (the 1989 WSJ). He measured the *word-word distance* using *t-score* inversely weighted by both distance and order of co-occurrence in the network. For a sentence in the test data (generated from the 1987 WSJ), the candidate near-synonym minimizing the sum of its distance from all other words in the sentence (*word-context distance*) was considered the correct answer. Average accuracy on the standard seven near-synonym sets was 55.7%.

Inkpen (2007) modeled word-word distance using *Pointwise Mutual Information* (PMI) approximated by word counts from querying the *Waterloo Multitext System* (Clarke et al., 1998). Word-context distance was the sum of PMI scores between a candidate and its neighboring words within a window-size of 10. An unsupervised model using word-context distance directly achieved an average accuracy of 66.0%, while a supervised method with lexical features added to the word-context distance further increased the accuracy to 69.2%.

Islam and Inkpen (2010) developed a system which completed a test sentence with possible candidates one at a time. The candidate generating the most probable sentence (measured by a 5-gram language model) was proposed as the correct answer. N-gram counts were collected from *Google Web1T Corpus* and smoothed with *missing counts*, yielding an average accuracy of 69.9%.

2.3 Lexical Choice Outside the Near-synonymy Domain

The problem of lexical choice also comes in many flavors outside the near-synonymy domain. Reiter and Sripada (2002) attributed the variation in lexical choice to cognitive and vocabulary differences among individuals. In their meteorology domain data, for example, the term *by evening* was interpreted as *before 00:00* by some forecasters but *before 18:00* by others. They claimed that NLG systems might have to include redundancy in their output to tolerate cognitive differences among individuals.

2.4 Latent Semantic Space Models and LoA

LSA has been widely applied in various fields since its introduction by Landauer and Dumais (1997). In their study, LSA was conducted on *document* LoA on encyclopedic articles and the latent space vectors were used for solving TOEFL synonym questions. Rapp (2008) used LSA on *lexical* LoA for the same task and achieved 92.50% in accuracy in contrast to 64.38% given by Landauer and Dumais (1997). This work confirmed our early postulation that document LoA might not be tailored for lexical level tasks, which might require lower LoAs for more fine-grained co-occurrence knowledge. Note, however, that confounding factors might also have led to the difference in performance, since the two studies used different weighting schemes and different corpora for the co-occurrence model[1]. In Section 3.2 we will compare models on the two LoAs in a more controlled setting to show their difference in the lexical choice task.

3 Representing Words and Contexts in Latent Semantic Space

We first formalize the FITB task to facilitate later discussions. A test sentence $t = \{w_1, \ldots, w_{j-1}, s_i, w_{j+1}, \ldots, w_m\}$ contains a near-synonym s_i which belongs to a set of synonyms $S = \{s_1, \ldots, s_n\}, 1 \leq i \leq n$. A FITB test case is created by removing s_i from t, and the *context* (the incomplete sentence) $c = t - \{s_i\}$ is presented to subjects with a set of possible choices S to guess which of the near-synonyms in S is the missing word.

3.1 Constructing the Latent Space Representation

The first step in LSA is to build a *co-occurrence matrix M* between words and documents, which is further decomposed by *Singular Value Decomposition* (SVD) according to the following equation:

$$M_{v\times d} = U_{v\times k}\Sigma_{k\times k}V^T_{k\times d}$$

[1] The former used *Groliers Academic American Encyclopedia* with weights divided by word entropy, while the latter used the *British National Corpus* with weights multiplied by word entropy.

Here, subscripts denote matrix dimensions, U, Σ, and V together create a decomposition of M, v and d are the number of word types and documents, respectively, and k is the number of dimensions for the latent semantic space. A word w is represented by the row in U corresponding to the row for w in M. For a context c, we construct a vector $\mathbf{c}$ of length v with zeros and ones, each corresponding to the presence or absence of a word w_i with respect to c, i.e.,

$$\mathbf{c}_i = \begin{cases} 1 & \text{if } w_i \in c \\ 0 & \text{otherwise} \end{cases}$$

We then take this *lexical space* vector $\mathbf{c}_{v\times 1}$ as a *pseudo-document* and transform it into a *latent semantic space* vector $\hat{c}$:

$$\hat{\mathbf{c}} = \Sigma^{-1} U^T \mathbf{c} \qquad (1)$$

An important observation is that this representation is equivalent to a *weighted centroid* of the context word vectors: when $\mathbf{c}$ is multiplied by $\Sigma^{-1}U^T$ in Equation (1), the product is essentially a weighted sum of the rows in U corresponding to the context words. Consequently, simple modifications on the weighting can yield other interesting representations of context. Consider, for example, the weighting vector $\mathbf{w}_{k\times 1} = (\sigma_1, \cdots, \sigma_k)^T$ with

$$\sigma_i = \frac{1}{|2(p_{\text{gap}} - i) - 1|}$$

where p_{gap} is the position of the "gap" in the test sentence. Multiplying $\mathbf{w}$ before Σ^{-1} in Equation (1) is equivalent to giving the centroid gradient-decaying weights with respect to the distance between a context word and the near-synonym. This is a form of a *Hyperspace Analogue to Language* (HAL) model, which is sensitive to word order, in contrast to a *bag-of-words* model.

3.2 Dimensionality and Level of Association

The number of dimensions k is an important choice to make in latent semantic space models. Due to the lack of any principled guideline for doing otherwise, we conducted a brute force grid search for a proper k value for each LoA, on the basis of the performance of the unsupervised model (Section 4.1 below).

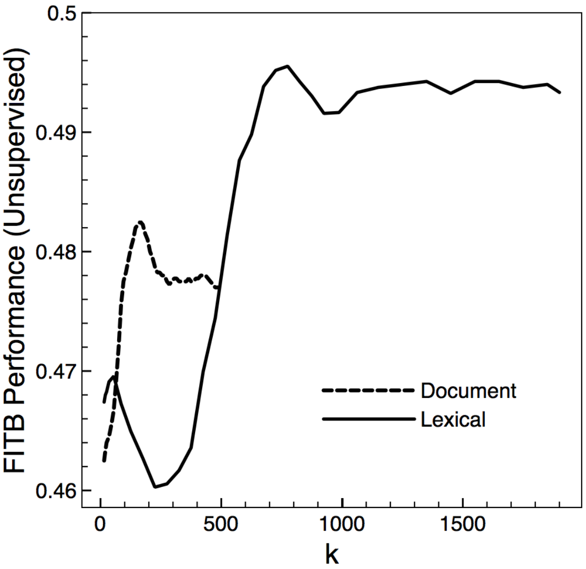


Figure 1: FITB Performance on different LoAs as a function of the latent space dimensionality.

In Figure 1, performance on FITB using this unsupervised model is plotted against k for document and lexical LoAs. Document LoA is very limited in the available number of dimensions[2]; higher dimensional knowledge is simply unavailable from this level of co-occurrence. In contrast, lexical LoA stands out around $k = 550$ and peaks around $k = 700$. Although the advantage of lexical LoA in the unsupervised setting is not significant, later we show that lexical LoA nonetheless makes higher-dimensional information available for other learning methods.

Note that the scale on the y-axis is stretched to magnify the trends. On a zero-to-one scale, the performance of these unsupervised methods is almost indistinguishable, indicating that the unsupervised model is not capable of using the high-dimensional information made available by lexical LoA. We will elaborate on this point in Section 5.2.

[2] The dimensions for document and lexical LoAs on our development corpus are $55,938 \times 500$ and $55,938 \times 55,938$, respectively. The difference is measured between $v \times d$ and $v \times v$ (Section 3.1).

4 Learning in the Latent Semantic Space

4.1 Unsupervised Vector Space Model

When measuring distance between vectors, LSA usually adopts regular vector space model distance functions such as *cosine similarity*. With the context being a centroid of words (Section 3.1), the FITB task then becomes a *k-nearest neighbor* problem in the latent space with $k = 1$ to choose the best near-synonym for the context:

$$s^* = \underset{s_i}{\operatorname{argmax}} \cos(U_{\text{rowId}(\mathbf{v}(s_i),M)}, \hat{\mathbf{c}})$$

where $\mathbf{v}(s_i)$ is the corresponding row for near-synonym s_i in M, and $\text{rowId}(\mathbf{v}, M)$ gives the row number of a vector $\mathbf{v}$ in a matrix M containing $\mathbf{v}$ as a row.

In a model with a cosine similarity distance function, it is detrimental to use Σ^{-1} to weight the context centroid $\hat{\mathbf{c}}$. This is because elements in Σ are the singular values of the co-occurrence matrix along its diagonal, and the amplitude of a singular value (intuitively) corresponds to the significance of a dimension in the latent space; when the inverted matrix is used to weight the centroid, it will "misrepresent" the context by giving more weight to less-significantly co-occurring dimensions and thus sabotage performance. We thus use Σ instead of Σ^{-1} in our experiments. As shown in Figure 1, the best unsupervised performance on the standard FITB dataset is 49.6%, achieved on lexical LoA at $k = 800$.

4.2 Supervised Learning on the Latent Semantic Space Features

In traditional latent space models, the latent space vectors have almost invariantly been used in the unsupervised setting discussed above. Although the number of dimensions has been reduced in the latent semantic space, the inter-relations between the high-dimension data points may still be complex and non-linear; such problems lend themselves naturally to supervised learning.

We therefore formulate the near-synonym lexical choice problem as a supervised classification problem with latent semantic space features. For a test sentence in the FITB task, for example, the context is represented as a latent semantic space vector as discussed in Section 3.1, which is then paired with the correct answer (the near-synonym removed from the sentence) to form one training case.

We choose *Support Vector Machines* (SVMs) as our learning algorithm for their widely acclaimed classification performance on many tasks as well as their noticeably better performance on the lexical choice task in our pilot study. Table 2 lists the supervised model performance on the FITB task together with results reported by other related studies. The model is trained on the 1989 WSJ and tested on the 1987 WSJ to ensure maximal comparability with other results. The optimal k value is 415. Context window size[3] around the gap in a test sentence also affects the model performance. In addition to using the words in the original sentence, we also experiment with enlarging the context window to neighboring sentences and shrinking it to a window frame of n words on each side of the gap. Interestingly, when making the lexical choice, the model tends to favor more-local information — a window frame of size 5 gives the best accuracy of 74.5% on the test. Based on *binomial exact test*[4] with a 95% confidence interval, our result outperforms the current state-of-the-art with statistical significance.

5 Formalizing Subtlety in the Latent Semantic Space

In this section, we formalize the notion of subtlety through its relation to dimensionality, and use the formalization to provide empirical support for some of the common intuitions about subtlety and its complexity with respect to dimensionality and size of context.

5.1 Characterizing Subtlety Using Collocating Differentiator of Subtlety

In language generation, subtlety can be viewed as a subordinate semantic trait in a linguistic realiza-

[3] Note that the *context window* in this paragraph is implemented on FITB test cases, which is different from the context size we compare in Section 5.3 for building co-occurrence matrix.

[4] The binomial nature of the outcome of an FITB test case (right or wrong) makes *binomial exact test* a more suitable significance test than the *t-test* used by Inkpen (2007).

Table 2: Supervised performance on the seven standard near-synonym sets in the FITB task. *95% Confidence* based on *Binomial Exact Test*.

	Co-occur.	SVMs	5-gram	SVMs on
Near-synonyms	network	& PMI	language model	latent vectors
	(Edmonds, 1997)	(Inkpen, 2007)	(Islam and Inkpen, 2010)	(Section 4.2)
difficult, hard, tough	47.9%	57.3%	**63.2**%	61.7%
error, mistake, oversight	48.9%	70.8%	78.7%	**82.5**%
job, task, duty	68.9%	**86.7**%	78.2%	82.4%
responsibility, burden, obligation, commitment	45.3%	66.7%	**72.2**%	63.5%
material, stuff, substance	64.6%	71.0%	70.4%	**78.5**%
give, provide, offer	48.6%	56.1%	55.8%	**75.4**%
settle, resolve	65.9%	75.8%	70.8%	**77.9**%
Average	55.7%	69.2%	69.9%	**74.5**%
Data size	29,835	31,116	31,116	30,300
95% confidence	55.1–56.3%	68.7–69.7%	69.3–70.4%	74.0–75.0%

tion of an intention[5]. A key observation regarding subtlety is that it is non-trivial to *characterize* subtle differences between two linguistic units by their collocating linguistic units. More interestingly, the difficulty in such characterization can be approximated by the difficulty in finding a third linguistic unit satisfying the following constraints:

1. The unit must collocate closely with at least one of the two linguistic units under differentiation;

2. The unit must be characteristic of the difference between the pair.

Such approximation is meaningful in that it transforms the abstract *characterization* into a concrete task of finding this third linguistic unit. For example, suppose we want to find out whether the difference between *glass* and *mug* is subtle. The approximation boils the answer down to the difficulty of finding a third word satisfying the two constraints, and we may immediately conclude that the difference between the pair is *not* subtle since it is relatively easy to find *wine* as the qualifying third word, which 1) collocates closely with *glass* and 2) characterizes the difference between the pair by instantiating one of their major differences — the purpose of use. The same reasoning applies to concluding *non-subtlety* for word pairs such as *pen* and *pencil* with *sharpener*, *weather* and *climate* with *forecast*, *watch* and *clock* with *wrist*, etc.

In contrast, for the pair *forest* and *woods*, it might be easy to find words satisfying one but not both constraints. Consequently, the lack of such qualifying words — or at least the relative difficulty for finding one — makes the difference between this pair more subtle than in the previous examples.

We call a linguistic unit satisfying both constraints a *collocating differentiator of subtlety* (CDS). Notably, the second constraint puts an important difference between CDSs and the conventional sense of collocation. On the lexical level, CDSs are not merely words that collocate more with one word in a pair than with the other; they have to be *characteristic of the differences* between the pair. In the example of *forest* and *woods*, one can easily find a word exclusively collocating with one but not the other — such as *national forest* but not **national woods*; however, unlike the CDSs in the previous examples, the word *national* does not characterize any of the differences between the pair in size, primitiveness,

[5] The same principle applies when we replace "generation" with "understanding" and "an intention" with "a cognition".

proximity to civilization, or wildness (Edmonds and Hirst, 2002), and consequently fails to satisfy the second constraint.

5.2 Relating Subtlety to Latent Space Dimensionality[6]

As mentioned in Section 4.1, elements of a latent space vector are in descending order in terms of co-occurrence significance, i.e., the information within the first few dimensions is obtained from more closely collocating linguistic units. From the two constraints in the previous section, it follows that it should be relatively easier to find a CDS for words that can be well distinguished in a lower-dimensional sub-space of the latent semantic space, and the difference among such words should *not* be considered subtle.

We thus claim that co-occurrence-based information capable of characterizing subtle differences must then reside in higher dimensions in the latent space vectors. Furthermore, our intuition on the complexity of subtlety can also be empirically tested by comparing the performance of supervised and unsupervised models at different k values. One of the differences between the two types of models is that supervised models are better at unraveling the convoluted inter-relations between high-dimensional data points. Under this assumption, if we hypothesize that subtlety is a certain form of complex, high-dimensional relation between semantic elements, then the difference in performance between the supervised and unsupervised model should increase as the former recovers subtle information in higher dimensions.

As shown in Figure 2, performance of both models is positively correlated to the number of dimensions in the latent semantic space (with correlation coefficient $\rho = 0.95$ for supervised model and $\rho = 0.81$ for unsupervised model). This suggests that the lexical choice process is indeed “picking up” implicit information about subtlety in the higher dimensions of the latent vectors. Meanwhile, the difference between the performance of the two models correlates strongly to k with $\rho = 0.95$. Significance tests on the “difference of *difference*”[7] between their performances further reveal increasing difference in growth rate of their performance. Significance is witnessed in both the F-test and the paired t-test,[8] indicating that the subtlety-related information in the higher dimensions exhibits complex clustering patterns that are better recognized by SVMs but beyond the capability of the KNN model.

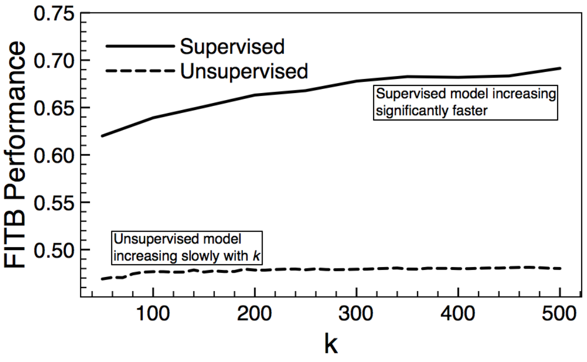


Figure 2: Supervised performance increasing further from unsupervised performance in higher dimensions.

5.3 Subtlety and the Level of Context

Our previous models on lexical LoA associated words within the same sentence to build the co-occurrence matrix. Lexical LoA also allows us to associate words that co-occur in different *levels of context* (LoC) such as paragraphs or documents. This gives an approximate measurement of how much context a lexical LoA model uses for word co-occurrence. Intuitively, by looking at more context, higher LoC models should be better at differentiating more subtle differences.

We compare the performance of models with different LoCs in Figure 3. The sentence LoC model constantly out-performs the paragraph LoC model after $k = 500$, indicating that, by *inter-model comparison*, larger LoC models do not necessarily perform better on higher dimensions. However, there is a noticeable difference in the optimal dimensionality for the model performances. Sentence LoC performance peaks around

[6] In order to keep the test data (1987 *WSJ*) unseen before producing the results in Table 2, models in this section were trained on *The Brown Corpus* and tested on 1988–89 *WSJ*.

[7] The italicized *difference* is used in its mathematical sense as the discrete counterpart of *derivative*.

[8] F-test: $f(1,16) = 9.13, p < 0.01$. Paired t-test: $t(8) = 4.16$ with two-tailed $p = 0.0031$. Both conducted on 10 data points at $k = 50$ to 500 with a step of 50.

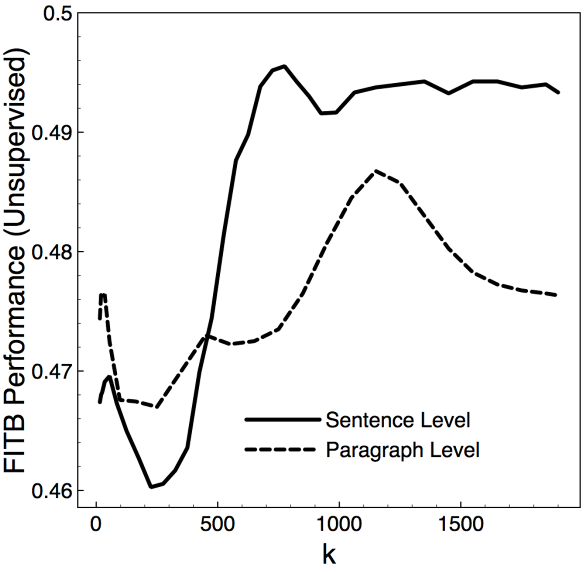


Figure 3: LoC in correlation to latent space dimensionality for optimal model performance.

$k = 700$ — much lower than that of paragraph LoC which is around $k = 1,100$. Such difference may suggest that, by *intra-model comparison*, each model may have its own "comfort zone" for the degree of subtlety it differentiates; models on larger LoC are better at differentiating between more subtle nuances, which is in accordance with our intuition.

One possible explanation for sentence LoC models outperforming paragraph LoC models is that, although the high-dimensional elements are weighed down by Σ due to their insignificance in the latent space, their contribution to the output of distance function is larger in paragraph LoC models because the vectors are much *denser* than that in the sentence LoC model; since the unsupervised method is incapable of recognizing the clustering patterns well in high-dimensional space, the "amplified" subtlety information is eventually taken as noise by the KNN model. An interesting extension to this discussion is to see whether a *supervised* model can consistently perform better on higher LoC in all dimensions.

6 Conclusions and Future Work

We propose a latent semantic space representation of near-synonyms and their contexts, which allows a thorough investigation of several aspects of the near-synonym lexical choice problem. By employing supervised learning on the latent space features, we achieve an accuracy of 74.5% on the "fill-in-the-blank" task, outperforming the current state-of-the-art with statistical significance.

In addition, we formalize the notion of *subtlety* by relating it to the dimensionality of the latent semantic space. Our empirical analysis suggests that subtle differences between near-synonyms reside in higher dimensions in the latent semantic space in complex clustering patterns, and that the degree of subtlety correlates to the level of context for co-occurrence. Both conclusions are consistent with our intuition.

As future work, we will make better use of the easy customization of the context representation to compare HAL and other models with *bag-of-words* models. The correlation between subtlety and dimensionality may lead to many interesting tasks, such as measuring the degree of subtlety for individual near-synonyms or near-synonym sets. With regard to context representation, it is also intriguing to explore other dimensionality reduction methods (such as *Locality Sensitive Hashing* or *Random Indexing*) and to compare them to the SVD-based model.

Acknowledgment

This study is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). We greatly appreciate the valuable suggestions and feedback from all of our anonymous reviewers and from our colleagues Julian Brooke, Frank Rudzicz, and George Dahl.

References

Charles L. A. Clarke, Gordon Cormack, and Christopher Palmer. An overview of MultiText. *ACM SIGIR Forum*, 32(2):14–15, 1998.

D. A. Cruse. *Lexical Semantics*. Cambridge University Press, 1986.

Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

Chrysanne DiMarco, Graeme Hirst, and Manfred Stede. The semantic and stylistic differentiation of synonyms and near-synonyms. *AAAI Spring Symposium on Building Lexicons for Machine Translation*, pages 114–121, 1993.

Philip Edmonds. Choosing the word most typical in context using a lexical co-occurrence network. In *Proceedings of the 35th annual meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 507–509, 1997.

Philip Edmonds and Graeme Hirst. Near-synonymy and lexical choice. *Computational Linguistics*, 28(2):105–144, 2002.

Diana Inkpen. A statistical model for near-synonym choice. *ACM Transactions on Speech and Language Processing*, 4(1):1–17, 2007.

Diana Inkpen and Graeme Hirst. Building and using a lexical knowledge-base of near-synonym differences. *Computational Linguistics*, 32(2): 223–262, 2006.

Aminul Islam and Diana Inkpen. Near-synonym choice using a 5-gram language model. *Research in Computing Sciences*, 46:41–52, 2010.

Thomas Landauer and Susan Dumais. A solution to Plato's problem: the latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997.

Shixiao Ouyang, Helena Hong Gao, and Soo Ngee Koh. Developing a computer-facilitated tool for acquiring near-synonyms in Chinese and English. In *Proceedings of the Eighth International Conference on Computational Semantics*, pages 316–319, 2009.

Reinhard Rapp. The automatic generation of thesauri of related words for English, French, German, and Russian. *International Journal of Speech Technology*, 11(3):147–156, 2008.

Ehud Reiter and Somayajulu Sripada. Human variation and lexical choice. *Computational Linguistics*, 28(4):545–553, 2002.

Manfred Stede. Lexical choice criteria in language generation. In *Proceedings of the sixth conference of the European Chapter of the Association for Computational Linguistics*, pages 454–459, 1993.

Disambiguating Dynamic Sentiment Ambiguous Adjectives

Yunfang Wu
Key Laboratory of Computational Linguistics (Peking University),
Ministry of Education of China
wuyf@pku.edu.cn

Miaomiao Wen*
Department of Electrical Engineering and Information Systems,
University of Tokyo
wenmiaomiao98@gmail.com

Abstract

Dynamic sentiment ambiguous adjectives (DSAAs) like "large, small, high, low" pose a challenging task on sentiment analysis. This paper proposes a knowledge-based method to automatically determine the semantic orientation of DSAAs within context. The task is reduced to sentiment classification of target nouns, which we refer to sentiment expectation instead of semantic orientation widely used in previous researches. We mine the Web using lexico-syntactic patterns to infer sentiment expectation of nouns, and then exploit character-sentiment model to reduce noises caused by the Web data. At sentence level, our method achieves promising result with an f-score of 78.52% that is substantially better than baselines. At document level, our method outperforms previous work in sentiment classification of product reviews.

1 Introduction

In recent years, sentiment analysis has attracted considerable attention in the NLP community. It is the task of mining positive and negative opinions from natural language, which can be applied to many research fields. Previous work on this problem falls into three groups: opinion mining of documents, sentiment classification of sentences and polarity prediction of words. Sentiment analysis both at document and sentence level rely heavily on word level.

The most frequently explored task at the word level is to determine the polarity of words, in which most work centers on assigning a prior polarity to words or word senses in the lexicon out of context. However, for some words, the polarity varies strongly with context, making it hard to attach each to a fixed sentiment category in the lexicon. For example, the word "low" has a positive orientation in "low cost" but a negative orientation in "low salary". We call these words like "low" dynamic sentiment ambiguous adjectives (DSAAs). Turney and Littman (2003) claim that DSAAs cannot be avoided in a real-world application. But unfortunately, DSAAs are discarded by most research concerning sentiment analysis.

In this paper, we are devoted to the challenging task of disambiguating DSAAs. The task is to automatically determine the semantic orientation (SO) of DSAAs within context. We limit our work to 14 frequently used adjectives in Chinese, such as "large, small, many, few, high, low", which all have the meaning of measurement. Although the number of such ambiguous adjectives is not large, they are frequently used in real text, especially in the texts expressing opinions and emotions. As demonstrated by the experimental results in this paper, the disambiguation of 14 DSAAs can obviously improve the performance of sentiment classification of product reviews.

The task of disambiguating DSAAs is reduced to sentiment classification of nouns. Previous studies classify nouns into three categories: positive, negative and neutral. In contrast, we propose two categories of sentiment expectation

*Most of the work was performed when the author was a student at Peking University.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1191–1199,
Beijing, August 2010

of nouns: positive expectation and negative expectation. This paper presents a novel approach to automatically predict sentiment expectation of nouns. First, we infer the sentiment expectation of a noun by mining the Web with strongly-polar-steering lexico-syntactic patterns. Secondly, we derive the sentiment expectation of a noun from its component characters, which capture the semantic relationship between Chinese words and characters. Finally, a better performance is obtained by combing the two methods. We conduct two types of experiments: the experimental results at the sentence level validate the effectiveness of our approach; the experimental results at the document level confirm the significance of the problem we addressed.

2 Related Work

2.1 Word-level Sentiment Analysis

Recently there has been extensive research in sentiment analysis, for which Pang and Lee (2008) give an in-depth survey of literature. Closer to our study is the large body of work on automatic SO prediction of words (Hatzivassiloglou and McKeown, 1997; Turney and Littman, 2003; Kim and Hovy, 2004; Andreevskaia and Bergler, 2006), but unfortunately they all discard DSAAs in their research. In recent years, some studies go a step further, attaching SO to senses instead of word forms (Esuli and Sebastiani, 2006; Wiebe and Mihalcea, 2006; Su and Markert 2008), but their work is still limited in lexicon out of context.

The most relevant work is Ding et al. (2008), in which DSAAs are named as context dependant opinions. They argue that there is no way to know the SO of DSAAs without prior knowledge, and asking a domain expert to provider such knowledge is scalable. They adopt a holistic lexicon-based approach to solve this problem, which exploits external information and evidences in other sentences and other reviews. On the contrary in this paper, we obtain the prior knowledge of a product by mining the web, and then use such knowledge to determine the SO of DSAAs. The prior knowledge of a product, which is closer to the sentiment expectation of nouns described in this paper, is an important research issue in itself and has many applications in sentiment analysis, as discussed in section 3.2.

2.2 Phrase-level Sentiment Analysis

The disambiguation of DSAAs can also be considered as a problem of phrase-level sentiment analysis. Wilson et al. (2004) present a two-step process to recognize contextual polarity that employs machine learning and a variety of features. Takamura et al. (2006, 2007) propose latent variable model and lexical network to determine SO of phrases, focusing on “noun+adjective” pairs. Their experimental results suggest that the classification of pairs containing ambiguous adjectives is much harder than those with unambiguous adjectives. The above mentioned approaches are all supervised, and need human labeled data for training. In contrast, our method is unsupervised and can overcome the data acquisition bottleneck. Moreover, we focus on the much harder task of disambiguating DSAAs in “noun+adjective” pairs.

2.3 Pattern-based Method

Previous studies have applied pattern-based method to sentiment analysis (Riloff and Wiebe, 2003; Wiebe et al., 2004; Riloff et al., 2005; Wiebe and Mihalcea, 2006; Andreevskaia and Berger; 2006). The differences with our method lie in two aspects: the used resources (corpus versus web) and the research target (subjective expressions versus sentiment expectation).

2.4 Character-based Method

Chinese characters carry semantic information that is indicative of semantic properties of words. Previous studies have exploited the character-based model to predict the semantic categories of Chinese unknown words (Chen, 2004; Lu, 2007). Yuen et al. (2004) presents a method to infer the SO of a Chinese word from its statistical association with strong-polarized characters rather than with strong-polarized words. The work by Ku et al. (2006) is similar to ours because they also define the sentiment score of a word by its composite characters. However, their algorithm is based only on frequency, while we exploit point mutual information that can capture the character-sentiment association.

3 Determining SO of Adjective by Target Noun

3.1 Classification of DSAAs

The frequently used DSAAs are given below. We group them into two categories: positive-like adjectives and negative-like adjectives. These adjectives are neutral out of context, but positive or negative emotion will be evoked when they co-occur with some target nouns, making it hard to assign each to a fixed sentiment category in lexicon.

(1) Positive-like adjectives (*Pa*) = {大 da|large, 多 duo|many, 高 gao|high, 厚 hou|thick, 深 shen|deep, 重 zhong|heavy, 巨大 ju-da|huge, 重大 zhong-da|great}

(2) Negative-like adjectives (*Na*) ={小 xiao|small, 少 shao |few, 低 di|low, 薄 bao|thin, 浅 qian|shallow, 轻 qing|light}

3.2 Sentiment Expectation of Noun

The SO of most DSAAs can be determined by target nouns in noun-adjective phrases, as shown in Table 1. For example, the word "high" has a positive orientation when the target noun is "salary" but a negative orientation when the target noun is "price". Therefore, the task can be reduced to sentiment classification of nouns.

Positive 潜力大|potential is great 工资高|salary is high	Negative 潜力小|potential is small 工资低|salary is low
Negative 压力大|pressure is big 价格高|price is high	Positive 压力小|pressure is small 价格低|price is low

Table 1: The SO of DSAAs in noun-adjective phrases

In previous research, the SO of nouns is classified into three categories: positive, negative and neutral. Accordingly, "压力 ya-li|pressure" will be assigned as negative and "潜力 qian-li|potential" as positive, while "工资 gong-zi|salary" and "价格 jia-ge|price" will be assigned as neutral, as the two terms are objective and cannot evoke positive or negative emotion. Different from the traditional classification scheme, we propose sentiment expectation and classify nouns into two categories: positive expectation and negative expectation. For a positive expectation noun, people usually expect the thing referred to by the noun to be bigger, higher or happen frequently. On the contrary, for a negative expectation noun, people usually expect the thing referred to by the noun to be smaller, lower or don't happen . For example, "价格 jia-ge|price" is a negative expectation noun, as most people in most cases expect that the product prices become low, whereas "工资 gong-zi|salary" is a positive expectation noun, as most people in most cases expect that their salaries become high. The relationship between traditional SO and sentiment expectation can be defined as: positive (negative) terms correspond to positive (negative) expectation terms, but some neutral terms may also carry positive (negative) expectation.

Su and Markert (2008) argue that polarity can also be attached to objective words. The difference with our scheme is that, for example, "价格 jia-ge|price" is attached to negative expectation in our scheme while is still neutral in Su and Markert's method.

The distinction between positive and negative expectation nouns is vital to determine the SO of some phrases. Using it to disambiguate DSAAs is a good example. Another application is the phrase containing verbs with the meaning of status change. For example, "工资上涨了|salary has been raised" will evoke positive emotion, while "价格上涨了 jiage-shangzhang-le|prices have gone up" will evoke negative emotion. As far as we are aware, this is the first sentiment analysis scheme that tries to exploit people's expectation towards nouns.

3.3 Determination of DSAAs

The SO of DSAAs in a given phrase can be calculated by Eq. (1).

$$C(a) = \begin{cases} 1 & \text{if a is positive-like} \\ -1 & \text{if a is negative-like} \end{cases} \qquad (1)$$

$$C(n) = \begin{cases} 1 & \text{if n is positive expectation} \\ -1 & \text{if n is negative expectation} \end{cases}$$

$$SO(a)=C(a)*C(n)$$

If adverb="不 bu|not", $SO(a)= -SO(a)$

Where C(a) denotes the category of DSAAs; C(n) denotes the sentiment expectation of nouns; SO(a) is the SO of DSAAs in a give noun-adjective phrase. When the adverb is the negation term "不 bu|not", the SO is reversed.

4 Predicting Sentiment Expectation of Noun

4.1 Pattern-based Prediction Using a Web Search Engine

In natural language, there are some lexico-syntactic patterns that people frequently use when they express their opinion about something. For example:

(3) 工资*有点*低 | Salary is *a little* low.

(4) 价格*有点*高| Price is *a little* high.

The pattern "<n> 有点 <a>" carries a strong negative association in Chinese language. When a man is saying "工资有点低| Salary is a little low", it indicates that he wishes his "工资 |salary" to be raised. On the contrary, when a man is saying "价格有点高| price is a little high", it indicates that he wishes "价格 |price" to go down. As a result, "工资 |salary" has positive expectation while "价格 |price" has negative expectation.

With the rapid development and expansion of the internet, Web has become an important medium for people to post their ideas. The opinions expressed on the Web reflect the common cognition shared by collection of people in a culture. Therefore, using a Web search engine with the strong-polar-steering lexico-syntactic patterns as queries, we can infer the sentiment expectation of a noun, by calculating its statistical association with positive and negative hits.

As an example, using the search engine Baidu[2] with the pattern "<n> 有点 <a>" as queries, we obtain the following hits:

(5) 工资*有点*低 | Salary is a little low. (2890 hits)
工资*有点*高 | Salary is a little high (67 hits)

(6) 价格*有点*高 | Price is a little high. (19400 hits)
价格*有点*低 | Price is a little low. (1080 hits)

The more than 40 times more numerous hits for "工资有点低 |Salary is a little low" indicate that that "工资|salary" is a positive expectation noun. For the same reason, we can infer that "价格 |price" has negative expectation.

DSAAs are classified into two opposite sets *Pa* and *Na*, as listed in (1) and (2) respectively. Here two-character adjectives ("巨大 |huge" and "重大 |great") are discarded. Four types of lexico-syntactic patterns, which are also classified into two opposite sets in consistent with *Pa* and *Na*, are used in this paper, as listed in Table 2. These patterns were manually designed, inspired by linguistic knowledge and after a deep investigation on the Web.

Pos. expectation patterns	Neg. expectation patterns
1) <n> 有点 *Na* n is a little *Na* 2) <n> 有点儿 *Na* n is a little *Na* 3) <n> *Na*, 怎么办 n is *Na*, what should we do? 4) 嫌 <n> *Na* n is too *Na*	1) <n> 有点 *Pa* n is a little *Pa* 2) <n> 有点儿 *Pa* n is a little *Pa* 3) <n> *Pa*, 怎么办 n is *Pa*, what should we do? 4) 嫌 <n> *Pa* n is too *Pa*

Table 2: The lexico-syntactic patterns

Here the noun (n) in these patterns was instantiated by 9,468 nouns in our collected data. A noun has together 48 patterns, 24 positive and 24 negative ones. For each noun, we obtain the hits of both positive and negative expectation patterns, using the search engine Baidu. The sentiment expectation of a noun is acquired by Eq. (2) and Eq. (3), where the magnitude of $PT\_SO(n)$ can be considered as the strength of sentiment expectation.

$$\begin{aligned} PT\_SO(n) = & \sum_{b\in Na}\sum_{i=1}^{4} PositivePatternHit_i(n,b) \\ & - \sum_{a\in Pa}\sum_{i=1}^{4} NegativePatternHit_i(n,a) \end{aligned} \tag{2}$$

$$\text{n is } \begin{cases} \text{positive expectation} & \text{if } PT\_SO(n)>0 \\ \text{negative expectation} & \text{if } PT\_SO(n)<0 \\ \text{not predicted} & \text{if } PT\_SO(n)=0 \end{cases} \tag{3}$$

Table 3 gives some nouns with sentiment expectation predicted by the pattern-based method, descending (the left column) and ascending (the right column) by the absolute value of $PT\_SO(n)$. Most words (9 out of 10) are correctly predicted, demonstrating that the result of pattern-based method is promising. The only wrong predicted noun is "感觉 |feeling", due to the fact that most instances of it on the Web data are used as verb rather than noun, like "感觉有点大| I think it is large".

[2] http://baidu.com.cn.

Positive expectation	Negative expectation
Noun ($PT\_SO(n)$)	Noun ($PT\_SO(n)$)
钱|money (31349)	温度|temperature(-111576)
工资|wage (26311)	噪音|noise (-45790)
感觉|feeling (20102)	价格|price (-25653)
收入|income(19429)	代价|cost (-22051)
官|officer (10630)	血压|blood pressure (-21788)

Table 3: Examples of nouns with sentiment expectation predicted by the pattern-based method

4.2 Character-based Derivation Using Sentiment Lexicons

But the sentiment expectation of some nouns cannot be predicted with the pattern-based method, mainly due to the reason that these nouns don't occur in the listed patterns in Table 2. An alternate way is to exploit the semantic knowledge of Chinese characters. It is assumed that there is a strong association between the sentiment category of a word and its component characters. For example, the three words "罪恶 zui'e|evil, 罪行 zuixing|crime, 罪过 zuiguo|fault", which all contain the character "罪 zui|sin" that carries negative meaning, are all negative expectation nouns.

First, we compute the character-word sentiment association by the following PMI formula, based on a sentiment lexicon:

$$PMI(c, Positive) = \log \frac{P(c, Positive)}{P(c)P(Positive)}$$

$$PMI(c, Negative) = \log \frac{P(c, Negative)}{P(c)P(Negative)} \quad (4)$$

$$SO(c) = PMI(c, Positive) - PMI(c, Negative)$$

Where $P(c, Positive)$ is the probability of a character c in the positive category; $P(c)$ is the probability of a character c in the sentiment lexicon; $P(Positive)$ is the probability of the positive category in the sentiment lexicon. $PMI(c, Negative)$ has the similar meaning. Probabilities are estimated according to the maximum likelihood principle.

The open language resources for Chinese sentiment analysis are quite limited. We selected the following two sentiment lexicons.

Sentiment HowNet. HowNet has published the Chinese vocabulary for sentiment analysis[3], which was manually constructed. The positive category contains 4,566 words and the negative category contains 4,370 words.

Sentiment BaiduHit. In our collected data, we extracted 9,468 nouns. Using the pattern-based method we acquired sentiment expectation of these nouns, where 2,530 ones were assigned as positive expectation, 1,837 ones as negative expectation and 5,101 ones were not predicted. It is assumed that most nouns are correctly predicted. These nouns with their sentiment expectation constitute the lexicon of Sentiment BaiduHit, which is automatically constructed.

Combining HowNet and BaiduHit. Most sentiment characters derived from HowNet have adjective property, since most words in Sentiment HowNet are adjectives. On the contrary, most sentiment characters derived from BaiduHit have noun property. Therefore, the combination of the two lexicons can cover more characters. As Sentiment HowNet is manually compiled, the sentiment characters derived from it should be more reasonable than those from BaiduHit. When combining the two lexicons in computing character polarity, we assign a high priority to HowNet. Only when a character is out of vocabulary in HowNet, we resort to BaiduHit.

Then, we acquire the sentiment category of a word by computing the following equation. Let a word consist of n characters $w = c_1, c_2, \ ... c_n$, the sentiment category of the word is calculated by the average sentiment value of its component characters:

$$CH\_SO(w) = \frac{1}{n}\sum_{i=1}^{n} SO(c_i) \quad (5)$$

$$\text{w is} \begin{cases} \text{positive expectation if } CH\_SO(w)>0 \\ \text{negative expectation if } CH\_SO(w)<0 \\ \text{neutral} \qquad\qquad\quad \text{if } CH\_SO(w)=0 \end{cases} \quad (6)$$

We acquired *sentiment expectation* of 9,468 nouns in our collected data, based on Sentiment HowNet, Sentiment BaiduHit, and the combination of the two lexicons, respectively.

Table 6 gives examples of nouns with sentiment expectation acquired by the character-based method combining the two lexicons of HowNet and BaiduHit, descending (the left column) and ascending (the right column) by the absolute value of $CH\_SO(w)$.

[3] http://www.keenage.com/html/c_index.html.

Positive expectation	Negative expectation
Noun($CH\_SO(w)$)	Noun($CH\_SO(w)$)
美称|good name (3.23)	灰|ash (-3.22)
健美|health (3.06)	毛|gross (-2.93)
香|fragrance (3.05)	税|tax (-2.89)
美方|U.S.A (2.98)	毛病|fault (-2.84)
职称|title (2.64)	毒|poison (-2.82)

Table 4: Example of nouns with sentiment expectation predicted by the character-based method

4.3 Integrating Pattern-based Prediction and Character-based Derivation

The two methods of pattern-based prediction and character-based derivation have complementary properties. The pattern-based method concentrates on a word's usage on the Web, whereas the character-based method focuses on the internal structure of a word. So the two methods can be integrated to get better performance. The results using pattern-based method are much better than character-based method, as illustrated in Table 3 and Table 4. So in the integrated scheme, we give a high priority to pattern-based method. The pattern-based approach is mainly used, and only when the value of $|PT\_SO(n)|$ is smaller than a threshold r, the character-based method is adopted. Because when the value of $|PT\_SO(n)|$ is very small, it could be caused by random noises on the Web. We set r to 9 according to empirical analysis in the development data.

5 Experiments

5.1 Sentiment Analysis at Sentence Level

5.1.1 Data

We collected data from two sources. The main part was extracted from Xinhua News Agency of Chinese Gigaword (Second Edition) released by LDC. The texts were automatically word-segmented and POS-tagged using the open software ICTCLAS[4]. In order to concentrate on the disambiguation of DSAAs, and reduce the noise introduced by the parser, we extracted sentences containing strings in pattern of (7), where the target noun is modified by the adjective in most cases.

(7) noun+adverb+adjective (adjective $\in$ DSAAs)
e.g. 成本/n 较/d 低/a | the cost is low.

Another small part of data was extracted from the Web. Using the search engine Google[5], we searched the queries as in (8):

(8) 很| very+ adjective (adjective $\in$ DSAAs)
From the returned snippets, we manually picked out some sentences that contain the strings of (7). Also, the sentences were automatically word-segmented and POS-tagged using ICTCLAS.

DSAAs in the data were assigned as positive, negative or neutral, independently by two annotators. Since we focus on the distinction between positive and negative categories, the neutral instances were removed. Table 5 gives statistics of the data, and the inter-annotator agreement is in a high level with a kappa of 0.91. After cases with disagreement were negotiated between the two annotators, a gold standard annotation was agreed upon. In this paper, 3066 instances were divided randomly into three parts, 1/3 of which were used as the development data, and 2/3 were the test data.

Most of the data has been used as the benchmark dataset of SemEval-2010 task 18 "disambiguating sentiment ambiguous adjectives" (Wu and Jin, 2010), and so it can be downloaded freely for research.

	Pos#	Neg#	Total#
Pos#	1280	58	1338
Neg#	72	1666	1738
Total#	1352	1724	3066

Table 5: The statistics of DSAAs data

5.1.2 Baseline

We conducted two types of baseline.

Simple Baseline. Not considering the context, assign all positive-like adjectives as positive, and all negative-like adjectives as negative.

HowNet Baseline. Acquiring SO of nouns from Sentiment HowNet, the polarity of DSAAs is computed by Eq. (1).

5.1.3 Methods

Pattern-based method. Acquiring sentiment expectation of nouns using the pattern-based method, the polarity of DSAAs is computed by Eq.(1).

[4] http://www.ictclas.org/.

[5] http://www.google.com/.

Character-based method. Acquiring sentiment expectation of nouns using the character-based method, based on Sentiment HowNet, Sentiment BaiduHit and the combination of the two lexicons respectively, the polarity of DSAAs is computed by Eq.(1).

Integrated method. Acquiring sentiment expectation of nouns by integrating pattern-based and character-based methods, the polarity of DSAAs is computed by Eq. (1).

5.1.4 Results

Table 6 gives the experimental results at sentence level with different methods.

Methods	Pre.	Rec.	F
Simple Baseline	61.20	61.20	61.20
HowNet Baseline	97.58	9.88	17.94
Pattern-based	75.83	71.67	73.69
Character-based (HowNet)	69.89	69.37	69.63
Character-based (BaiduHit)	68.66	68.59	68.62
Character-based (Combined)	71.01	70.94	70.97
Integrated method	**78.52**	**78.52**	**78.52**

Table 6: The experimental results at sentence level

As for the simple baseline, both the precision and recall are low, suggesting that DSAAs cannot be neglected for sentiment analysis in a real-world application.

The HowNet baseline achieves a quite high precision of 97.58%, but a rather poor recall of 9.88%, suggesting that SO of nouns described in traditional sentiment lexicon, like HowNet, cannot effectively disambiguate DSAAs.

The proposed methods in this paper all yield results that are substantially better than two types of baseline. The pattern-based method, as straightforward as it is, achieves promising result with an f-score of 73.69%, which is 12.49% higher than the simple baseline. The pattern-based method outperforms the character-based method (combined) by 4.82% in precision and 0.73% in recall. The performance of the character-based method based on Sentiment BaiduHit is competitive with that based on Sentiment HowNet, which again proves the effectiveness of the pattern-based method. The character-based method combining the two lexicons outperforms each lexicon with small improvement. The approach integrating pattern-based and character-based methods outperforms each method in isolation, achieving an f-score of 78.52% that is 17.32% higher than the simple baseline and 60.58% higher than HowNet baseline.

5.2 Sentiment Analysis at Document Level

5.2.1 Data

We also investigated the impact of disambiguating DSAAs on the sentiment classification of product reviews. Following the work of Wan (2008), we selected the same dataset. The dataset contains 886 Chinese product reviews, which are manually annotated with polarity labels: positive or negative. Also, the files are automatically word-segmented and POS-tagged using ICTCLAS. We extracted the files that contain the following strings, where the nouns are modified by DSAAs in most cases.

(9) noun+adjective (adjective $\in$ DSAAs)
noun+adverb+adjective
noun+adverb+adverb+adjective.

We obtained 212 files, up to 24% of the overall data, suggesting again that DSAAs are frequently used in product reviews and cannot be avoided in a real-world application.

5.2.2 Methods

Our goal is not to propose a new method, but instead to test the performance gain by adding the disambiguation of DSAAs. We adopted the same algorithm with Wan (2008), and also used Sentiment-HowNet. But in our experiment, *Negation_Dic* contains only one term "不 bu|not", for the sake of repeatable experiments.

The baseline algorithm is illustrated by the non-italic part in Figure 1, where we set the same parameters with Wan's approach: *PosValue*=1, *NegValue*=-2, *q*=2, *ρ*=2.

We added the disambiguation of DSAAs to the algorithm, as illustrated by the italic part in Figure 1. When a word is a DSAA, compute its SO with the proposed integrated method, rather than using its prior polarity specified in HowNet. For *Dy_PosValue* and *Dy_NegValue*, we first set *Dy_PosValue*=1 and *Dy_NegValue*=-2, just the same as *PosValue* and *NegValue.* In the second attempt, in order to further intensify the polarity of DSAAs, we set *Dy_PosValue*=1.5 and *Dy_NegValue*=-2.5. Other parameters were set the same as baseline.

Algorithm Compute_SO:

```
1. Tokenize document d into sentence set S, and each sentence s∈S is tokenized into word set Ws;
2. For any word w in a sentence s∈S, compute its value SO(w) as follows:
    1) if w ∈DSAAs, compute SO(w) with the integrated method.
        If SO(w)=1, SO(w)=Dy_PosValue;
        If SO(w)=-1, SO(w)=Dy_NegValue;
    2) if w∈Positive_Dict, SO(w)=PosValue;
    3) If w∈Negative_Dict, SO(w)=NegValue;
    4) Otherwise, SO(w)=0;
    5) Within the window of q words previous to w, if there is a term w'∈Negation_Dict,
        SO(w)= –SO(w);
    6) Within the window of q words previous to w, if there is a term w' ∈Intensifier_Dict,
        SO(w) =ρ×SO(w);
3. $S(d) = \sum_{s\in S} \sum_{w\in Ws} SO(w)$
```

Figure 1: Algorithm of computing SO of documents

5.2.3 Results

Adding the disambiguation of *DSAAs*, the performance of sentiment classification of 212 product reviews was significantly improved, as shown in Table 7.

		Baseline	DSAAs (1, -2)	DSAAs (1.5,-2.5)
Pos.	Pre.	75.89	77.50	76.61
	Rec.	78.70	86.11	87.96
	F	77.27	81.58	81.90
Neg.	Pre.	87.01	88.46	87.06
	Rec.	64.42	66.35	71.15
	F	74.03	75.82	78.31
Total	MacroF	75.62	78.60	**80.06**
	Accu.	71.70	76.42	**79.72**

Table 7: The experimental results at document level

As an example, the following review, which consists of only one sentence, is correctly classified as positive by DSAAs method, but is classified as negative by the baseline approach.

(10) 体积小，重量轻，携带很方便。
| Small size, light weight, and easy to carry.

According to HowNet, as shown in Table 8, the sentence contains two negative words "小|small" and "轻|light" and one positive word "方便 fangbian|easy", resulting the overall negative prediction. In our approach, "体积 tiji|size" and "重量 zhongliang|weight" are assigned as negative expectation, and consequently both "体积小|small size" and "重量轻|light weight" have positive meaning, resulting the overall positive prediction.

Pos.	大 |large, 高 |high, 厚 |thick, 深 |deep, 重|heavy, 重大|great
Neg.	小 |small, 低 |low, 薄 |thin, 浅 |shallow, 轻|light
OOV	多|many, 少|few, 巨大|huge

Table 8: The SO of DSAAs described in HowNet

Adding the disambiguation of DSAAs, our method obviously outperforms the baseline by 4.44% in f-score and 8.02% in accuracy. The improvement in recall is especially obvious. When intensifying the polarity of DSAAs by setting *Dy_PosValue*=1.5 and *Dy_NegValue*=-2.5, the recall is improved by 9.26% for positive category and 6.73% for negative category.

6 Conclusion and Future Work

This paper presents a knowledge-based unsupervised method to automatically disambiguate dynamic sentiment ambiguous words, focusing on 14 DSAAs. We exploit pattern-based and character-based methods to infer sentiment expectation of nouns, and then determine the polarity of DSAAs based on the nouns. For the sentiment analysis at sentence level, our method achieves promising result that is significantly better than two types of baseline, which validates the effectiveness of our approach. We also apply the disambiguation of 14 DSAAs to the sentiment classification of product reviews, resulting obvious improvement in performance, which proves the significance of the issue.

There leaves room for improvement. Our future work will explore more contextual information in disambiguating DSAAs. In addition, we will find out new methods to reduce noises when mining the Web to infer sentiment expectation of nouns. Discovering the lexico-syntactic patterns for sentiment expectation of nouns automatically or semi-automatically with bootstrapping method is also a challenging direction.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 60703063) and National Social Science Foundation of China (No. 08CYY016).

References

Andreevskaia A. and Bergler S. 2006. Sentiment tagging of adjectives at the meaning level. *The 19th Canadian Conference on Artificial Intelligence.*

Andreevskaia, A. and Bergler, S. 2006. Mining WordNet for fuzzy sentiment: Sentiment tag extraction from WordNet glosses. *Proceedings of EACL 2006.*

Chen, C-J. 2004. Character-sense association and compounding template similarity: automatic semantic classification of Chinese compounds. *Proceedings of the 3rd workshop on Chinese language processing.*

Ding X., Liu B. and Yu, P. 2008. A holistic lexicon-based approach to opinion mining. *Proceedings of WSDM'08.*

Esuli, A. and Sebastiani, F. 2006. SentiWordNet: a publicly available lexical resource for opinion mining. *Proceedings of LREC'06.*

Hatzivassiloglou, V. and McKeown, K. 1997 Predicting the semantic orientation of adjectives. *Proceedings of ACL'97.*

Kim, S and Hovy, E. 2004. Determining the sentiment of opinions. *Proceedings of COLING'04.*

Ku, L, Liang Y. and Chen, H. 2006. Opinion extraction, summarization and tracking in news and blog corpora. *Proceedings of AAAI-2006 Spring Symposium on Computational Approaches to Analyzing Weblogs.*

Lu X-F, 2007. Hybrid models for semantic classification of Chinese unknown words. *Proceedings of NAACL HLT'07..*

Pang, B. and Lee, L. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval.*

Riloff, E. and Wiebe, J. 2003. Learning Extraction Patterns for Subjective Expressions. *Proceedings of EMNLP'03.*

Riloff, E., Wiebe, J. and Phillips, W. 2005. Exploiting Subjectivity Classification to Improve Information Extraction. *Proceedings of AAAI'05.*

Su, F. and Markert, K. 2008. From words to senses: a case study of subjectivity recognition. *Proceedings of COLING'08.*

Takamura, H., Inui,T. and Okumura, M. 2006. Latent Variable Models for Semantic Orientations of phrases. *Proceedings of EACL'06.*

Takamura, H., Inui,T. and Okumura, M. 2007. Extracting Semantic Orientations of Phrases from Dictionary. *Proceedings of NAACL HLT '07.*

Turney, P. and Littman, M. 2003. Measuring praise and criticism: inference of semantic orientation from association. *ACM transaction on information systems.*

Wan, X. 2008. Using Bilingual Knowledge and Ensemble Techniques for Unsupervised Chinese Sentiment Analysis. *Proceedings of EMNLP'08.*

Wiebe, J. and Mihalcea, R. 2006. Word sense and subjectivity. *Proceedings of ACL'06.*

Wiebe, J., Wilson, T., Bruce, R., Bell, M. and Martin, M. 2004. Learning Subjective Language. *Computational Linguistics.*

Wilson, T., Wiebe, J. and Hoffmann, P. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. *Proceedings of HLT/EMNLP'05.*

Wu, Y. and Jin, P. 2010. SemEval-2010 task 18: disambiguating sentiment ambiguous adjectives. *Proceedings of SemEval 2010.*

Yuen R., Chan T., Lai T., Kwong O., T'sou B. 2004. Morpheme-based derivation of bipolar semantic orientation of Chinese words. *Proceedings of COLING'04.*

Joint Tokenization and Translation

Xinyan Xiao † **Yang Liu** † **Young-Sook Hwang** ‡ **Qun Liu** † **Shouxun Lin** †

†Key Lab. of Intelligent Info. Processing
Institute of Computing Technology
Chinese Academy of Sciences
{xiaoxinyan,yliu,liuqun,sxlin}@ict.ac.cn

‡HILab Convergence Technology Center
C&I Business
SKTelecom
yshwang@sktelecom.com

Abstract

As tokenization is usually ambiguous for many natural languages such as Chinese and Korean, tokenization errors might potentially introduce translation mistakes for translation systems that rely on 1-best tokenizations. While using lattices to offer more alternatives to translation systems have elegantly alleviated this problem, we take a further step to tokenize and translate jointly. Taking a sequence of atomic units that can be combined to form words in different ways as input, our joint decoder produces a tokenization on the source side and a translation on the target side simultaneously. By integrating tokenization and translation features in a discriminative framework, our joint decoder outperforms the baseline translation systems using 1-best tokenizations and lattices significantly on both Chinese-English and Korean-Chinese tasks. Interestingly, as a tokenizer, our joint decoder achieves significant improvements over monolingual Chinese tokenizers.

1 Introduction

Tokenization plays an important role in statistical machine translation (SMT) because tokenizing a source-language sentence is always the first step in SMT systems. Based on the type of input, Mi and Huang (2008) distinguish between two categories of SMT systems : *string-based* systems (Koehn et al., 2003; Chiang, 2007; Galley et al., 2006; Shen et al., 2008) that take a string as input and *tree-based* systems (Liu et al., 2006; Mi et al., 2008) that take a tree as input. Note that a tree-based system still needs to first tokenize the input sentence and then obtain a parse tree or forest of the sentence. As shown in Figure 1(a), we refer to this pipeline as **separate** tokenization and translation because they are divided into single steps.

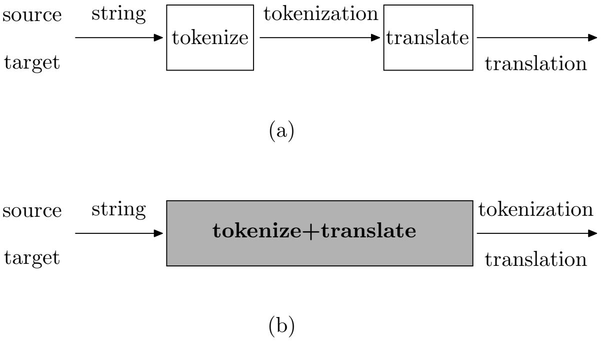


Figure 1: (a) Separate tokenization and translation and (b) joint tokenization and translation.

As tokenization for many languages is usually ambiguous, SMT systems that separate tokenization and translation suffer from a major drawback: tokenization errors potentially introduce translation mistakes. As some languages such as Chinese have no spaces in their writing systems, how to segment sentences into appropriate words has a direct impact on translation performance (Xu et al., 2005; Chang et al., 2008; Zhang et al., 2008). In addition, although agglutinative languages such as Korean incorporate spaces between "words", which consist of multiple morphemes, the granularity is too coarse and makes the training data considerably sparse. Studies reveal that segmenting “words” into morphemes effectively improves translating morphologically rich languages (Oflazer, 2008). More importantly, a tokenization close to a gold standard does not necessarily leads to better translation quality (Chang et al., 2008; Zhang et al., 2008). Therefore, it is necessary to offer more tokenizations to SMT systems to alleviate the tokenization error propagation problem. Recently, many researchers have shown that replacing 1-best tokenizations with lattices improves translation performance significantly (Xu et al., 2005; Dyer et al., 2008; Dyer, 2009).

We take a next step towards the direction of offering more tokenizations to SMT systems by proposing **joint** tokenization and translation. As shown in Figure 1(b), our approach tokenizes and translates jointly to find a tokenization and a translation for a source-language string simultaneously. We integrate translation and tokenization models into a discriminative framework (Och and Ney, 2002), within which tokenization and translation models interact with each other. Experiments show that joint tokenization and translation outperforms its separate counterparts (1-best tokenizations and lattices) significantly on the NIST 2004 and 2005 Chinese-English test sets. Our joint decoder also reports positive results on Korean-Chinese translation. As a tokenizer, our joint decoder achieves significantly better tokenization accuracy than three monolingual Chinese tokenizers.

2 Separate Tokenization and Translation

Tokenization is to split a string of characters into meaningful elements, which are often referred to as words. Typically, machine translation separates tokenization from decoding as a preprocessing step. An input string is first preprocessed by a tokenizer, and then is translated based on the tokenized result. Take the SCFG-based model (Chiang, 2007) as an example. Given the character sequence of Figure 2(a), a tokenizer first splits it into the word sequence as shown in Figure 2(b), then the decoder translates the word sequence using the rules in Table 1.

This approach makes the translation process simple and efficient. However, it may not be optimal for machine translation. Firstly, optimal granularity is unclear for machine translation. We might face severe data sparseness problem by using large granularity, while losing much useful information with small one. Consider the example in Figure 2. It is reasonable to split *duo fen* into two words as *duo* and *fen*, since they have one-to-one alignments to the target side. Nevertheless, while *you* and *wang* also have one-to-one alignments, it is risky to segment them into two words. Because the decoder is prone to translate *wang* as a verb *look* without the context *you*. Secondly, there may be tokenization errors. In Figure2(c), *tao fei ke* is recognized as a Chinese person name with the second name *tao* and the first name *fei-ke*, but the whole string *tao fei ke* should be a name of the Indonesian badminton player.

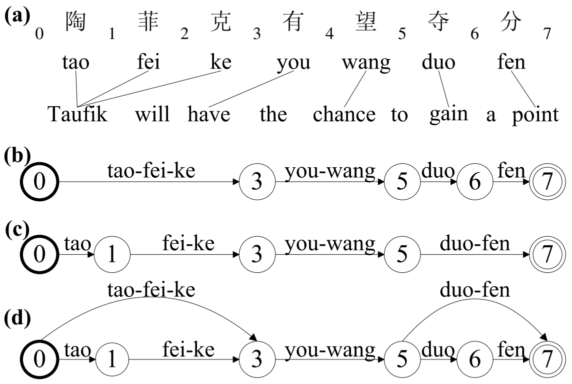

(a)
0 陶 1 菲 2 克 3 有 4 望 5 夺 6 分 7
tao fei ke you wang duo fen
Taufik will have the chance to gain a point
(b)
0 tao-fei-ke 3 you-wang 5 duo 6 fen 7
(c)
0 tao 1 fei-ke 3 you-wang 5 duo-fen 7
(d)
tao-fei-ke
duo-fen
0 tao 1 fei-ke 3 you-wang 5 duo 6 fen 7

Figure 2: Chinese tokenization: (a) character sequence; (b) and (c) tokenization instances; (d) lattice created from (b) and (c). We insert “-” between characters in a word just for clarity.

r_1	*tao-fei-ke* →*Taufik*
r_2	*duo fen* → *gain a point*
r_3	x_1 *you-wang* x_2 → x_1 *will have the chance to* x_2

Table 1: An SCFG derivation given the tokenization of Figure 2(b).

Therefore, it is necessary to offer more tokenizations to SMT systems to alleviate the tokenization error propagation problem. Recently, many researchers have shown that replacing 1-best tokenizations with lattices improves translation performance significantly. In this approach, a lattice compactly encodes many tokenizations and is fixed before decoding.

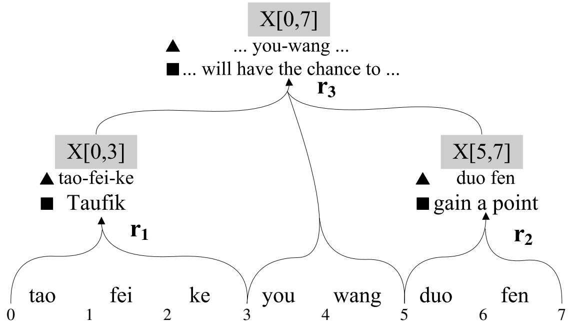


Figure 3: A derivation of the joint model for the tokenization in Figure 2(b) and the translation in Figure 2 by using the rules in Table 1. ▲ means tokenization while ■ represents translation.

3 Joint Tokenization and Translation

3.1 Model

We take a next step towards the direction of offering more tokenizations to SMT systems by proposing **joint** tokenization and translation. As shown in Figure 1(b), the decoder takes an untokenized string as input, and then tokenizes the source side string while building the corresponding translation of the target side. Since the traditional rules like those in Table 1 natively include tokenization information, we can directly apply them for simultaneous construction of tokenization and translation by the source side and target side of rules respectively. In Figure 3, our joint model takes the character sequence in Figure 2(a) as input, and synchronously conducts both translation and tokenization using the rules in Table 1.

As our model conducts tokenization during decoding, we can integrate tokenization models as features together with translation features under the discriminative framework. We expect tokenization and translation could collaborate with each other. Tokenization offers translation with good tokenized results, while translation helps tokenization to eliminate ambiguity. Formally, the probability of a derivation D is represented as

$$P(D) \propto \prod_i \phi_i(D)^{\lambda_i} \tag{1}$$

where ϕ_i are features defined on derivations including translation and tokenization, and λ_i are feature weights. We totally use 16 features:

- 8 traditional translation features (Chiang, 2007): 4 rule scores (direct and reverse translation scores; direct and reverse lexical translation scores); language model of the target side; 3 penalties for word count, extracted rule and glue rule.
- 8 tokenization features: maximum entropy model, language model and word count of the source side (Section 3.2). To handle the Out Of Vocabulary (OOV) problem (Section 3.3), we also introduce 5 OOV features: OOV character count and 4 OOV discount features.

Since our model is still a string-based model, the CKY algorithm and cube pruning are still applicable for our model to find the derivation with max score.

3.2 Adding Tokenization Features

Maximum Entropy model (ME). We first introduce ME model feature for tokenization by casting it as a labeling problem (Xue and Shen, 2003; Ng and Low, 2004). We label a character with the following 4 types:

- **b**: the **b**egin of a word
- **m**: the **m**iddle of a word
- **e**: the **e**nd of a word
- **s**: a **s**ingle-character word

Taking the tokenization *you-wang* of the string *you wang* for example, we first create a label sequence *b e* for the tokenization *you-wang* and then calculate the probability of tokenization by

$$\begin{aligned} & P(\textit{you-wang} \mid \textit{you wang}) \\ & = P(\mathrm{b\ e} \mid \textit{you wang}) \\ & = P(\mathrm{b} \mid \textit{you}, \textit{you wang}) \\ & \quad \times P(\mathrm{e} \mid \textit{wang}, \textit{you wang}) \end{aligned}$$

Given a tokenization w_1^L with L words for a character sequence c_1^n, we firstly create labels l_1^n for every characters and then calculate the probability by

$$P(w_1^L|c_1^n) = P(l_1^n|c_1^n) = \prod_{i=1}^{n} P(l_i|c_i, c_1^n) \tag{2}$$

Under the ME framework, the probability of assigning the character c with the label l is represented as:

$$P(l|c, c_1^n) = \frac{\exp[\sum_i \lambda_i h_i(l, c, c_1^n)]}{\sum_{l'} \exp[\sum_i \lambda_i h_i(l', c, c_1^n)]} \qquad (3)$$

where h_i is feature function, λ_i is the feature weight of h_i. We use the feature templates the same as Jiang et al., (2008) to extract features for ME model. Since we directly construct tokenization when decoding, it is straight to calculate the ME model score of a tokenization according to formula (2) and (3).

Language Model (LM). We also use the n-gram language model to calculate the probability of a tokenization w_1^L:

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i | w_{i-n+1}^{i-1}) \qquad (4)$$

For instance, we compute the probability of the tokenization shown in Figure 2(b) under a 3-gram model by

$$\begin{aligned} & P(\textit{tao-fei-ke}) \\ \times & P(\textit{you-wang} \mid \textit{tao-fei-ke}) \\ \times & P(\textit{duo} \mid \textit{tao-fei-ke}, \textit{you-wang}) \\ \times & P(\textit{fen} \mid \textit{you-wang}, \textit{duo}) \end{aligned}$$

Word Count (WC). This feature counts the number of words in a tokenization. Language model is prone to assign higher probabilities to short sentences in a biased way. This feature can compensate this bias by encouraging long sentences. Furthermore, using this feature, we can optimize the granularity of tokenization for translation. If larger granularity is preferable for translation, then we can use this feature to punish the tokenization containing more words.

3.3 Considering All Tokenizations

Obviously, we can construct the potential tokenizations and translations by only using the extracted rules, in line with traditional translation decoding. However, it may limits the potential tokenization space. Consider a string *you wang*. If *you-wang* is not reachable by the extracted rules, the tokenization *you-wang* will never be considered under this way. However, the decoder may still create a derivation by splitting the string as small as possible with tokenization *you wang* and translating *you* with *a* and *wang* with *look*, which may hurt the translation performance. This case happens frequently for named entity especially. Overall, it is necessary to assure that the decoder can derive all potential tokenizations (Section 4.1.3).

To assure that, when a span is not tokenized into a single word by the extracted rules, we will add an operation, which is considering the entire span as an OOV. That is, we tokenize the entire span into a single word with a translation that is the copy of source side. We can define the set of all potential tokenizations $\tau(c_1^n)$ for the character sequence c_1^n in a recursive way by

$$\tau(c_1^n) = \bigcup_{i}^{n-1} \{\tau(c_1^i) \bigotimes \{w(c_{i+1}^n)\}\} \qquad (5)$$

here $w(c_{i+1}^n)$ means a word contains characters c_{i+1}^n and $\bigotimes$ means the times of two sets. According to this recursive definition, it is easy to prove that all tokenizations is reachable by using the glue rule ($S \Rightarrow SX, SX$) and the added operation. Here, glue rule is used to concatenate the translation and tokenization of the two variables S and X, which acts the role of the operator $\bigotimes$ in equation (5).

Consequently, this introduces a large number of OOVs. In order to control the generation of OOVs, we introduce the following OOV features:

OOV Character Count (OCC). This feature counts the number of characters covered by OOV. We can control the number of OOV characters by this feature. It counts 3 when *tao-fei-ke* is an OOV, since *tao-fei-ke* has 3 characters.

OOV Discount (OD). The chances to be OOVs vary for words with different counts of characters. We can directly attack this problem by adding features OD_i that reward or punish OOV words which contains with i characters, or $OD_{i,j}$ for OOVs contains with i to j characters. 4 OD features are used in this paper: 1, 2, 3 and 4+. For example, OD_3 counts 1 when the word *tao-fei-ke* is an OOV.

Method	Train	#Rule	Test	TFs	MT04	MT05	Speed
Separate	ICT	151M	ICT	×	34.82	33.06	2.48
	SF	148M	SF	×	35.29	33.22	2.55
	ME	141M	ME	×	33.71	30.91	2.34
	All	219M	Lattice	×	35.79	33.95	3.83
				√	35.85	33.76	6.79
Joint	ICT	151M	Character	√	36.92	34.69	17.66
	SF	148M			37.02	34.56	17.37
	ME	141M			36.78	34.17	17.23
	All	219M			**37.25****	**34.88****	17.52

Table 2: Comparison of Separate and Joint methods in terms of BLEU and speed (second per sentence). Columns *Train* and *Test* represents the tokenization methods for training and testing respectively. Column *TFs* stands for whether the 8 tokenization features is used (√) or not (×). *ICT*, *SF* and *ME* are segmenter names for preprocessing. *All* means combined corpus processed by the three segmenters. Lattice represent the system implemented as Dyer et al., (2008). ** means significantly (Koehn, 2004) better than Lattice ($p < 0.01$).

4 Experiments

In this section, we try to answer the following questions:

1. Does the joint method outperform conventional methods that separate tokenization from decoding. (Section 4.1)

2. How about the tokenization performance of the joint decoder? (Section 4.2)

4.1 Translation Evaluation

We use the SCFG model (Chiang, 2007) for our experiments. We firstly work on the Chinese-English translation task. The bilingual training data contains 1.5M sentence pairs coming from LDC data.[1] The monolingual data for training English language model includes Xinhua portion of the GIGAWORD corpus, which contains 238M English words. We use the NIST evaluation sets of 2002 (MT02) as our development data set, and sets of 2004(MT04) and 2005(MT05) as test sets. We use the corpus derived from the People's Daily (Renmin Ribao) in Feb. to Jun. 1998 containing 6M words for training LM and ME tokenization models.

Translation Part. We used GIZA++ (Och and Ney, 2003) to perform word alignment in both directions, and grow-diag-final-and (Koehn et al., 2003) to generate symmetric word alignment. We extracted the SCFG rules as describing in Chiang (2007). The language model were trained by the SRILM toolkit (Stolcke, 2002).[2] Case insensitive NIST BLEU (Papineni et al., 2002) was used to measure translation performance.

Tokenization Part. We used the toolkit implemented by Zhang (2004) to train the ME model. Three Chinese word segmenters were used for comparing: ICTCLAS (*ICT*) developed by institute of Computing Technology Chinese Academy of Sciences (Zhang et al., 2003); *SF* developed at Stanford University (Huihsin et al., 2005) and *ME* which exploits the ME model described in section (3.2).

4.1.1 Joint Vs. Separate

We compared our joint tokenization and translation with the conventional separate methods. The input of separate tokenization and translation can either be a single segmentation or a lattice. The lattice combines the 1-best segmentations of segmenters. Same as Dyer et al., (2008), we also extracted rules from a combined bilingual corpus which contains three copies from different segmenters. We refer to this version of rules as *All*.

Table 2 shows the result.[3] Using *all* rule table, our joint method significantly outperforms the best single system *SF* by $+1.96$ and $+1.66$ points on MT04 and MT05 respectively, and also outperforms the lattice-based system by $+1.46$ and $+0.93$ points. However, the 8 tokenization features have small impact on the lattice system, probably because the tokenization space limited

[1] including LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06

[2] The calculation of LM probabilities for OOVs is done by the SRILM without special treatment by ourself.

[3] The weights are retrained for different test conditions, so do the experiments in other sections.

ME	LM	WC	OCC	OD	MT05
×	×	×	×	×	24.97
√	×	×	×	×	25.30
×	√	×	×	×	24.70
×	×	√	×	×	24.84
×	×	×	√	×	25.51
×	×	×	×	√	25.34
×	√	√	×	×	25.74
√	√	√	√	√	26.37

Table 3: Effect of tokenization features on Chinese-English translation task. "√" denotes using a tokenization feature while "×" denotes that it is inactive.

Method	BLEU	#Word	Grau	#OOV
ICT	33.06	30,602	1.65	644
SF	33.22	30,119	1.68	882
ME	30.91	29,717	1.70	1,614
Lattice	33.95	30,315	1.66	494
Joint$_{ICT}$	34.69	29,723	1.70	996
Joint$_{SF}$	34.56	29,839	1.69	972
Joint$_{ME}$	34.17	29,771	1.70	1,062
Joint$_{All}$	34.88	29,644	1.70	883

Table 4: Granularity (Grau, counts of character per word) and counts of OOV words of different methods on MT05. The subscript of joint means the type of rule table.

by lattice has been created from good tokenization. Not surprisingly, our decoding method is about 2.6 times slower than lattice method with tokenization features, since the joint decoder takes character sequences as input, which is about 1.7 times longer than the corresponding word sequences tokenized by segmenters. (Section 4.1.4).

The number of extracted rules with different segment methods are quite close, while the *All* version contains about 45% more rules than the single systems. With the same rule table, our joint method improves the performance over separate method up to $+3.03$ and $+3.26$ points (*ME*). Interestingly, comparing with the separate method, the tokenization of training data has smaller effect on joint method. The BLEU scores of MT04 and MT05 fluctuate about 0.5 and 0.7 points when applying the joint method, while the difference of separate method is up to 2 and 3 points respectively. It shows that the joint method is more robust to segmentation performance.

4.1.2 Effect of Tokenization Model

We also investigated the effect of tokenization features on translation. In order to reduce the time for tuning weights and decoding, we extracted rules from the FBIS part of the bilingual corpus, and trained a 4-gram English language model on the English side of FBIS.

Table 3 shows the result. Only using the 8 translation features, our system achieves a BLEU score of 24.97. By activating all tokenization features, the joint decoder obtains an absolute improvement by 1.4 BLEU points. When only adding one single tokenization feature, the *LM* and *WC* fail to show improvement, which may result from their bias to short or long tokenizations. However, these two features have complementary advantages and collaborate well when using them together (line 8). The OCC and OD features also contribute improvements which reflects the fact that handling the generation of OOV is important for the joint model.

4.1.3 Considering All Tokenizations?

In order to explain the necessary of considering all potential tokenizations, we compare the performances of whether to tokenize a span as a single word or not as illustrated in section 3.3. When only tokenizing by the extracted rules, we obtain 34.37 BLEU on MT05, which is about 0.5 points lower than considering all tokenizations shown in Table 2. This indicates that spuriously limitation of the tokenization space may degenerate translation performance.

4.1.4 Results Analysis

To better understand why the joint method can improve the translation quality, this section shows some details of the results on the MT05 data set.

Table 4 shows the granularity and OOV word counts of different configurations. The lattice method reduces the OOV words quite a lot which is 23% and 70% comparing with ICT and ME. In contrast, the joint method gain an absolute improvement even thought the OOV count do not decrease. It seems the lattice method prefers to translate more characters (since smaller granularity and less OOVs), while our method is inclined to maintain integrity of words (since larger granularity and more OOVs). This also explains the difficulty of deciding optimal tokenization for translation before decoding.

There are some named entities or idioms that

Method	Type	F_1	Time
Monolingual	ICT	97.47	0.010
	SF	97.48	0.007
	ME	95.53	0.008
Joint	ICT	97.68	9.382
	SF	97.68	10.454
	ME	97.60	10.451
	All	97.70	9.248

Table 5: Comparison of segmentation performance in terms of F_1 score and speed (second per sentence). *Type* column means the segmenter for monolingual method, while represents the rule tables used by joint method.

are split into smaller granularity by the segmenters. For example:"史 东" which is an English name "Stone" or "豆-蔻 年-华" which means "teenage". Although the separate method is possible to translate them using smaller granularity, the translation results are in fact wrong. In contrast, the joint method tokenizes them as entire OOV words, however, it may result a better translation for the whole sentence.

We also count the overlap of the segments used by the $Joint_{All}$ system towards the single segmentation systems. The tokenization result of $Joint_{All}$ contains $29,644$ words, and shares $28,159$, $27,772$ and $27,407$ words with ICT, SF and ME respectively. And 46 unique words appear only in the joint method, where most of them are named entity.

4.2 Chinese Word Segmentation Evaluation

We also test the tokenization performance of our model on Chinese word segmentation task. We randomly selected 3k sentences from the corpus of People's Daily in Jan. 1998. 1k sentences were used for tuning weights, while the other 2k sentences were for testing. We use MERT (Och, 2003) to tune the weights by minimizing the error measured by F_1 score.

As shown in Table 5, with all features activated, our joint decoder achieves an F_1 score of 97.70 which reduces the tokenization error comparing with the best single segmenter *ICT* by 8.7%. Similar to the translation performance evaluation, our joint decoder outperforms the best segmenter with any version of rule tables.

Feature	F_1
TFs	97.37
TFs + RS	97.65
TFs + LM	97.67
TFs + RS + LM	97.62
All	97.70

Table 6: Effect of the target side information on Chinese word segmentation. *TFs* stands for the 8 tokenization features. *All* represents all the 16 features.

4.2.1 Effect of Target Side Information

We compared the effect of the 4 Rule Scores (RS), target side Language Model (LM) on tokenization. Table 6 shows the effect on Chinese word segmentation. When only use tokenization features, our joint decoder achieves an F_1 score of 97.37. Only integrating language model or rule scores, the joint decoder achieves an absolute improvement of 0.3 point in F_1 score, which reduces the error rate by 11.4%. However, when combining them together, the F_1 score deduces slightly, which may result from the weight tuning. Using all feature, the performance comes to 97.70. Overall, our experiment shows that the target side information can improve the source side tokenization under a supervised way, and outperform state-of-the-art systems.

4.2.2 Best Tokenization = Best Translation?

Previous works (Zhang et al., 2008; Chang et al., 2008) have shown that preprocessing the input string for decoder by better segmenters do not always improve the translation quality, we reverify this by testing whether the joint decoder produces good tokenization and good translation at the same time. To answer the question, we used the feature weights optimized by maximizing BLEU for tokenization and used the weights optimized by maximizing F_1 for translation. We test BLEU on MT05 and F_1 score on the test data used in segmentation evaluation experiments. By tuning weights regarding to BLEU (the configuration for Joint$_{All}$ in table 2), our decoder achieves a BLEU score of 34.88 and an F_1 score of 92.49. Similarly, maximizing F_1 (the configuration for the last line in table 6) leads to a much lower BLEU of 27.43, although the F_1 is up to 97.70. This suggests that better tokenization may not always lead to better translations and vice versa

Rule	#Rule	Method	Test	Time
Morph	46M	Separate	21.61	4.12
Refined	55M		21.21	4.63
All	74M	Joint	21.93*	5.10

Table 7: Comparison of Separate and Joint method in terms of BLEU score and decoding speed (second per sentence) on Korean-Chinese translation task.

even by the joint decoding. This also indicates the hard of artificially defining the best tokenization for translation.

4.3 Korean-Chinese Translation

We also test our model on a quite different task: Korean-Chinese. Korean is an agglutinative language, which comes from different language family comparing with Chinese.

We used a newswire corpus containing 256k sentence pairs as training data. The development and test data set contain 1K sentence each with one single reference. We used the target side of training set for language model training. The Korean part of these data were tokenized into morpheme sequence as atomic unit for our experiments.

We compared three methods. First is directly use morpheme sequence (Morph). The second one is refined data (Refined), where we use selective morphological segmentation (Oflazer, 2008) for combining morpheme together on the training data. Since the selective method needs alignment information which is unavailable in the decoding, the test data is still of morpheme sequence. These two methods still used traditional decoding method. The third one extracting rules from combined (All) data of methods 1 and 2, and using joint decoder to exploit the different granularity of rules.

Table 7 shows the result. Since there is no gold standard data for tokenization, we do not use ME and LM tokenization features here. However, our joint method can still significantly ($p < 0.05$) improve the performance by about +0.3 points. This also reflects the importance of optimizing granularity for morphological complex languages.

5 Related Work

Methods have been proposed to optimize tokenization for word alignment. For example, word alignment can be simplified by packing (Ma et al., 2007) several consecutive words together. Word alignment and tokenization can also be optimized by maximizing the likelihood of bilingual corpus (Chung and Gildea, 2009; Xu et al., 2008). In fact, these work are orthogonal to our joint method, since they focus on training step while we are concerned of decoding. We believe we can further the performance by combining these two kinds of work.

Our work also has connections to multilingual tokenization (Snyder and Barzilay, 2008). While they have verified that tokenization can be improved by multilingual learning, our work shows that we can also improve tokenization by collaborating with translation task in a supervised way.

More recently, Liu and Liu (2010) also shows the effect of joint method. They integrate parsing and translation into a single step and improve the performance of translation significantly.

6 Conclusion

We have presented a novel method for joint tokenization and translation which directly combines the tokenization model into the decoding phase. Allowing tokenization and translation to collaborate with each other, tokenization can be optimized for translation, while translation also makes contribution to tokenization performance under a supervised way. We believe that our approach can be applied to other string-based model such as phrase-based model (Koehn et al., 2003), string-to-tree model (Galley et al., 2006) and string-to-dependency model (Shen et al., 2008).

Acknowledgement

The authors were supported by SK Telecom C&I Business, and National Natural Science Foundation of China, Contracts 60736014 and 60903138. We thank the anonymous reviewers for their insightful comments. We are also grateful to Wenbin Jiang, Zhiyang Wang and Zongcheng Ji for their helpful feedback.

References

Chang, Pi-Chuan, Michel Galley, and Christopher D. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *the Third Workshop on SMT*.

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Chung, Tagyoung and Daniel Gildea. 2009. Unsupervised tokenization for machine translation. In *Proc. EMNLP 2009*.

Dyer, Christopher, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. In *Proc. ACL 2008*.

Dyer, Chris. 2009. Using a maximum entropy model to build segmentation lattices for mt. In *Proc. NAACL 2009*.

Galley, Michel, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. ACL 2006*.

Huihsin, Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter. In *Fourth SIGHAN Workshop*.

Jiang, Wenbin, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proc. ACL 2008*.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. HLT-NAACL 2003*.

Koehn, Philipp. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP 2004*.

Liu, Yang and Qun Liu. 2010. Joint parsing and translation. In *Proc. Coling 2010*.

Liu, Yang, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. ACL 2006*.

Ma, Yanjun, Nicolas Stroppa, and Andy Way. 2007. Bootstrapping word alignment via word packing. In *Proc. ACL 2007*.

Mi, Haitao, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proc. of ACL 2008*.

Ng, Hwee Tou and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In *Proc. EMNLP 2004*.

Och, Franz J. and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. ACL 2002*.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, Franz Josef. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL 2003*.

Oflazer, Kemal. 2008. Statistical machine translation into a morphologically complex language. In *Proc. CICL 2008*.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. ACL 2002*.

Shen, Libin, Xu Jinxi, and Weischedel Ralph. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. ACL 2008*.

Snyder, Benjamin and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proc. ACL 2008*.

Stolcke, Andreas. 2002. Srilm – an extensible language modeling toolkit.

Xu, Jia, Evgeny Matusov, Richard Zens, and Hermann Ney. 2005. Integrated chinese word segmentation in statistical machine translation. In *Proc. IWSLT2005*.

Xu, Jia, Jianfeng Gao, Kristina Toutanova, and Hermann Ney. 2008. Bayesian semi-supervised chinese word segmentation for statistical machine translation. In *Proc. Coling 2008*.

Xue, Nianwen and Libin Shen. 2003. Chinese word segmentation as LMR tagging. In *SIGHAN Workshop*.

Zhang, Hua-Ping, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. Hhmm-based chinese lexical analyzer ictclas. In *the Second SIGHAN Workshop*.

Zhang, Ruiqiang, Keiji Yasuda, and Eiichiro Sumita. 2008. Improved statistical machine translation by multiple Chinese word segmentation. In *the Third Workshop on SMT*.

Zhang, Le. 2004. Maximum entropy modeling toolkit for python and c++.

Build Chinese Emotion Lexicons Using A Graph-based Algorithm and Multiple Resources

Ge Xu, Xinfan Meng, Houfeng Wang
Key Laboratory of Computational Linguistics (Peking University), Ministry of Education
Institute of Computational Linguistics, Peking University
{xuge, mxf, wanghf}@pku.edu.cn

Abstract

For sentiment analysis, lexicons play an important role in many related tasks. In this paper, aiming to build Chinese emotion lexicons for public use, we adopted a graph-based algorithm which ranks words according to a few seed emotion words. The ranking algorithm exploits the similarity between words, and uses multiple similarity metrics which can be derived from dictionaries, unlabeled corpora or heuristic rules. To evaluate the adopted algorithm and resources, two independent judges were asked to label the top words of ranking list.

It is observed that noise is almost unavoidable due to imprecise similarity metrics between words. So, to guarantee the quality of emotion lexicons, we use an iterative feedback to combine manual labeling and the automatic ranking algorithm above. We also compared our newly constructed Chinese emotion lexicons (happiness, anger, sadness, fear and surprise) with existing counterparts, and related analysis is offered.

1 Introduction

Emotion lexicons have a great impact on the results of related tasks. With high-quality emotion lexicons, systems using simple methods can achieve competitive performance. However, to manually build an emotion lexicon is time-consuming. Many research works in building lexicons use automatic methods to assist the building procedure. Such works commonly rank words by the similarities to a set of seed words, then those words with high ranking scores are more likely to be added to the final lexicons or used as additional seed words.

For Chinese, emotion lexicons are scarce resources. We can get a small set of emotion words from semantic dictionary (such as CCD, HowNet, synonym dictionaries) or directly from related papers (Xu and Tao, 2003) (Chen et al. , 2009), but it is often not sufficient for practical systems. Xu et al. (2008) constructed a large-scale emotion ontology dictionary, but it is not publicly available yet.

In this paper, we adopted a graph-based algorithm to automatically rank words according to a few seed words. Similarity between words can be utilized and multiple resources are used to boost performance. Combining manual labeling with automatic ranking through an iterative feedback framework, we can produce high-quality emotion lexicons. Our experiments focused on Chinese, but the method is applicable to any other language as long as suitable resources exist.

The remainder of this paper is organized as follows. In Section 2, related works are introduced. In Section 3, we describe a graph-based algorithm and how to incorporate multiple resources. Section 4 gives the details of applying the algorithm on five emotions and shows how to evaluate the results. Section 5 focuses on how to build and evaluate emotion lexicons, linguistic consideration and instruction for identifying emotions are also included. Finally, conclusion is made in Section 6.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1209–1217,
Beijing, August 2010

2 Related work

Riloff and Shepherd (1997) presented a corpus-based method that can be used to build semantic lexicons for specific categories. The input to the system is a small set of seed words for a category and a representative text corpus. The output is a ranked list of words that are associated with the category. An approach proposed by (Turney, 2002) for the construction of polarity started with a few positive and negative seeds, then used a similarity method (pointwise mutual information) to grow this seed list from web corpus. Our experiments are similar with these works, but we use a different ranking method and incorporate multiple resources. To perform rating inference on reviews, Goldberg and Zhu (2006) created a graph on both labeled and unlabeled reviews, and then solved an optimization problem to obtain a smooth rating function over the whole graph. Rao and Ravichandran (2009) used three semi-supervised methods in polarity lexicon induction based on WordNet, and compared them with corpus-based methods. Encouraging results show methods using similarity between words can improve the performance. Wan and Xiao (2009) presented a method to use two types of similarity between sentences for document summarization, namely similarity within a document and similarity between documents. The ranking method in our paper is similar to the ones used in above three papers, which fully exploit the relationship between any pair of sample points (both labeled and unlabeled). When only limited labeled data are available, such method achieves significantly better predictive accuracy over other methods that ignore the unlabeled examples during training.

Xu et al. (2008) at first formed a taxonomy for emotions, under which an affective lexicon ontology exploiting various resources was constructed. The framework of ontology is filled by the combination of manual classification and automatic methods. To our best knowledge, this affective lexicon ontology is the largest Chinese emotion-oriented dictionary.

3 Our method

3.1 A graph-based algorithm

For our experiments, we chose the graph-based algorithm in (Zhou et al. , 2004) which is transductive learning and formulated as follows:

Given a point set $\chi = \{x_1, ..., x_l, x_{l+1}, ..., x_n\}$, the first l points $x_i (i \leq l)$ are labeled and the remaining points $x_u (l+1 \leq u \leq n)$ unlabeled. The goal is to rank the unlabeled points.

Let F denotes an n-dimensional vector whose elements correspond to ranking scores on the data set χ. Define another n-dimensional vector Y with $Y_i = 1$ if x_i is labeled and $Y_i = 0$ otherwise. Y denotes the initial label assignment.

The iterative algorithm is shown in the following:

Algorithm 1 A graph-based algorithm

```
1. Construct the weight matrix $W$ and set $W_{ii}$ to zero to avoid self-reinforcement. $W$ is domain-dependent.
2. Construct the similarity matrix $S = D^{1/2}WD^{1/2}$ using symmetric normalization. $D$ is a diagonal matrix with $D_{ii} = \Sigma_j W_{ij}$.
3. Iterate $F(t+1) = \alpha SF(t) + (1-\alpha)Y$ until convergence, where $\alpha$ is a parameter in (0, 1), and $F(0) = Y$. We clamp labeled points to 1 after each iteration.
4. Let $F^*$ denote $F(t)$ when the iteration converges.
```

In our experiments, labeled points are seed emotion words, S_{ij} denotes the similarity between *i*th word and *j*th word. In an iteration, each word absorbs label information from other words. More similar two words are, more influence they have on each other. The label information (initially from seed emotion words) will propagate along S. The final output F^* contains ranking scores for all words, and a score indicates how similar the corresponding word is to the seed emotion words.

The implementation of the iterative algorithm is theoretically simple, which only involves basic matrix operation. Compared with methods which do not exploit the relationship between samples, experiments showing advantages of graph-based learning methods can be found in (Rao and Ravichandran, 2009),(Goldberg and Zhu, 2006),(Tong et al. , 2005),(Wan and Xiao, 2009),(Zhu and Ghahramani, 2002) etc. When labeled data are scarce, such graph-based transductive learning methods are especially useful.

3.2 Incorporate multiple resources

For building the emotion lexicons, we are faced with lots of resources, such as semantic dictionaries, labeled or unlabeled corpora, and some linguistic experiences which can be presented as heuristic rules. Naturally we want to use these resources together, thus boosting the final performance. In graph-base setting, such resources can be used to construct the emotion-oriented similarity between words, and similarities will be represented by matrices.

The schemes to fuse similarity matrices are presented in (Sindhwani et al. , 2005), (Zhou and Burges, 2007), (Wan and Xiao, 2009) and (Tong et al. , 2005) etc. In our paper, not aiming at comparing different fusion schemes, we used a linear fusion scheme to fuse different similarities matrices from different resources. The scheme is actually a convex combination of matrices, with weights specified empirically.

The fusion of different similarity matrices falls in the domain of multi-view learning. A well-known multi-view learning method is Co-Training, which uses two views (two resources) to train two interactive classifiers (Blum and Mitchell, 1998). Since we focus on building emotion lexicons using multiple resources (multiple views), those who want to see the advantages of multi-view learning over learning with one view can refer to (Blum and Mitchell, 1998), (Sindhwani et al. , 2005), (Zhou and Burges, 2007), (Wan and Xiao, 2009) and (Tong et al. , 2005) etc.

4 Experiments

We use the method in section 3 to rank for each emotion with a few seed emotion words. Once we implement the ranking algorithm 1, the main work resides in constructing similarity matrices, which are highly domain-dependent.

4.1 Construct similarity matrices

Here, we introduce how to construct four similarity matrices used in building emotion lexicons. Three of them are based on cooccurrence of words; the fourth matrix is from a heuristic rule.

We use ictclas3.0[1] to perform word segmentation and POS tagging.

In our experiments, the number of words involved in ranking is 93506[2], so theoretically, the matrices are 93506 × 93506. If the similarity between any pair of words is considered, the computation becomes impractical in both time and space cost. So we require that each word has at most 500 nearest neighbors.

Four matrices are constructed as follows:

4.1.1 Similarity based on a unlabeled corpus

The unlabeled corpus used is People's Daily[3](人民日报1997∼2004). After word segmentation and POS tagging, we chose three POS's (i,a,l)[4]. The nouns were not included to limit the scale of word space. We set the cooccurrence window to a sentence, and removed the duplicate occurrences of words. Any pair of words in a sentence will contribute a unit weight to the edge which connects the pair of words.

4.1.2 Similarity based on a synonym dictionary

We used the Chinese synonym dictionary (哈工大同义词词林扩展版[5]) for this matrix. In this dictionary, the words in a synonym set are presented in one line and separated by spaces, so there is no need to perform word segmentation and POS tagging. Any pair of words in one line will contribute a unit weight to the edge which connects the pair of words.

4.1.3 Similarity based on a semantic dictionary

We used The Contemporary Chinese Dictionary (现代汉语词典) to construct the third simi-

[1] downloaded from http://www.ictclas.org/

[2] Words are selected after word segmentation and POS tagging, see section 4.1.1∼4.1.3 for selection of words in details.

[3] http://icl.pku.edu.cn/

[4] i=Chinese idiom, a=adjective, l=Chinese phrase

[5] http://ir.hit.edu.cn/

larity matrix. Since word segmentation may segment the entries of the dictionary, we extracted all the entries in the dictionary and store them in a file whose words ictclas3.0 was required not to segment. Furthermore, for an entry in the dictionary, the example sentences or phrases appearing in its gloss may contain many irrelevant words in terms of emotions, so they were removed from the gloss.

After word segmentation and POS tagging[6], we set the cooccurrence window to one line (an entry and its gloss without example sentences or phrases), and removed the duplicate occurrences of words. An entry and any word in the modified gloss will contribute a unit weight to the edge which connects the pair of words. This constructing was a bit different, since we did not consider the similarity between words in modified gloss.

4.1.4 similarity based on a heuristic rule

In Chinese, a word is composed of one or several Chinese characters. A Chinese character is normally by itself an independent semantic unit, so the similarity between two words can be inferred from the character(s) that they share. For example, the Chinese word 欣 (happy) appears in the word 欣然 (readily). Since 欣然 and 欣 share one Chinese character, they are regarded as similar. Naturally, the larger the proportion that two words share, the more similar they are. In this way, the fourth weighted matrix was formed. To avoid incurring noises, we exclude the cases where one Chinese character is shared, with the exception that the Chinese character itself is one of the two Chinese words.

4.1.5 Fusion of four similarity matrices

After processing all the lines (or sentences), the weighted matrices are normalized as in algorithm 1, then four similarity matrices are linearly fused with equal weights (1/4 for each matrix).

4.2 Select seed emotion words

In our experiments, we chose emotions of *happiness, sadness, anger, fear and surprise* which are widely accepted as basic emotions[7]. Empirically, we assigned each emotion with seed words given in Table 1.

Emotion	**Seed words**
喜(happiness)	高兴, 愉快, 欢乐, 喜悦, 兴高采烈, 欢畅, 开心
怒(anger)	愤怒, 不满, 恼火, 生气, 愤恨, 恼怒, 愤懑, 震怒, 悲愤, 窝火, 痛恨, 恨之入骨, 义愤填膺, 怒气冲天
哀(sadness)	悲伤, 沮丧, 痛苦, 伤心, 难过, 悲哀, 难受, 消沉, 灰心丧气, 悲戚, 闷闷不乐, 哀伤, 悲愤, 悲切, 悲痛欲绝, 欲哭无泪
惧(fear)	恐惧, 惧怕, 担心, 提心吊胆, 害怕, 惊恐, 疑惧, 畏惧, 不寒而栗, 望而生畏
惊(surprise)	惊讶, 大吃一惊 ,震惊, 惊恐, 惊异, 惊骇, 惊, 出乎意料, 惊喜, 惊叹

Table 1: Seed emotion words

4.3 Evaluation of our method

We obtained five ranking lists of words using the method in section 3. Following the work of (Riloff and Shepherd, 1997), we adopted the following evaluation setting.

To evaluate the quality of emotion ranking lists, each list was manually rated by two persons independently. For each emotion, we selected the top 200 words of each ranking list and presented them to judges. We presented the words in random order so that the judges had no idea how our system had ranked the words. The judges were asked to rate each word on a scale from 1 to 5 indicating how strongly it was associated with an emotion, 0 indicating no association. We allowed the judges to assign -1 to a word if they did not know what it meant. For the words rated as -1, we manually assigned ratings that we thought were appropriate.

The results of judges are shown in figures 1-5. In these figures, horizontal axes are the number of reviewed words in ranking lists and vertical axes are number of emotion words found (with 5 different strength). The curve labeled as $> x$ means that it counts the number of words which are rated

[6] since we do not segment entries in this dictionary, all POS's are possible

[7] Guidelines for identifying emotions is in section 5, before that, we understand emotions through common sense.

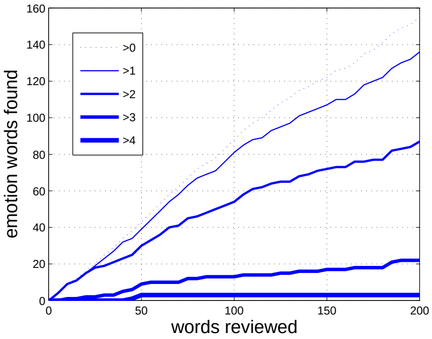


Figure 1: happiness

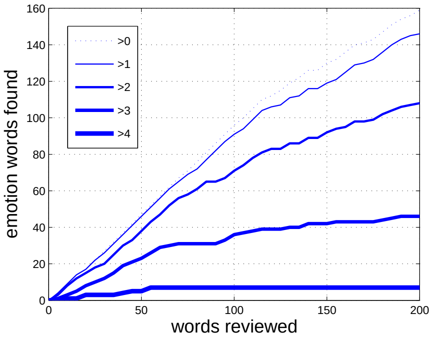


Figure 2: anger

greater than x by either judge.

Curves ($> 0, > 1, > 2$) display positive slopes even at the end of the 200 words, which implies that more emotion words would occur if more than 200 words are reviewed. By comparison, curves ($> 3, > 4$) tend to be flat when they are close to the right side, which means the cost of identifying high-quality emotion words will increase greatly as one checks along the ranking list in descendent order.

It is observed that words which both judges assign 5 are few. In *surprise* emotion, the number is even 0. Such results may reflect that emotion is harder to identify compared with topical categories in (Riloff and Shepherd, 1997).

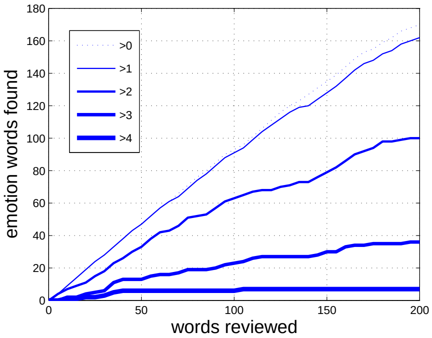


Figure 3: sadness

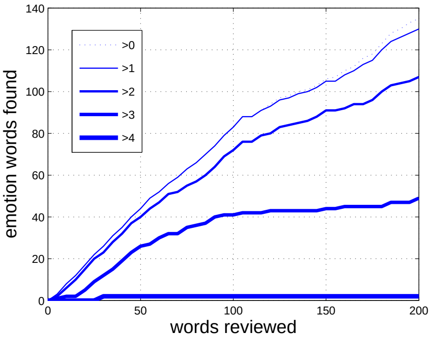


Figure 4: fear

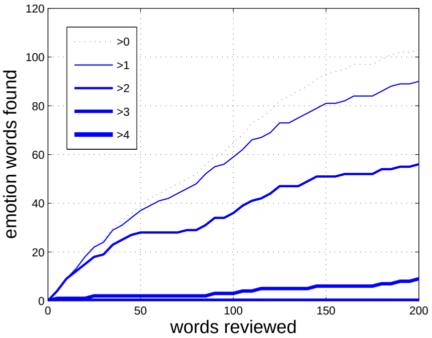


Figure 5: surprise

From the semantic dictionary, our method found many low-frequency emotion words such as 忭 (pleasant, glad), 蘧然 (surprise and happy), 忉怛 (sad), or those used in Chinese dialects such as 毛咕 (fear), 挂气 (angry). Such emotion words are necessary for comprehensive emotion lexicons.

Because more POS's than adjectives and verbs are included in our experiments, some emotion words such as the noun 冷门 (unexpected winner),and the adverb 竟然 (to one's surprise) are also spotted, which to some extent implies the generality of our method.

5 Construct emotion lexicons

The above section introduced a method to rank words with a few seed emotion words. However, to build emotion lexicons requires that we manually remove the noises incurred by the automatic ranking method. Accordingly, guidelines for identifying emotions are needed, and also some linguistic consideration in identifying emoting words should be given.

5.1 An iterative feedback to denoise

In our experiments, we observed that noises incurred by similarity matrices are almost unavoidable. For example, in the unlabeled corpus, 国事访问 (state visits) always co-occurred with 高兴 (happy) or 愉快 (happy), so in happiness emotion, 国事访问 acquired a high ranking position (174th); in terms of the heuristic rule, 意料 (expected) shares two Chinese characters with 出乎意料 (unexpected, surprised), however they have opposite meaning because 出乎 (exceed, beyond) is a negative word. 意料 unfavorably ranked high (88th) in surprise emotion; from the semantic dictionary, the gloss of 年画 (Chinese Spring Festival pictures) contains 欢乐 (happy), thus in happiness emotion, 年画 ranked high (158th).

So after each ranking of an emotion, in the descendent order of ranking scores, we manually revised some scores in about top 500. Several criteria (see 5.2 and 5.3) were given to guide if a word has a specified emotion. For those words surely bearing the specified emotion, we assigned 1 to them ,and left others unchanged. Seeing the words newly revised to be 1 as new seed emotion words, we run the ranking algorithm again. After such feedback was repeated 2∼3 times, we collected all the words labeled with 1 to form the final emotion lexicons. In (Zhou et al. , 2004), the author also suggested such *iterative feedback* to extend the query (seed) set and improve the ranking output. Commonly, the size of an emotion lexicon is small, so we do not have to check too many words.

The human revising procedure is sensitive to annotators' background. To improve the quality of the emotion lexicons, experts with linguistic or psychology background will help.

Furthermore, the ranking algorithm used in our paper is clearly sensitive to the initial seed words, but since we adopt an iterative feedback framework, the words not appearing in the initial set of seed words will show up in next iteration with high ranking scores. We also performed experiments which selected emotion seed words based on the Chinese synonym dictionary and the emotion words in (Chen et al. , 2009), similar results were found.

5.2 Guidelines for identifying emotions

The same as (Chen et al. , 2009), we used the definition that emotion is the felt awareness of bodily reactions to something perceived or thought. Also, we were highly influenced by the structure of the affective lexicon presented by (Ortony et al. , 1987), and used the Affective states and Affective-Behavioral conditions in the structure to identify emotion words in our paper[8].

With such guidelines, 胆小 (cowardice, relates more to external evaluation) is not an emotional word of fear. We also intentionally distinguish between emotions and expression of emotions. For example, 大笑 (laugh), 哈哈 (haw-haw) are seen as expression of happiness and 颤抖 (tremble) as of fear, but not as emotion words. In addition, we try to distinguish between an emotion and the cause of an emotion, see 5.3 for an example.

For each emotion, brief description is given as below[9]:

1. **Happiness**：the emotional reaction to something that is satisfying.
2. **Anger**：do not satisfy the current situation and have a desire to fight or change the situation. Often there exists a target for this emotion.
3. **Sadness**：an emotion characterized by feelings of disadvantage, loss, and helplessness. Sadness often leads to cry.
4. **Fear**：the emotional response to a perceived threat. Fear almost always relates to future events, such as worsening of a situation, or continuation of a situation that is unacceptable.
5. **Surprise**：the emotional reaction to something unexpected.

5.3 Linguistic consideration for identifying emotion words

If a word has multiple senses, we only consider its emotional one(s). For example, 生气 (as a verb, it means *be angry*, but means *vitality or spirits* as a noun) will appear in the emotion lexicon of anger.

[8] According to (Ortony et al. , 1987), *surprise* should not be seen as a basic emotion for it relates more to cognition. However, our paper focuses on the building of emotion lexicons, not the disputable issue of basic emotions

[9] we mainly referred to http://en.wikipedia.org/wiki

If one sense of a word is the combination of emotions, the word will appear in all related emotions.

We mainly consider four POS's, namely nouns, verbs, adjectives and adverb[10]. If a word has multiple POS's, we normally consider its POS with strongest emotion (Empirically, we think the emotion strength ranks in decedent order as following: adjectives, verbs, adverbs, nouns.). So we consider the verb of 恐惧 (fear) when it can be used as a noun and a verb in Chinese. The 生气 example above also applies here.

For each of four POS's, instruction for emotion identification is given as below:

Nouns: For example, 怒火 (rage, anger), 喜气 (joy or jubilation), 冷门 (an unexpected winner) are selected as emotion words. We distinguish between an emotion and the cause of an emotion. For example, calamity often leads to sadness, but does not directly contain the emotion of sadness. 冷门 appears in the surprise lexicon because we believe it contains surprise by itself.

Adverbs: The adverbs selected into emotion lexicons contain the emotions by themselves. For example, 竟然 (unexpectedly), 欣欣然 (cheerily), 气哼哼 (angrily), 蓦地 (unexpectedly), 伤心地 (sadly) etc.

Verbs: As in (Ortony et al. , 1987), Chinese emotion verbs also fall into at least two distinct classes, causatives and noncausatives. Both classes are included in our emotion lexicons. For example, 动肝火 (be angry), 担心 (fear) are noncausative verbs, while 激怒 (enrage), 震惊 (to make someone surprised) are causative ones. Probably due to the abundant usage of 令人/让人/使人 (to make someone) etc., causative emotion verbs are few compared to noncausative ones in Chinese.

Adjective：Quite a lot of emotion words fall in this POS, since adjectives are the natural expression of internal states of humans. For example, 高兴 (happy), 惊讶 (surprised), 愤怒 (angry) etc.

For any word that it is hard to identify at first sight, we used a search tool[11] to retrieve sentences which contain the word, and then identify if the word is emotional or not by its usage in the sentences.

5.4 Comparison with existing Chinese emotion resources

诧、骇、惊、讶、矍、蘧、愕、遽、骇然、赫然、竟然、居然、蘧然、愕然、愕然、矍然、爆冷、爆冷门、不料、不意、不虞、诧异、吃惊、出乎意料、出乎意外、出乎预料、出冷门、出其不意、出人意料、出人意外、触目惊心、错愕、大吃一惊、大惊失色、大惊小怪、怪讶、骇怪、骇然、骇人听闻、骇异、好家伙、赫然、赫然而怒、黑马、惊诧、惊呆、惊服、惊骇、惊慌、惊慌失措、惊惶、惊惶失措、惊魂未定、惊悸、惊惧、惊恐、惊恐万状、惊奇、惊人、惊世骇俗、惊叹、惊悉、惊喜、惊喜交集、惊喜万分、惊吓、惊羡、惊讶、惊疑、惊异、惊厥、惊愕、竟然、竟是、竟至、竟自、居然、冷不丁、冷不防、冷孤丁、冷门、没成想、猛不防、猛孤丁地、纳罕、始料不及、始料未及、受宠若惊、受惊、谁料、谁知、突如其来、未料、闻所未闻、想不到、心惊、心惊胆颤、心惊胆战、讶异、一语惊人、意料之外、意外、意想不到、又惊又喜、震惊、蓦地

Table 2: The emotion lexicon of surprise

Under the guidelines for manually identifying emotion words, we finally constructed five Chinese emotion lexicons using the iterative feedback. The newly constructed emotion lexicons were also reported as resources together with our paper. The emotion lexicon of *surprise* is shown in Table 2. In this part, we compare our lexicons with the following counterparts, see Table 3.

Ours1 in the table is the final emotion lexicons, and Ours2 is the abridged version that excludes the words of single Chinese character and Chinese idioms.

Chinese Concept Dictionary (CCD) is a WordNet-like semantic lexicon(Liu et al. , 2003).

[10] For Chinese idioms, we only considered those used as these four POS's, omitted those used as a statement, such as 哀兵必胜 (an army burning with righteous indignation is bound to win)

[11] provided by Center for Chinese Linguistics of Peking University, http://ccl.pku.edu.cn

	喜	怒	哀	惧	惊
CCD nouns	22	27	38	46	10
(Xu and Tao, 2003)	45	12	28	21	12
(Chen et al. , 2009)	28	34	28	17	11
(Xu et al. , 2008)	609	187	362	182	47
Ours1	95	118	97	106	99
Ours2	52	77	72	57	65

Table 3: Compare various emotion lexicons

We only considered the noun network which is richly developed in CCD, as in other semantic dictionaries. For each emotion, we chose its synset as well as the synsets of its hypernym and hyponym(s). In fact, most of words in the emotion nouns extracted can be used as verbs or adjectives in Chinese. However, since CCD is not designed for emotion analysis, words which are expression of emotions such as 哭泣 (cry) or evaluation such as 胆小 (cowardice) were included.

Selecting nouns and verbs, Xu and Tao (2003) offered an emotion taxonomy of 390 emotion words. The taxonomy contains 24 classes of emotions and excludes Chinese idioms. By our inspection to the offered emotion words in this taxonomy, the authors tried to exclude expression of emotions, evaluation and cause of emotions from emotions, which is similar with our processing[12]. Ours2 is intentionally created to compare with this emotion taxonomy.

Based on (Xu and Tao, 2003), Chen et al. (2009) removed the words of single Chinese character; let two persons to judge if a word is an emotional one and only those agreed by the two persons were seen as emotion words. It is worth noting that Chen et al. (2009) merges 怒 (anger) and 烦 (fidget) in (Xu and Tao, 2003) to form the 怒 (anger) lexicon, thus 讨厌 (dislike) appears in anger lexicon. However, we believe 讨厌 (dislike) is different with 怒 (anger), and should be put into another emotion. Also, we distinguish between 恨 (hate) and 怒 (anger).

Xu et al. (2008) constructed a large-scale affective lexicon ontology. Given the example words in their paper, we found that the authors did not intentionally exclude the expression of emotions such as 面红耳赤 (literally, red face and ear), 笑眯眯 (literally, be smiling). Such criteria of identifying emotion words may partially account for the large size of their emotion resources.

6 Conclusion and future work

In this paper, aiming to build Chinese emotion lexicons, we adopt a graph-based algorithm and incorporate multiple resources to improve the quality of lexicons and save human labor. This is an initial attempt to build Chinese emotion lexicons, the quality of constructed emotion lexicons is far from perfect and is supposed to be improved step by step.

The method in this paper can be further extended to subjectivity/polarity classification and other non-sentimental tasks such as word similarity computing, and can be also adapted to other languages. The more resources we use, the more human cost can be saved and the higher the quality of built emotion lexicons is.

In the future work, we want to construct other emotion lexicons such as 好 (like, love), 恶 (dislike), 欲 (desire) etc. using the same method.

Acknowledgement This research is supported by National Natural Science Foundation of China (No.60973053, No.90920011)

References

A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training。 *In Proceedings of the 11th Annual Conference on Computational Learning Theory*, 92-100.

Ying Chen, Sophia Y. M. Lee, and Churen Huang. 2009. A Cognitive-based Annotation System for Emotion Computing. *Proceedings of the Third Linguistic Annotation Workshop (LAW III).*

Andrew B. Goldberg, Xiaojin Zhu. 2006. Seeing stars when there aren't many stars: graph-based semi-supervised learning for sentiment categorization. *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing on the First Workshop on Graph Based Methods for Natural Language Processing.*

Y. Liu and et al. 2003. The CCD Construction Model and Its Auxiliary Tool VACOL. *Applied Linguistics, 45(1):83-88.*

A. Ortony, G. L. Clore, and M. A. Foss. 1987. The referential structure of the affective lexicon. *Cognitive Science, 11, 341-364.*

[12] Xu and Tao (2003) included words such as 情愿/愿意 (be willing to), 留神 (be careful) in their happiness lexicon, which we think should not be classified into happiness.

Delip Rao and D. Ravichandran. 2009. Semisupervised polarity lexicon induction. *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, 675-682.

Ellen Riloff and Jessica Shepherd. 1997. A Corpus-Based Approach for Building Semantic Lexicons. *In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing, pages 117-124*.

V. Sindhwani, P. Niyogi, and M. Belkin. 2005. A co-regularization approach to semisupervised learning with multiple views. *Proc. ICML Workshop on Learning with Multiple views*.

H. Tong, J. He, M. Li, C. Zhang, and W. Ma. 2005. Graph based multi-modality learning. *In Proceedings of the 13th Annual ACM international Conference on Multimedia. MULTIMEDIA '05. ACM*, New York, NY, 862-871.

Peter D. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. *ACL 2002*, 417-424.

Xiaojun Wan and Jianguo Xiao. 2009. Graph-Based Multi-Modality Learning for Topic-Focused MultiDocument Summarization. *IJCAI 2009*, 1586-1591.

Linhong Xu, Hongfei Lin, Yu Pan, Hui Ren and Jianmei Chen. 2008. Constructing the Afective Lexicon Ontology. *JOURNAL OF THE CHINA SOCIETY F0R SCIENTIFIC AND TECHNICAL INFORMATION Vo1.27 No.2, 180-185*.

X. Y. Xu, and J. H. Tao. 2003. The study of affective categorization in Chinese. *The 1st Chinese Conference on Affective Computing and Intelligent Interaction. Beijing, China*.

Hongbo Xu, Tianfang Yao, and Xuanjing Huang. 2009. The second Chinese Opinion Analysis Evaluation(in Chinese). *COAE 2009*.

D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Scholkopf. 2004. Learning with local and global consistency. *Advances in Neural Information Processing Systems 16. MIT Press, Cambridge, MA.* .

D. Zhou and C. J. C. Burges. 2007. Spectral clustering and transductive learning with multiple views. *Proceedings of the 24th international conference on Machine learning*.

X. Zhu and Z. Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. *Technical Report CMUCALD02107*. CMU.

A Methodology for Automatic Identification of Nocuous Ambiguity

Hui Yang[1] **Anne de Roeck**[1] **Alistair Willis**[1] **Bashar Nuseibeh**[1, 2]
[1]Department of Computing, The Open University
[2]Lero, University of Limerick
{h.yang, a.deroeck, a.g.willis, b.nuseibeh}@open.ac.uk

Abstract

Nocuous ambiguity occurs when a linguistic expression is interpreted differently by different readers in a given context. We present an approach to automatically identify nocuous ambiguity that is likely to lead to misunderstandings among readers. Our model is built on a machine learning architecture. It learns from a set of heuristics each of which predicts a factor that may lead a reader to favor a particular interpretation. An ambiguity threshold indicates the extent to which ambiguity can be tolerated in the application domain. Collections of human judgments are used to train heuristics and set ambiguity thresholds, and for evaluation. We report results from applying the methodology to coordination and anaphora ambiguity. Results show that the method can identify nocuous ambiguity in text, and may be widened to cover further types of ambiguity. We discuss approaches to evaluation.

1 Introduction

Traditional accounts of ambiguity have generally assumed that each use of a linguistic expression has a unique intended interpretation in context, and attempted to develop a model to determine it (Nakov and Hearst, 2005; Brill and Resnik, 1994). However, disambiguation is not always appropriate or even desirable (Poesio and Artstein, 2008). Ambiguous text may be interpreted differently by different readers, with no consensus about which reading is the intended one. Attempting to assign a preferred interpretation may therefore be inappropriate. Misunderstandings among readers do occur and may have undesirable consequences. In requirements engineering processes, for example, this results in costly implementation errors (Boyd et al., 2005).

Nonetheless, most text does not lead to significant misinterpretation. Our research aims to establish a model that estimates how likely an ambiguity is to lead to misunderstandings. Our previous work on nocuous ambiguity (Chantree et al., 2006; Willis et al., 2008) cast ambiguity not as a property of a text, but as a property of text in relation to a set of stakeholders. We drew on human judgments - interpretations held by a group of readers of a text – to establish criteria for judging the presence of nocuous ambiguity. An ambiguity is *innocuous* if it is read in the same way by different people, and *nocuous* otherwise. The model was tested on co-ordination ambiguity only.

In this paper, we implement, refine and extend the model. We investigate two typical ambiguity types arising from coordination and anaphora. We extend the previous work (Willis et al., 2008) with additional heuristics, and refine the concept of ambiguity threshold. We experiment with alternative machine learning algorithms to find optimal ways of combining the output of the heuristics. Yang et al. (2010a) describes a complete implementation in a prototype tool running on full text. Here we present our experimental results, to illustrate and evaluate the extended methodology.

The rest of the paper is structured as follows. Section 2 introduces the methodology for automatic detection of nocuous ambiguity. Sections 3 and 4 provide details on how the model is applied to coordination and anaphora ambiguity. Experimental setup and results are reported in Section 5, and discussed in Section 6. Section 7 reports on related work. Conclusions and future work are found in Section 8.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1218–1226,
Beijing, August 2010

2 Methodology for Nocuous Ambiguity Identification

This section describes the main ideas underpinning our model of ambiguity. We distinguish between structural and interpretative aspects. The former captures the fact that text may have structure (i.e. syntax) which, in principle, permits multiple readings. These are relatively straightforward to identify from the linguistic constructs present in the text. The latter acknowledges that if text is interpreted in the same way by different readers, it has a low risk of being misunderstood. Modelling interpretive aspects requires access to human judgments about texts. Our approach has three elements, which we describe in turn: collection of human judgments; heuristics that model those judgments, and a machine learning component to train the heuristics.

Human judgments. We define an ambiguity as nocuous if it gives rise to diverging interpretations. Wasow et al. (2003) suggests that ambiguity is always a product of the meaning that people assign to language, and thus a subjective phenomenon. We capture individual interpretations of instances of ambiguity by surveying participants, asking them for their interpretation. We use this information to decide whether, given some ambiguity threshold, a particular instance is seen as innocuous or nocuous depending on the degree of dissent between judges.

A key concept in determining when ambiguity is nocuous is the *ambiguity threshold.* Different application areas may need to be more or less tolerant of ambiguity (Poesio and Artstein, 2008). For instance, requirements documents describing safety critical systems should seek to avoid misunderstandings between stakeholders. Other cases, such as cookbooks, could be less sensitive. Willis et al. (2008)'s general concept of ambiguity threshold sought to implement a flexible tolerance level to nocuous ambiguity. Given an instance of ambiguous text, and a set of judgments as to the correct interpretation, the *certainty* of an interpretation is the percentage of readers who assign that interpretation to the text. For example, in Table 1 below (sec. 3.1), the certainty of the two interpretations, HA and LA of expression (a) are 12/17=71% and 1/17=5.9% respectively. Here, an expression shows *nocuous ambiguity* if none of the possible interpretations have a certainty exceeding the chosen threshold. Later in this section, we will describe further experiments with alternative, finer grained approaches to setting and measuring thresholds, that affect the classifier's behaviour.

Heuristics. Heuristics capture factors that may favour specific interpretations. Each heuristic embodies a hypothesis, drawn from the literature, about a linguistic phenomenon signifying a preferred reading. Some use statistical information (e.g., word distribution information obtained from a generic corpus, the BNC [1], using the Sketch Engine[2]). Others flag the presence of surface features in the text, or draw on semantic or world knowledge extracted from linguistic resources like WordNet[3] or VerbNet[4].

Machine learning (ML). Individual heuristics have limited predictive power: their effectiveness lies in their ability to operate in concert. Importantly, the information they encapsulate may be interdependent. We harness this by using ML techniques to combine the outputs of individual heuristics. ML is an established method for recognizing complex patterns automatically, making intelligent decisions based on empirical data, and learning of complex and nonlinear relations between data points. Our model uses supervised learning ML techniques, deducing a function from training data, to classify instances of ambiguity into nocuous or innocuous cases. The classifier training data consists of pairs of input objects (i.e. vectors made up of heuristics scores) and desired outputs (i.e. the class labels determined by the distribution of human judgments as captured by thresholds). To select an appropriate ML algorithm for the nocuity classifier, we tested our datasets (described in later sections) on several algorithms in the WEKA[5] package (e.g., decision tree, J48, Naive Bayes, SVM, Logistic Regression, LogitBoost, etc.)

To train, and validate, a nocuity classifier for a particular form of ambiguity, we build a dataset of judgments, and select heuristics that model

[1] http://www.natcorp.ox.ac.uk/

[2] http://sketchengine.co.uk/

[3] http://wordnet.princeton.edu/

[4] http://verbs.colorado.edu/~mpalmer/projects/verbnet.html

[5] http://www.cs.waikato.ac.nz/~ml/index.html

the information underlying the human judgements about a preferred interpretation.

We validated the approach on two forms of ambiguity. Sections 3 and 4 discuss how the methodology is applied to forms of coordination and anaphoric ambiguity, and evaluate the performance of the final classifiers.

3 Automatic Identification of Nocuous Coordination Ambiguity

Our previous work on nocuous ambiguity has focused on coordination ambiguity: a common kind of structural ambiguity. A coordination structure connects two words, phrases, or clauses together via a coordination conjunction (e.g., '*and*', '*or*', etc) as in the following examples:

(1) *They support a typing system for architectural components and connectors.*

(2) *It might be rejected or flagged for further processing.*

In (1), the coordination construction '*architectural components and connectors*' consists of a **near conjunct** (NC) (i.e. '*components*'), a **far conjunct** (FC) (i.e. '*connectors*'), and the attached **modifier** (M) (i.e. '*architectural*'). This construction allows two bracketings corresponding to high modifier attachment (*[architectural [components and connectors]]*) or low modifier attachement (*[[architectural components] and connector]*). Our aim is to refine Chantree et al (2006) and Willis et al (2008), hence our focus is on the two phenomena they treated: modification in noun phrase coordination (as in (1)) and in verb phrase coordination (as in (2)).

We implemented the heuristics described in the earlier work, and introduced two further ones (local document collocation frequency, and semantic similarity). We used the Chantree et al (2006) dataset of human judgments, but employed the LogitBoost algorithm for implementing the nocuity classifier (rather than the Logistic Regression equation). The following subsections give more detail.

3.1 Building a dataset

Coordination instances. Our dataset was collected and described by Chantree et al. (2006). It contains 138 coordination instances gathered from a set of requirement documents. Noun compound conjunctions account for the majority (85.5%) of cases (118 instances). Nearly half of these arose as a result of noun modifiers, while there are 36 cases with adjective and 18 with preposition modifiers.

Human judgment collection. The coordination instances containing potential ambiguity were presented to a group of 17 computing professionals including academic staff or research students. For each instance, the judges were asked to select one of three options: high modifier attachment (HA), low modifier attachment (LA), or ambiguous (A). Table 1 shows the judgment count for two sample instances. In instance (a) in table 1, the *certainty* of HA is 12/17=71%, and the *certainty* of LA is 1/17=6%. Instance (b) was judged mainly to be ambiguous.

	Judgments		
	HA	LA	A
(a) *security* and *privacy* requirements	12	1	4
(b) electrical *characteristics* and *interface*	4	4	9

Table 1. Judgment count for the sample instances (HA=high attachment; LA=low attachment; and A=Ambiguous)

We set an ambiguity threshold, τ, to determine whether the distribution of interpretations is nocuous or innocuous with respect to that particular τ. If the *certainty* of neither interpretation, HA or LA, exceeds the threshold τ, we say this is an instance of *nocuous* coordination. Otherwise it is *innocuous*. Here, (a) displays nocuous ambiguity for τ>71%.

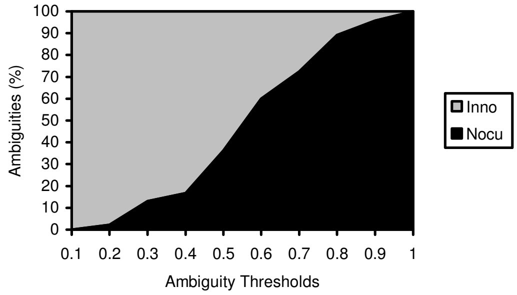


Figure 1. Proportions of interpretations at different ambiguity thresholds in the coordination instances

Figure 1 shows the systematic relationship between ambiguity threshold and the incidence of nocuous ambiguity in the dataset. Low thresholds can be satisfied with a very low certainty scores resulting in few instances being considered nocuous. At high thresholds, almost all instances are classified as nocuous unless the judges report a consensus interpretation.

3.2 Heuristics to predict Nocuity

Each heuristic tests a factor favouring a high or low modifier attachment (HA or LA). We implemented and extended Willis et al. (2008).

Coordination matching favours HA when the head words of near and far conjuncts are frequently found coordinated in a general corpus like BNC, suggesting they may form a single syntactic unit.

Distribution similarity measures how often two words are found in the same contexts. It favours HA where it detects a strong distributional similarity between the headwords of the two conjuncts, suggesting these form a syntactic unit (Kilgariff 2003).

Collocation frequency favours LA when the modifier is collocated much more frequently with the headword of the near conjunct than the far conjunct, in the document, or in the BNC.

Morphology favours HA when the conjunct headwords share a morphological marker (suffix) (Okumura and Muraki 1994).

Semantic similarity favours HA when the conjunct headwords display strong similarity in the taxonomic structure in WordNet[6].

3.3 Nocuity classification

To train, and test, the nocuity classifier, each ambiguity training/test instance is represented as an attribute-value vector, with the values set to the score of a particular heuristic. The class label of each instance (nocuous (Y) or innocuous (N) at a given ambiguity threshold) is determined by the *certainty* measure as discussed earlier. We selected the LogitBoost algorithm for building the classifier, because it outperformed other candidates on our training data than. To determine whether a test instance displays nocuity or not, we presented its feature vector to the classifier, and obtained a predicted class label (Y or N).

4 Automatic Identification of Nocuous Anaphora Ambiguity

An **anaphor** is an expression referring to an **antecedent**, usually a noun phrase (NP) found in the preceding text. A*naphora ambiguity* occurs when there are two or more candidate antecedents, as in example (3).

(3) *The procedure shall convert the 24 bit image to an 8 bit image, then display* ***it*** *in a dynamic window.*

In this case, both of the NPs, '*the 24 bit image*' and '*an 8 bit image*', are considered potential candidate antecedents of the anaphor '*it*'.

Anaphora ambiguity is difficult to handle due to contextual effects spread over several sentences. Our goal is to determine whether a case of anaphora ambiguity is nocuous or innocuous, automatically, by using our methodology.

4.1 The building of the Dataset

Anaphora instances. We collected 200 anaphora instances from requirements documents from RE@UTS website[7]. We are specifically concerned with 3rd person pronouns, which are widespread in requirements texts. The dataset contains different pronoun types. Nearly half the cases (48%) involve subject pronouns, although pronouns also occurred in objective and possessive positions (15% and 33%, respectively). Pronouns in prepositional phrases (e.g., '*under it')* are rarer (4% - only 8 instances).

Human judgment collection. The instances were presented to a group of 38 computing professionals (academic staff, research students, software developers). For each instance, the judges were asked to select the antecedent from the list of NP candidates. Each instance was judged by at least 13 people. Table 2 shows an example of judgment counts, where 12 out of 13 judges committed to '*supervisors*' as the antecedent of *'they'*, whereas 1 chose '*tasks*'.

1. Supervisors may only modify tasks **they** supervise to the agents they supervise.		
	Response Percent	Response Count
(a) supervisors	92.3%	12
(b) tasks	7.7%	1

Table 2. Judgment count for an anaphora ambiguity instance.

Ambiguity threshold. Given an anaphor, the interpretation *certainty* of a particular NP candidate is calculated as the percentage of the judgments for this NP against the total judgments for the instance. For example, consider the example in Table 2. The certainty of the NP '*supervisors*'

[6] Implemented by the NLP tool - Java WordNet Similarity Library. http://nlp.shef.ac.uk/result/software.html

[7] http://research.it.uts.edu.au/re/

is 12/13=92.3% and the certainty of the NP '*tasks*' is 1/13=7.7%. Thus, at an ambiguity threshold of, for instance, *τ* = 0.8, the ambiguity in Table 2 is *innocuous* because the agreement between the judges exceeds the threshold.

Figure 2 shows the relationship between ambiguity threshold and occurrence of nocuous ambiguity. As in Figure 1, the number of nocuous ambiguities increases with threshold τ. For high thresholds (e.g., τ≥0.9), more than 60% of instances are classified as nocuous. Below threshold (τ≤0.4), fewer than 8 cases are judged nocuous. Also, comparing Figures 1 and 2 would appear to suggest that, in technical documents, anaphora ambiguity is less likely to lead to misunderstandings than coordination.

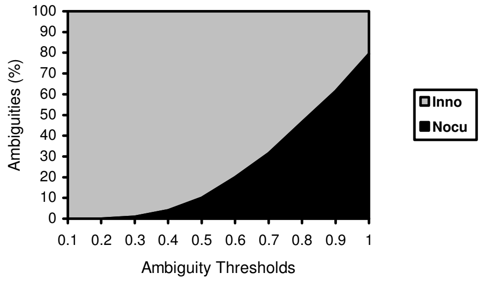


Figure 2. Proportions of interpretations at different ambiguity thresholds in the anaphora instances.

4.2 Antecedent Preference Heuristics

Drawing on the literature on anaphoric reference, we developed 12 heuristics of three types: related to *linguistic properties* of text components, to *context and discourse* information, or to *statistical* information drawn from standard corpora. Yang et al. (2010b) gives more detail. A heuristic marks candidate antecedents which it favours, or disfavours. For instance, heuristics favour definite NPs as antecedents, candidate NPs which agree in number and syntactic role with the anaphor, and those which share a syntactic collocation pattern in the text. They also favour those which respect the semantic constraints (e.g., animacy) propagated from subcategorisation information, and reward proximity to the anaphor. They disfavour candidate antecedents that occur in prepositional phrases, and those occupying a syntactic role distinct from the anaphor. Note: not all NPs are marked by all heuristics, and some heuristics are interdependent.

4.3 Nocuous Ambiguity Identification

Unlike coordination ambiguity, where judges chose for high or low modifier attachment, anaphora have scope over a variable set of potential antecedents, depending on each particular instance. To accommodate this, we developed an *antecedent* classifier which assigns a weighted antecedent tag to each NP candidate associated with an instance. Tag information is used subsequently to predict the whether the instance displays nocuous ambiguity.

The antecedent classifier is built using the Naive Bayes algorithm within the WEKA package and is trained to return three classes of candidate antecedent: *positive (Y)*, *questionable (Q)*, or *negative (N)*. In an *innocuous* case, a candidate NP will be classed as *Y* if its interpretation certainty exceeds the threshold set by τ, and tagged as *N* otherwise; in a *nocuous* case, it will be classed as *N* if its certainty is 0%, and classified as *Q* otherwise.

1. The LPS operational scenarios represent sequences of activities performed by operations personnel as **they** relate to the LPS software.		
	Response	Label
(a) the LPS operational scenarios	33.3%	Q
(b) sequences of activities	66.7%	Q
(c) activities	0%	N
(d) operations personnel	0%	N

Table 3. The determination of antecedent label for the NP candidates in a NOCUOUS ambiguity case (τ =0.8)

2. Testing performed to demonstrate to the acquirer that a CSCI system meets **its** specified requirements.		
	Response Percent	Class Label
(a) Testing	0%	N
(b) the acquirer	16.7%	N
(c) a CSCI system	83.3%	Y

Table 4. The determination of antecedent label for the NP candidates in a INNOCUOUS ambiguity case (τ =0.8)

	Antecedent Class Label		
	Y	Q	N
τ = 0.5	181	54	623
τ = 0.6	160	99	599
τ = 0.7	137	149	572
τ = 0.8	107	209	542
τ = 0.9	77	261	520
τ = 1.0	41	314	503

Table 5. The distribution of three antecedent class label at different ambiguity thresholds

Table 3 and 4 illustrate antecedent labels for NP antecedent candidates in a nocuous and innocuous case. Candidates (a) and (b) in Table 3 are labeled Q because their certainty falls below the threshold (τ = 0.8). For the same threshold, candidate (c) in Table 4 is tagged as *Y*. Table 5 shows the distribution of tags at certainty thresholds $\tau \geq 0.5$ for all (858) candidate antecedents in our sample.

Our intended application is a system to alert experts to risk of misunderstandings. This suggests we should emphasise recall even at the expense of some precision (Berry et al. 2003). We developed two versions of the algorithm that determines whether an instance is nocuous or not, depending on the contribution made by its antecedent candidates tagged *Y*. We relax constraints by introducing two concepts: a weak positive threshold W_Y and a weak negative threshold W_N set at 0.5 and 0.4, respectively[8]. The rationale for weak thresholds is that antecedent preference reflects a spectrum with *Y* (high), *Q* (medium), and *N* (low). Weak positive and negative thresholds act as buffers to the *Q* area. Antecedent NPs that fall in the W_Y or W_N buffer area are treated as possible false negative (FN) for the classification of the label *Q*. An antecedent tag *Y*/*N* is labeled as weak positive or negative depending on these thresholds. The algorithm for identifying nocuous ambiguity is given in Figure 3. It treats as innocuous those cases where the antecedent label list contains one clear *Y* candidate, whose certainty exceeds all others by a margin.

Given an anaphora ambiguity instance with multiple potential NPs, the antecedent classifier returns a label list, $R = \{r_1, r_2, \ldots, r_n\}$, for individual NPs.

Parameters:

1) W_Y - the threshold for the *weak positive* label. The label *Y* is viewed as *weak positive* when the positive prediction score $r_i < W_Y$

2) W_N - the threshold for the *weak negative* label. The label *N* is viewed as *weak negative* when the negative prediction score $r_i < W_N$

Procedure:

```
if the label list R contains
        (one Y, no Q, one or more N )
   or
        (no Y, one Q, one or more N but not weak negative )
   or
       (one Y but not weak positive, any number of Q or N)
then
        the ambiguity is INNOCUOUS
else
        the ambiguity is NOCUOUS
```

Figure 3. The algorithm for nocuous ambiguity identification

5 Experiments and Results

In all experiments, the performance was evaluated using 5-fold cross-validation, using standard measures of Precision (P), Recall (R), F-measure (F), and Accuracy. We use two naive baselines: BL-1 assumes that all ambiguity instances are *innocuous*; BL-2 assumes that they are all *nocuous*. For fair comparison against the baselines, for both forms of ambiguity, we only report the performance of our ML-based models when the incidence of nocuous ambiguities falls between 10% ~ 90% of the set (see Figures 1 and 2). We first report our findings for the identification of nocuous coordination ambiguities and then discuss the effectiveness of our model in distinguishing possible nocuous ambiguities from a set of ambiguity instances.

5.1 Nocuous Coordination Ambiguity Identification

Willis et al (2008) demonstrated the ability of their approach to adapt to different thresholds by plotting results against the two naïve base lines. Since we extended and refined their approach described we plot our experimental results (CM-1), for comparison, using the same measures, against their evaluation data (CM-2), in Figure 4.

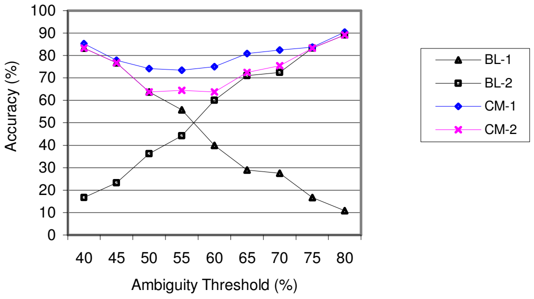


Figure 4. The performance comparison of the ML-based models, CM-1 and CM-2, to the two baseline models, BL-1 and BL-2, in nocuous coordination ambiguity identification.

Our CM-1 model performed well with an accuracy of above 75% on average at all ambiguity threshold levels. As expected, at very high and very low thresholds, we did not improve on the naive baselines (which have perfect recall and hence high accuracy). The CM-1 model displayed its advantage when the ambiguity threshold fell in the range between 0.45 and 0.75 (a significantly wider range than reported for CM-2 Willis et al (2008)). CM-1 maximum improvement was achieved around the 58% crossover point where the two naïve baselines intersect and our model achieved around 21% increased accu-

[8] Weak positive and negative thresholds are set experimentally.

racy. This suggests that the combined heuristics do have strong capability of distinguishing nocuous from innocuous ambiguity at the weakest region of the baseline models.

Figure 4 also shows that, the CM-1 model benefitted from the extended heuristics and the LogitBoost algorithm with an increased accuracy of around 5.54% on average compared with CM-2. This suggests that local context information and semantic relationships between coordinating conjuncts provide useful clues for the identification of nocuous ambiguity. Furthermore, the LogitBoost algorithm is more suitable for dealing with a numeric-attribute feature vector than the previous Logistic Regression algorithm.

5.2 Nocuous Anaphora Ambiguity Identification

We report on two implementations: one with weak thresholds (AM-1) and one without (AM-2). We compare both approaches using the baselines, BL-1 and BL-2 (in Figure 5). It shows that AM-1 and AM-2 achieve consistent improvements on baseline accuracy at high thresholds (τ≥0.75). Here also, the improvement maximises around the 83% threshold point where the two baselines intersect. However, the ML-based models perform worse than BL-1 at the lower thresholds (0.5≤τ≤0.7). One possible explanation is that, at low thresholds, performance is affected by lack of data for training of the *Q* class label, an important indicator for nocuous ambiguity (see Table 5). This is also consistent with the ML models performing well at higher thresholds, when enough nocuous instances are available for training.

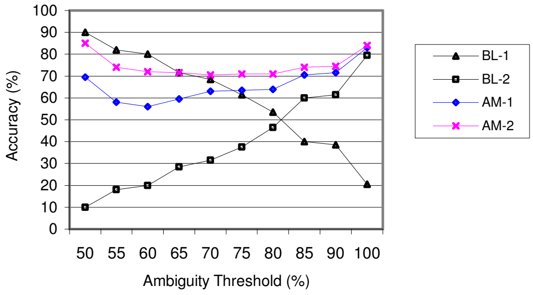


Figure 5. The performance comparison of the ML-based models, AM-1 and AM-2, to the two baseline models, BL-1 and BL-2, in nocuous anaphora ambiguity identification.

Figure 5 further shows that the model with weak thresholds (AM-1) did not perform as well as the model without weak thresholds (AM-2) on accuracy. Although both models perform much better than the baselines on precision (more experimental results are reported in Yang et al. (2010b)), the actual precisions for both models are relatively low, ranging from 0.3 ~ 0.6 at different thresholds. When the AM-1 model attempts to discover more nocuous instances using weak thresholds, it also introduces more false positives (innocuous instances incorrectly classed as nocuous). The side-effect of introducing false positives for AM-1 is to lower accuracy. However, the AM-1 model outperforms both AM-2 and BL-2 models on F-measure (Figure 6), with an average increase of 5.2 and 3.4 percentage points respectively. This reveals that relaxing sensitivity to the ambiguity threshold helps catch more instances of nocuous anaphora ambiguity.

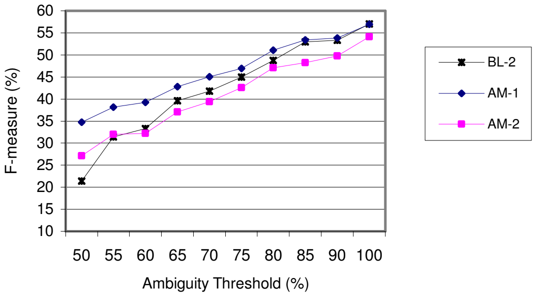


Figure 6. The performance comparison of the ML-based models, AM-1 and AM-2, to the baseline model BL-2 (naïve nocuous)

6 Discussions

We presented judges with sentences containing ambiguities without any surrounding context, even though contextual information (e.g., discourse focus) clearly contributes to interpretation. This is a weakness in our data collection technique. Besides contextual information, van Deemter's Principle of Idiosyncratic Interpretation (1998) suggests that some factors, including the reader's degree of language competence, can affect perceptions of ambiguity. Similarly, familiarity with a domain, including tacit specialist information (Polanyi, 1966), and the extent to which this is shared by a group, will have an effect on the extent to which stakeholders arrive at diverging interpretations.

In our case, we extracted instances from requirements documents covering several technical domains. Judgements are sensitive to the backgrounds of the participants, and the extent to which stakeholder groups share such a background. Also, we used several large, generic NL resources, including the BNC and WordNet. The performance of several heuristics would change if they drew on domain specific resources. Different interpretations may be compatible, and so not necessarily contribute to misunderstanding.

Finally, we used different machine learning algorithms to tackle different types of ambiguity instances: LogitBoost for coordination ambiguity and Naive Bayes for anaphora ambiguity. The main reason is that coordination heuristics returned numeric values, whereas the anaphora heuristics were Boolean. Our method assumes tailoring of the ML algorithm to the choice of heuristic. These limitations indicate that the methodology has a high degree of flexibility, but also that it has several interdependent components and background assumptions that have to be managed if an application is to be developed.

7 Related Work

Many researchers have remarked on the fact that some ambiguities are more likely than others to lead to misunderstandings, and suggested classifying them accordingly. Poesio (1996) discussed cases where multiple readings are intended to coexist, and distinguished between language inherent and human disambiguation factors from a philosophical perspective. His notion of '*perceived ambiguity*' suggests that human perceptions are what actually cause an ambiguity to be misunderstood. Van Deemter's (2004) '*vicious ambiguity*' refers to an ambiguity that has no single, strongly preferred interpretation. He proposed quantifying 'viciousness' using probabilities taken from corpus data. Van Rooy (2004) defined a notion of '*true ambiguity*': a sentence is truly ambiguous only if there are at least two interpretations that are optimally relevant. These last two approaches rely on probability analysis of language usage, and not directly on human perception, which we believe to be the key to evaluating ambiguity. Our work differs in that it takes into account the distribution of interpretations arrived at by a group of human judges engaged with a text. Our model treats ambiguity not as a property of a linguistic construct or a text, or a relation between a text and the perceptions of a single reader, but seeks to understand the mechanisms that lead to misunderstandings between people in a group or process.

Poesio *et al* (2006) have pointed out that disambiguation is not always necessary; for instance, in some complex anaphora cases, the final interpretation may not be fully specified, but only 'good enough'. Our work does not attempt disambiguation. It seeks to highlight the risk of multiple interpretations (whatever those are).

8 Conclusions and Future Work

We have presented a general methodology for automatically identifying nocuous ambiguity (i.e. cases of ambiguity where there is a risk that people will hold different interpretations) relative to some tolerance level set for such a risk. The methodology has been implemented in a ML based architecture, which combines a number of heuristics each highlighting factors which may affect how humans interpret ambiguous constructs. We have validated the methodology by identifying instances of nocuous ambiguity in coordination and anaphoric constructs. Human judgments were collected in a dataset used for training the ML algorithm and evaluation. Results are encouraging, showing an improvement of approximately 21% on accuracy for coordination ambiguity and about 3.4% on F-measure for anaphora ambiguity compared with naive baselines at different ambiguity threshold levels. We showed, by comparison with results reported in Willis et al (2008) that the methodology can be fine tuned, and extended to other ambiguity types, by including different heuristics.

Our method can highlight the risk of different interpretations arising: this is not a task a single human could perform, as readers typically have access only to their own interpretation and are not routinely aware that others hold a different one. Nonetheless, our approach has limitations, particularly around data collection, and for anaphora ambiguity at low thresholds. We envisage further work on the implementation of ambiguity tolerance thresholds

Several interesting issues remain to be investigated to improve our system's performance and validate its use in practice. We need to explore how to include different and complex ambiguity types (e.g., PP attachment and quantifier scoping), and investigate whether these are equally amenable to a heuristics based approach.

Acknowledgement

This work is supported financially by UK EPSRC for the MaTREx project (EP/F068859/1), and Irish SFI for the grant 03/CE2/I303_1.

References

Daniel M. Berry, Erik Kamsties, and Michael M. Krieger. 2003. From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity. Technical Report, School of Computer Science, University of Waterloo.

Stephen Boyd, Didar Zowghi, and Alia Farroukh. 2005. Measuring the Expressiveness of a Constrained Natural Language: An Empirical Study. In *Proceedings of the 13th IEEE International Conference on Requirements Engineering (RE'05)*, Washington, DC, pages 339-52.

Eric Brill and Philip Resnik. 1994. A Rule-Based Approach to Prepositional Phrase Attachment Disambiguation. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 1198-204.

Francis Chantree, Bashar Nuseibeh, Anne de Roeck, and Alistair Willis. 2006. Identifying Nocuous Ambiguities in Natural Language Requirements. In *Proceedings of 14th IEEE International Requirements Engineering Conference (RE'06)*, Minneapolis, USA, pages 59-68.

Adam Kilgarriff. 2003. Thesauruses for Natural Language Processing. In *Proceedings of NLP-KE*, pages 5-13.

Preslav Nakov and Marti Hearst. 2005. Using the Web as an Implicit Training Set: Application to Structural Ambiguity Resolution. In *Proceedings of HLT-NAACL'05*, pages 835-42.

Akitoshi Okumura and Kazunori Muraki. 1994. Symmetric Pattern Matching Analysis for English Coordinate Structures. In *Proceedings of the 4th Conference on Applied Natural Language Processing*, pages 41-46.

Massimo Poesio. 1996. Semantic Ambiguity and Perceived Ambiguity In *Semantic Ambiguity and Underspecification* edited by K. van Deemter and S. Peters, pages 159-201.

Massimo Poesio and Ron Artstein. 2008. Introduction to the Special Issue on Ambiguity and Semantic Judgements. *Research on Language & Computation* 6: 241-45.

Massimo Poesio, Patick Sturt, Ron Artstein, and Ruth Filik. 2006. Underspecification and Anaphora: Theoretical Issues and Preliminary Evidence. *Discourse Processes* 42(2): 157-75.

Michael Polanyi. 1966. *The Tacit Dimension*. RKP, London.

Kees van Deemter. 1998. Ambiguity and Idiosyncratic Interpretation. *Journal of Semantics* 15(1): 5-36.

Kees van Deemter. 2004. Towards a Probabilistic Version of Bidirectional Ot Syntax and Semantics. *Journal of Semantics* 21(3): 251-80.

Robert van Rooy. 2004. Relevance and Bidirectional Ot. In *Optimality Theory and Pragmatic*, edited by R. Blutner and H. Zeevat, pages 173-210.

Thomas Wasow, Amy Perfors, and David Beaver. 2003. The Puzzle of Ambiguity. In *Morphology and the Web of Grammar: Essays in Menory of Steven G. Lapointe*, edited by O. Orgun and P. Sells.

Alistair Willis, Francis Chantree, and Anne De Roeck. 2008. Automatic Identification of Nocuous Ambiguity. *Research on Language & Computation* 6(3-4): 1-23.

Hui Yang, Alistair Willis, Anne de Roeck, and Bashar Nuseibeh. 2010a. Automatic Detection of Nocuous Coordination Ambiguities in Natural Language Requirements. *In Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering Conference (ASE'10).* (In press)

Hui Yang, Anne de Roeck, Alistair Willis, and Bashar Nuseibeh. 2010b. Extending Nocuous Ambiguity Analysis for Anaphora in Natural Language Requirements. *In Proceedings of the 18th International Requirements Engineering Conference (RE'10).* (In press)

Contextual Modeling for Meeting Translation Using Unsupervised Word Sense Disambiguation

Yang Mei
Department of Electrical Engineering
University of Washington
yangmei@u.washington.edu

Katrin Kirchhoff
Department of Electrical Engineering
University of Washington
katrin@ee.washington.edu

Abstract

In this paper we investigate the challenges of applying statistical machine translation to meeting conversations, with a particular view towards analyzing the importance of modeling contextual factors such as the larger discourse context and topic/domain information on translation performance. We describe the collection of a small corpus of parallel meeting data, the development of a statistical machine translation system in the absence of genre-matched training data, and we present a quantitative analysis of translation errors resulting from the lack of contextual modeling inherent in standard statistical machine translation systems. Finally, we demonstrate how the largest source of translation errors (lack of topic/domain knowledge) can be addressed by applying document-level, *unsupervised* word sense disambiguation, resulting in performance improvements over the baseline system.

1 Introduction

Although statistical machine translation (SMT) has made great progress over the last decade, most SMT research has focused on the translation of structured input data, such as newswire text or parliamentary proceedings. Spoken language translation has mostly concentrated on two-person dialogues, such as travel expressions or patient-provider interactions in the medical domain. Recently, more advanced spoken-language data has been addressed, such as speeches (Stüker et al., 2007), lectures (Waibel and Fügen, 2008), and broadcast conversations (Zheng et al., 2008). Problems for machine translation in these genres include the nature of spontaneous speech input (e.g. disfluencies, incomplete sentences, etc.) and the lack of high-quality training data. Data that match the desired type of spoken-language interaction in topic, domain, and, most importantly, in style, can only be obtained by transcribing and translating conversations, which is a costly and time-consuming process. Finally, many spoken-language interactions, especially those involving more than two speakers, rely heavily on the participants' shared contextual knowledge about the domain and topic of the discourse, relationships between speakers, objects in the real-world environment, past interactions, etc. These are typically not modelled in standard SMT systems.

The problem of speech disfluencies has been addressed by disfluency removal techniques that are applied prior to translation (Rao et al., 2007; Wang et al., 2010). Training data sparsity has been addressed by adding data from out-of-domain resources (e.g. (Matusov et al., 2004; Hildebrandt et al., 2005; Wu et al., 2008)), exploiting comparable rather than parallel corpora (Munteanu and Marcu, 2005), or paraphrasing techniques (Callison-Burch et al., 2006). The lack of contextual modeling, by contrast, has so far not been investigated in depth, although it is a generally recognized problem in machine translation. Early attempts at modeling contextual information in machine translation include (Mima et al., 1998), where information about the role, rank and gender of speakers and listeners was utilized in a transfer-based spoken-language translation system for travel dialogs. In (Kumar et al., 2008)

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1227–1235,
Beijing, August 2010

statistically predicted dialog acts were used in a phrase-based SMT system for three different dialog tasks and were shown to improve performance. Recently, contextual source-language features have been incorporated into translation models to predict translation phrases for traveling domain tasks (Stroppa et al., 2007; Haque et al., 2009). However, we are not aware of any work addressing contextual modeling for statistical translation of spoken meeting-style interactions, not least due to the lack of a relevant corpus.

The first goal of this study is to provide a quantitative analysis of the impact of the lack of contextual modeling on translation performance. To this end we have collected a small corpus of parallel multi-party meeting data. A baseline SMT system was trained for this corpus from freely available data resources, and contextual translation errors were manually analyzed with respect to the type of knowledge sources required to resolve them. Our analysis shows that the largest error category consists of word sense disambiguation errors resulting from a lack of topic/domain modeling. In the second part of this study we therefore present a statistical way of incorporating such knowledge by using a graph-based unsupervised word sense disambiguation algorithm at a *global* (i.e. document) level. Our evaluation on real-world meeting data shows that this technique improves the translation performance slightly but consistently with respect to position-independent word error rate (PER).

2 Data

2.1 Parallel Conversational Data

For our investigations we used a subset of the AMI corpus (McCowan, 2005), which is a collection of multi-party meetings consisting of approximately 100 hours of multimodal data (audio and video recordings, slide images, data captured from digital whiteboards, etc.) with a variety of existing annotations (audio transcriptions, topic segmentations, summaries, etc.). Meetings were recorded in English and fall into two broad types: scenario meetings, where participants were asked to act out roles in a pre-defined scenario, and non-scenario meetings where participants were not restricted by role assignments. In the first case, the scenario was a project meeting about the development of a new TV remote control; participant roles were project manager, industrial designer, marketing expert, etc. The non-scenario meetings are about the move of an academic lab to a new location on campus. The number of participants is four. For our study we selected 10 meetings (5 scenario meetings and 5 non-scenario meetings) and had their audio transcriptions translated into German (our chosen target language) by two native speakers each. Translators were able to simultaneously read the audio transcription of the meeting, view the video, and listen to the audio, when creating the translation. The translation guidelines were designed to obtain translations that match the source text as closely as possible in terms of style – for example, translators were asked to maintain the same level of colloquial as opposed to formal language, and to generally ensure that the translation was pragmatically adequate. Obvious errors in the source text (e.g. errors made by non-native English speakers among the meeting participants) were not rendered by equivalent errors in the German translation but were corrected prior to translation. The final translations were reviewed for accuracy and the data were filtered semi-automatically by eliminating incomplete sentences, false starts, fillers, repetitions, etc. Although these would certainly pose problems in a real-world application of spoken language translation, the goal of this study is not to analyze the impact of speech-specific phenomena on translation performance (which, as discussed in Section 1, has been addressed before) but to assess the impact of contextual information such as discourse and knowledge of the real-world surroundings. Finally, single-word utterances such as *yeah, oh, no, sure*, etc. were downsampled since they are trivial to translate and were very frequent in the corpus; their inclusion would therefore bias the development and tuning of the MT system towards these short utterances at the expense of longer, more informative utterances.

Table 1 shows the word counts of the translated meetings after the preprocessing steps described above. As an indicator of inter-translator

ID	type	# utter.	# word	S-BLEU
ES2008a	S	224	2327	21.5
IB4001	NS	419	3879	24.5
IB4002	NS	447	3246	30.5
IB4003	NS	476	5118	24.1
IB4004	NS	593	5696	26.9
IB4005	NS	381	4719	30.4
IS1008a	S	191	2058	25.8
IS1008b	S	353	3661	24.1
IS1008c	S	308	3351	19.6
TS3005a	S	245	2339	28.1

Table 1: Sizes and symmetric BLEU scores for translated meetings from the AMI corpus (S = scenario meeting, NS = non-scenario meeting).

agreement we computed the symmetric BLEU (S-BLEU) scores on the reference translations (i.e. using one translation as the reference and the other as the hypothesis, then switching them and averaging the results). As we can see, scores are fairly low overall, indicating large variation in the translations. This is due to (a) the nature of conversational speech, and (b) the linguistic properties of the target language. Conversational data contain a fair amount of colloquialisms, referential expressions, etc. that can be translated in a variety of ways. Additionally, German as the target language permits many variations in word order that convey slight differences in emphasis, which is turn is dependent on the translators' interpretation of the source sentence. German also has rich inflectional morphology that varies along with the choice of words and word order (e.g. verbal morphology depends on which subject is chosen).

2.2 SMT System Training Data

Since transcription and translation of multi-party spoken conversations is extremely time-consuming and costly, it is unlikely that parallel conversational data will ever be produced on a sufficiently large scale for a variety of different meeting types, topics, and target languages. In order to mimic this situation we trained an initial English-German SMT system on freely available out-of-domain data resources. We considered the following parallel corpora: news text (de-news[1], 1.5M words), EU parliamentary proceedings (Europarl (Koehn, 2005), 24M words) and EU legal documents (JRC Acquis[2], 35M words), as well as two generic English-German machine-readable dictionaries[3],[4] (672k and 140k entries, respectively).

3 Translation Systems

We trained a standard statistical phrase-based English-German translation system from the resources described above using Moses (Hoang and Koehn, 2008). Individual language models were trained for each data source and were then linearly interpolated with weights optimized on the development set. Similarly, individual phrase tables were trained and were then combined into a single table. Binary indicator features were added for each phrase pair, indicating which data source it was extracted from. Duplicated phrase pairs were merged into a single entry by averaging their scores (geometric mean) over all duplicated entries. The weights for binary indicator features were optimized along with all other standard features on the development set. Our previous experience showed that this method worked better than the two built-in features in Moses for handling multiple translation tables. We found that the JRC corpus obtained very small weights; it was therefore omitted from further system development. Table 2 reports results from six different systems: the first (System 1) is a system that only uses the parallel corpora but not the external dictionaries listed in Section 2.2. System 2 additionally uses the external dictionaries. All systems use two meetings (IB4002 and IS1008b) as a development set for tuning model parameters and five meetings for testing (IB4003-5,IS1008c,TS3005a). For comparison we also trained a version of the system where a small in-domain data set (meetings ES2008a, IB4001, and IS1008a) was added to the training data (System 3). Finally, we also compared our performance against Google Translate, which is a state-of-the-art statistical MT system with unconstrained ac-

[1] www.iccs.inf.ed.ac.uk/~pkoehn/publications/de-news

[2] http://wt.jrc.it/lt/Acquis/

[3] http://www.dict.cc

[4] http://www-user.tu-chemnitz.de/~fri/ding

	System description	Dev set				Eval set			
		OOV (%)		Trans. Scores		OOV (%)		Trans. Scores	
		EN	DE	BLEU	PER	EN	DE	BLEU	PER
System 1	OOD parallel data only	4.1	17.0	23.8	49.0	6.5	20.5	21.1	49.5
System 2	System 1 + dictionaries	1.5	15.9	24.6	47.3	2.8	16.3	21.7	48.4
System 3	System 1 + ID parallel data	3.5	13.4	24.7	47.2	5.8	19.7	21.9	48.3
System 4	System 2 + ID parallel data	1.2	12.9	25.4	46.1	2.5	15.9	22.0	48.2
System 5	System 4 + web data	1.2	12.8	26.0	45.9	2.5	15.8	22.1	48.1
System 6	Google Translate	–	–	25.1	49.1	–	–	23.7	50.8

Table 2: System performance using out-of-domain (OOD) parallel data only vs. combination with a small amount of in-domain (ID) data and generic dictionaries. For each of the development (DEV) and evaluation (Eval) set, the table displays the percentages of unknown word types (OOV) for English (EN) and German (DE), as well as the translation scores of BLEU (%) and PER.

cess to the web as training data (System 6). As expected, translation performance is fairly poor compared to the performance generally obtained on more structured genres. The use of external dictionaries helps primarily in reducing PER scores while BLEU scores are only improved noticeably by adding in-domain data. System 6 shows a more even performance across dev and eval sets than our trained system, which may reflect some degree of overtuning of our systems to the relatively small development set (about 7K words). However, the PER scores of System 6 are significantly worse compared to our in-house systems.

In order to assess the impact of adding web data specifically collected to match our meeting corpus we queried a web portal[5] that searches a range of English-German bilingual web resources and returns parallel text in response to queries in either English or German. As queries we used English phrases from our development and evaluation sets that (a) did not already have phrasal translations in our phrase tables, (b) had a minimum length of four words, and (c) occurred at least twice in the test data. In those cases where the search engine returned results with an exact match on the English side, we word-aligned the resulting parallel text (about 600k words) by training the word alignment together with the news text corpus. We then extracted new phrase pairs (about 3k) from the aligned data. The phrasal scores assigned to the new phrase pairs were set to 1; the lexical scores were computed from a word lexicon trained over both the baseline data resources and the parallel web data. However, results (Row 5 in Table 2) show that performance hardly improved, indicating the difficulty in finding matching data sources for conversational speech.

Table 2 also shows the impact of different data resources on the percentages of unknown word types (OOV) for both the source and target languages. The use of external dictionaries gave the largest reduction of OOV rates (System 1 vs. System 2 and System 3 vs. System 4), followed by the use of in-domain data (System 1 vs. System 3 and System 2 vs. System 4). Since they were retrieved by multi-word query phrases, adding the web data did not lead to significant reduction on the OOV rates (System 4 vs. System 5).

Finally, we also explored a hierarchical phrase-based system as an alternative baseline system. The system was trained using the Joshua toolkit (Li et al., 2009) with the same word alignments and language models as were used in the standard phrase-based baseline system (System 4). After extracting the phrasal (rule) tables for each data source, they were combined into a single phrasal (rule) table using the same combination approach as for the basic phrase-based system. However, the translation results (BLEU/PER of 24.0/46.6 (dev) and 20.8/47.6 (eval), respectively) did not show any improvement over the basic phrase-based system.

[5] http://www.linguee.com

4 Analysis of Baseline Translations: Effect of Contextual Information

The output from System 5 was analyzed manually in order to assess the importance of modeling contextual information. Our goal was not to determine how translation of meeting style data can be improved in general – better translations could certainly be generated by better syntactic modeling, addressing morphological variation in German, and generally improving phrasal coverage, in particular for sentences involving colloquial expressions. However, these are fairly general problems of SMT that have been studied previously. Instead, our goal was to determine the relative importance of modeling different contextual factors, such as discourse-level information or knowledge of the real-world environment, which have not been studied extensively.

We considered three types of contextual information: discourse coherence information (in particular anaphoric relations), knowledge of the topic or domain, and real-world/multimodal information. Anaphoric relations affect the translation of referring expressions in cases where the source and target languages make different grammatical distinctions. For example, German makes more morphological distinctions in noun phrases than English. In order to correctly translate an expression like "the red one" the grammatical features of the target language expression for the referent need to be known. This is only possible if a sufficiently large context is taken into account during translation and if the reference is resolved correctly. Knowledge of the topic or domain is relevant for correctly translating content words and is closely related to the problem of word sense disambiguation. In our current setup, topic/domain knowledge could be particularly helpful because in-domain training data is lacking and many word translations are obtained from generic dictionaries that do not assign probabilities to competing translations. Finally, knowledge of the real-world environment, such as objects in the room, other speakers present, etc. determines translation choices. If a speaker utters the expression "that one" while pointing to an object, the correct translation might depend on the grammatical features of the linguistic expression for that object; e.g. in German, the translation could be "die da", "der da" or "das da". Since the participants in our meeting corpus use slides and supporting documents we expect to see some effect of such exophoric references to external objects.

Error type	% (dev)	% (eval)
Word sense	64.5	68.2
Exophora (addressee)	24.3	23.4
Anaphora	10.2	7.8
Exophora (other)	1.0	0.6

Table 3: Relative frequency of different error types involving contextual knowledge. The total number of errors is 715, for 315 sentences.

In order to quantify the influence of contextual information we manually analyzed the 1-best output of System 5, identified those translation errors that require knowledge of the topic/domain, larger discourse, or external environment for their resolution, classified them into different categories, and computed their relative frequencies. We then corrected these errors in the translation output to match at least one of the human references, in order to assess the maximum possible improvement in standard performance scores that could be obtained from contextual modeling. The results are shown in Tables 3 and 4. We observe that out of all errors that can be related to the lack of contextual knowledge, word sense confusions are by far the most frequent. A smaller percentage of errors is caused by anaphoric expressions. Contrary to our expectations, we did not find a strong impact of exophoric references; however, there is one crucial exception where real-world knowledge does play an important role. This is the correct translation of the addressee *you*. In English, this form is used for the second person singular, second person plural, and the generic interpretation (as in "one", or "people"). German has three distinct forms for these cases and, additionally, formal and informal versions of the second-person pronouns. The required formal/informal pronouns can only be determined by prior knowledge of the relationships among the meeting participants. However, the singular-plural-generic distinction can potentially be resolved by multimodal informa-

	Original		Corrected	
	BLEU (%)	PER	BLEU (%)	PER
dev	26.0	45.9	27.5	44.0
eval	22.1	48.1	23.3	46.0

Table 4: Scores obtained by correcting errors due to lack of contextual knowledge.

tion such as gaze, head turns, body movements, or hand gestures of the current speaker. Since these errors affect mostly single words as opposed to larger phrases, the impact of the corrections on BLEU/PER scores is not large. However, for practical applications (e.g. information extraction or human browsing of meeting translations) the correct translation of content words and referring expressions would be very important. In the remainder of the paper we therefore describe initial experiments designed to address the most important source of contextual errors, viz. word sense confusions.

5 Resolving Word Sense Disambiguation Errors

The problem of word sense disambiguation (WSD) in MT has received a fair amount of attention before. Initial experiments designed at integrating a WSD component into an MT system (Carpuat and Wu, 2005) did not meet with success; however, WSD was subsequently demonstrated to be successful in data-matched conditions (Carpuat and Wu, 2007; Chan et al., 2007). The approach pursued by these latter approaches is to train a supervised word sense classifier on different phrase translation options provided by the phrase table of an initial baseline system (i.e. the task is to separate different phrase senses rather than word senses). The input features to the classifier consist of word features obtained from the immediate context of the phrase in questions, i.e. from the same sentence or from the two or three preceding sentences. The classifier is usually trained only for those phrases that are sufficiently frequent in the training data.

By contrast, our problem is quite different. First, many of the translation errors caused by choosing the wrong word sense relate to words obtained from an external dictionary that do not occur in the parallel training data; there is also little in-domain training data available in general. For these reasons, training a supervised WSD module is not an option without collecting additional data. Second, the relevant information for resolving a word sense distinction is often not located in the immediately surrounding context but it is either at a more distant location in the discourse, or it is part of the participants' background knowledge. For example, in many meetings the opening remarks refer to slides and an overhead projector. It is likely that subsequent mentioning of *slide* later on during the conversation also refer to overhead slides (rather than e.g. *slide* in the sense of “playground equipment”), though the contextual features that could be used to identify this word sense are not located in the immediately preceding sentences. Thus, in contrast to supervised, local phrase sense disambiguation employed in previous work, we propose to utilize unsupervised, *global* word sense disambiguation, in order to obtain better modeling of the topic and domain knowledge that is implicitly present in meeting conversations.

5.1 Unsupervised Word Sense Disambiguation

Unsupervised WSD algorithms have been proposed previously (e.g. (Navigli and Lapata, 2007; Cheng et al., 2009)). The general idea is to exploit measures of word similarity or relatedness to jointly tag all words in a text with their correct sense. We adopted the graph-based WSD method proposed in (Sinha and Mihalcea, 2007), which represents all word senses in a text as nodes in an undirected graph $G = (V, E)$. Pairs of nodes are linked by edges weighted by scores indicating the similarity or relatedness of the words associated with the nodes. Given such a graph, the likelihood of each node is derived by the PageRank algorithm (Brin and Page, 1998), which measures the relative importance of each node to the entire graph by considering the amount of “votes” the node receives from its neighboring nodes. The PageRank algorithm was originally designed for directed graphs, but can be easily extended to an undirected graph. Let $PR(v_i)$ denote the PageRank score of v_i. The PageRank algorithm iteratively updates this score as follows:

$$PR(v_i) = (1-d) + d \sum_{(v_i,v_j)\in E} PR(v_j)\frac{w_{ij}}{\sum_k w_{kj}}$$

where w_{ij} is the similarity weight of the undirected edge (v_i, v_j) and d is a damping factor, which is typically set to 0.85 (Brin and Page, 1998). The outcome of the PageRank algorithm is numerical weighting of each node in the graph. The sense with the highest score for each word identifies its most likely word sense. For our purposes, we modified the procedure as follows. Given a document (meeting transcription), we first identify all content words in the source document. The graph is then built over all target-language translation candidates, i.e. each node represents a word translation. Edges are then established between all pairs of nodes for which a word similarity measure can be obtained.

5.2 Word Similarity Measures

We follow (Zesch et al., 2008a) in computing the semantic similarity of German words by exploiting the Wikipedia and Wiktionary databases. We use the publicly available toolkits JWPL and JWKTL (Zesch et al., 2008b) to retrieve relevant articles in Wikipedia and entries in Wiktionary for each German word – these include the first paragraphs of Wikipedia articles entitled by the German word, the content of Wiktionary entries of the word itself as well as of closely related words (hypernyms, hyponyms, synonyms, etc.). We then concatenate all retrieved material for each word to construct a pseudo-gloss. We then lowercase and lemmatize the pseudo-glosses (using the lemmatizer available in the TextGrid package [6]), exclude function words by applying a simple stop-word list, and compute a word similarity measure for a given pair of words by counting the number of common words in their glosses.

We need to point out that one drawback in this approach is the low coverage of German content words in the Wikipedia and Wiktionary databases. Although the English edition contains millions of entries, the German edition of Wikipedia and Wiktionary is much smaller – the coverage of all content words in our task ranges between 53% and 56%, depending on the meeting, which leads to graphs with roughly 3K to 5K nodes and 8M to 13M edges. Words that are not covered mostly include rare words, technical terms, and compound words.

5.3 Experiments and Results

For each meeting, the derived PageRank scores were converted into a positive valued feature, referred to as the WSD feature, by normalization and exponentiation:

$$f_{WSD}(w_g|w_e) = \exp\left\{\frac{PR(w_g)}{\sum_{w_g\in H(w_e)} PR(w_g)}\right\}$$

where $PR(w_g)$ is the PageRank score for the German word w_g and $H(w_e)$ is the set of all translation candidates for the English word w_e. Since they are not modeled in the graph-based method, multi-words phrases and words that are not found in the Wikipedia or Wiktionary databases will receive the default value 1 for their WSD feature. The WSD feature was then integrated into the phrase table to perform translation. The new system was optimized as before.

It should be emphasized that the standard measures of BLEU and PER give an inadequate impression of translation quality, in particular because of the large variation among the reference translations, as discussed in Section 4. In many cases, better word sense disambiguation does not result in better BLEU scores (since higher gram matches are not affected) or even PER scores because although a feasible translation has been found it does not match any words in the reference translations. The best way of evaluating the effect of WSD is to obtain human judgments – however, since translation hypotheses change with every change to the system, our original error annotation described in Section 4 cannot be re-used, and time and resource constraints prevented us from using manual evaluations at every step during system development.

In order to loosen the restrictions imposed by having only two reference translations, we utilized a German thesaurus[7] to automatically extend the content words in the references with synonyms. This can be seen as an automated way of

[6] http://www.textgrid.de/en/beta.html

[7] http://www.openthesaurus.de

	No WSD			With WSD		
	BLEU (%)	PER	XPER	BLEU (%)	PER	XPER
dev	25.4	46.1	43.4	25.4	45.6	42.9
eval	22.0	48.2	44.6	22.0	47.9	44.0
IB4003	21.4	48.3	44.4	21.4	47.5	43.8
IB4004	22.4	48.5	44.4	23.1	48.4	43.9
IB4005	25.4	45.9	42.4	25.3	45.6	42.2
IS1008c	15.9	52.9	50.0	14.9	52.3	48.6
TS3005a	23.1	45.2	41.9	23.2	45.3	41.7

Table 5: Performance of systems with and without WSD for dev and eval sets as well as individual meetings in the eval set.

approximating the larger space of feasible translations that could be obtained by producing additional human references. Note that the thesaurus provided synonyms for only roughly 50% of all content words in the dev and eval set. For each of them, on average three synonyms are found in the thesaurus. We use these extended references to recompute the PER score as an indicator of correct word selection. All results (BLEU, PER and extended PER (or XPER)) are shown in Table 5. As expected, BLEU is not affected but WSD improves the PER and XPER slightly but consistently. Note that this is despite the fact that only roughly half of all content words received disambiguation scores.

Finally, we provide a concrete example of translation improvements, with improved words highlighted:
Source:
on the balcony
there's that **terrace**
there's no place inside the building
Translation, no WSD:
auf dem balkon
es ist das **absatz**
es gibt keinen platz innerhalb des gebäudes
Translation, with WSD:
auf dem balkon
es ist das **terrasse**
es gibt keinen platz gebäudeintern
References:
auf dem balkon / auf dem balkon
da gibt es die **terrasse** */ da ist die* **terrasse**
es gibt keinen platz im gebäude / es gibt keinen platz innen im gebäude

6 Summary and Conclusions

We have presented a study on statistical translation of meeting data that makes the following contributions: to our knowledge it presents the first quantitative analysis of contextual factors in the statistical translation of multi-party spoken meetings. This analysis showed that the largest impact could be obtained in the area of word sense disambiguation using topic and domain knowledge, followed by multimodal information to resolve addressees of *you*. Contrary to our expectations, further knowledge of the real-world environment (such as objects in the room) did not show an effect on translation performance. Second, it demonstrates the application of *unsupervised, global* WSD to SMT, whereas previous work has focused on supervised, local WSD. Third, it explores definitions derived from collaborative Wiki sources (rather than WordNet or existing dictionaries) for use in machine translation. We demonstrated small but consistent improvements even though word coverage was incomplete. Future work will be directed at improving word coverage for the WSD algorithm, investigating alternative word similarity measures, and exploring the combination of global and local WSD techniques.

Acknowledgments

This work was funded by the National Science Foundation under Grant IIS-0840461 and by a grant from the University of Washington's Provost Office. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.

References

S. Brin and L. Page. 1998. “The Anatomy of a Large-Scale Hypertextual Web Search Engine”. *Proceedings of WWW7.*

C. Callison-Burch, P. Koehn and M. Osborne. 2006. “Improved Statistical Machine Translation Using Paraphrases”. *Proceedings of NAACL.*

M. Carpuat and D. Wu. 2005. “Word sense disambiguation vs. statistical machine translation”. *Proceedings of ACL.*

M. Carpuat and D. Wu. 2007. “Improving statistical machine translation using word sense disambiguation”. *Proceedings of EMNLP-CoNLL.*

Y.S. Chan and H.T. Ng and D. Chiang 2007. ”Word sense disambiguation improves statistical machine translation”. *Proceedings of ACL.*

P. Chen, W. Ding, C. Bowes and D. Brown. 2009. “A fully unsupervised word sense disambiguation method using dependency knowledge”. *Proceedings of NAACL.*

E. Gabrilovich and S. Markovitch. 2007 “Computing semantic relatedness using Wikipedia-based explicit semantic analysis”. *Proceedings of IJCAI.*

R. Haque, S.K. Naskar, Y. Ma and A. Way. 2009. “Using supertags as source language context in SMT”. *Proceedings of EAMT.*

A.S. Hildebrandt, M. Eck, S. Vogel and A. Waibel. 2005. “Adaptation of the Translation Model for Statistical Machine Translation using Information Retrieval”. *Proceedings of EAMT.*

H. Hoang and P. Koehn. 2008. “Design of the Moses decoder for statistical machine translation”. *Proceedings of SETQA-NLP.*

P. Koehn. 2005. “Europarl: a parallel corpus for statistical machine translation”. *Proceedings of MT Summit.*

V. Kumar, R. Sridhar, S. Narayanan and S. Bangalore. 2008. “Enriching spoken language translation with dialog acts”. *Proceedings of HLT.*

Z. Li et al.. 2009. “Joshua: An Open Source Toolkit for Parsing-based Machine Translation”. *Proceedings of StatMT.*

E. Matusov, M. Popović, R. Zens and H. Ney. 2004. “Statistical Machine Translation of Spontaneous Speech with Scarce Resources”. *Proceedings of IWSLT.*

A. McCowan. 2005. “The AMI meeting corpus”,

H. Mima, O. Furuse and H. Iida. 1998. “Improving Performance of Transfer-Driven Machine Translation with Extra-Linguistic Information from Context, Situation and Environment”. *Proceedings of Coling. Proceedings of the International Conference on Methods and Techniques in Behavioral Research.*

D.S. Munteanu and D. Marcu. 2005. “Improving machine translation performance by exploiting non-parallel corpora”. *Computational Linguistics.*

R. Navigli and M. Lapata. 2007. “Graph Connectivity Measures for Unsupervised Word Sense Disambiguation”, *Proceedings of IJCAI*

S. Rao and I. Lane and T. Schultz. 2007. “Improving spoken language translation by automatic disfluency removal”. *Proceedings of MT Summit.* 31(4).

R. Sinha and R. Mihalcea. 2007. ”Unsupervised Graph-based Word Sense Disambiguation Using Measures of Word Semantic Similarity”, *Proceedings of IEEE-ICSC*

N. Stroppa, A. Bosch and A. Way. 2007. “Exploiting Source Similarity for SMT using Context-Informed Features”. *Proceedings of TMI.*

S. Stüker, C. Fügen, F. Kraft and M. Wölfel. 2007. “The ISL 2007 English Speech Transcription System for European Parliament Speeches”. *Proceedings of Interspeech.*

A. Waibel and C. Fügen. 2008. “Spoken Language Translation – Enabling cross-lingual human-human communication”. *Proceedings of Coling).*

W. Wang, G. Tur, J. Zheng and N.F. Ayan. 2010. “Automatic disfluency removal for improving spoken language translation”. *Proceedings of ICASSP. IEEE Signal Processing Magazine*

H. Wu, H. Wang and C. Zong. 2008. “Domain adaptation for statistical machine translation with domain dictionary and monolingual corpora”,

T. Zesch, C. Müller and Iryna Gurevych. 2008. “Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary”. *Proceedings of LREC.*

T. Zesch, Christof Müler and Iryna Gurevych. 2008. “Using Wiktionary for Computing Semantic Relatedness”.

J. Zheng, W. Wang and N.F. Ayan. 2008. “Development of SRI’s translation systems for broadcast news and broadcast conversations”. *Proceedings of Interspeech. Proceedings of AAAI.*

A Working Report on Statistically Modeling Dative Variation in Mandarin Chinese

Yao Yao
University of California, Berkeley
Department of Linguistics
yaoyao@berkeley.edu

Feng-hsi Liu
University of Arizona
Department of East Asian Studies
fliu@u.arizona.edu

Abstract

Dative variation is a widely observed syntactic phenomenon in world languages (e.g. *I gave John a book* and *I gave a book to John*). It has been shown that which surface form will be used in a dative sentence is not a completely random choice, rather, it is conditioned by a wide range of linguistic factors. Previous work by Bresnan and colleagues adopted a statistical modeling approach to investigate the probabilistic trends in English dative alternation. In this paper, we report a similar study on Mandarin Chinese. We further developed Bresnan et al.'s models to suit the complexity of the Chinese data. Our models effectively explain away a large proportion of the variation in the data, and unveil some interesting probabilistic features of Chinese grammar. Among other things, we show that Chinese dative variation is sensitive to heavy NP shift in both left and right directions.

1 Introduction

1.1 Overview

In traditional linguistic research, the study of syntax is most concerned with grammaticality. Sentences are either grammatical or ungrammatical, and syntactic theories are proposed to explain the structural features that cause (un)grammaticality. Meanwhile, little attention has been paid to the relative acceptability of grammatical sentences. If two sentences are both grammatical and basically express the same meaning, are they equally likely to occur in the language? The answer is probably *no*. For example, in English, the sentence *I have read that book* is much more frequent than *That book I have read*. The latter topicalized sentence is only used when the entity denoted by *That book* is in focus. This indicates that the choice of surface sentence form is not entirely random, but conditioned by some factors including information status.

Thus, instead of categorizing sentences as grammatical or ungrammatical, a better way to express the degree of grammaticality would be to use a likelihood continuum, from 0 to 1, where ungrammatical sentences have zero likelihood and grammatical sentences fall somewhere between 0 and 1, with some being more likely than others. The idea of associating linguistic forms with various probabilities has been around for a while (see Jurafsky, 2003 and Manning, 2003 for an extensive review). Recent psycholinguistic research has shown that just like grammaticality, the likelihoods of sentence forms are also part of the user's linguistic knowledge. Sentences with high probabilities are in general easier to comprehend and produce, and their production is more prone to phonetic reduction (Bresnan, 2007; Gahl and Garnsey, 2004; Levy, 2008; among others). The famous example of garden path sentences also exemplifies the difficulty of comprehension in low-probability sentence forms.

If we accept the premise of probabilistic syntax, then an immediate question is what determines these probabilities. In the current work, we address this question by investigating a particular type of probabilistic phenomenon, i.e. dative variation in Chinese. We show that the probabilities of

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1236–1244,
Beijing, August 2010

various surface forms of Chinese dative sentences can be well estimated by a linear combination of a set of formal and semantic features.

The remainder of this paper is organized as follows. Section 1.2 briefly reviews previous work on English dative variation. Section 1.3 introduces dative variation in Chinese. Section 2 describes the dataset and the statistical models used in the current study. Section 3 presents modeling results, followed by a discussion in Section 4. Section 5 concludes the paper with a short summary. To preview the results, we show that dative variation in Chinese is more complicated than in English, in that it features two levels of variation, which exhibit different (sometimes even opposite) probabilistic patterns.

1.2 Dative variation in English

A dative sentence is a sentence that encodes a transfer event. Typical verbs of transfer in English include *give, send, mail*, etc. A characterizing property of transfer events is that they often involve two objects. In addition to the direct object (DO), the verb also takes an indirect object (IO) which usually denotes the recipient of the transfer action. For instance, in sentence 1a, the direct object is *a book* and the indirect object is *John*.

Cross-linguistically, it has been documented that many languages in the world have multiple syntactic forms for encoding the same transfer event (Margetts and Austin, 2007 , among others). In English, both 1a and 1b describe the same event, but 1a is a double object form (V IO DO) while 1b takes a prepositional phrase (V DO *to* IO).

(1) a. I gave John a book. → V IO DO
b. I gave a book to John. → V DO *to* IO

A number of conditioning factors have been identified for the alternation between the two surface forms. For instance, when the indirect object is a pronoun (e.g. *him*), it is more likely to have the double object form (i.e. *I gave him a book*) than the PP form (i.e. *I gave a book to him*). On the other hand, if the indirect object is a complex NP (with relative clauses), it tends to occur at the end of the sentence. Since most of these effects are subtle and often correlated with each other (e.g. definiteness, pronominality and syntactic complexity), investigating individual factors can give convoluted and unreliable results. To avoid this problem, many recent works in the field adopted a statistical modeling approach (Bresnan et al., 2007; Wasow and Arnold, 2003, among others). Instead of investigating separate factors, statistical models are built on large-scale datasets, using all potential conditioning factors to predict the surface form. In Bresnan et al. (2007), a dozen predictors relating to the verb (type of transfer event), the two object NPs (accessibility, pronominality, definiteness, syntactic complexity, etc), and the discourse (presence of parallel structures) were used to make the prediction. Using data input from 2,360 dative sentences from the Switchboard corpus, the model correctly predicted surface form in 97% of the sentences, which was a great improvement over the baseline prediction accuracy of 79% (i.e. the percentage of correct responses if the model knows nothing but which variant is more frequently used). It also showed that dative variation in English was indeed sensitive to all the predictors in the model.

1.3 Dative variation in Chinese

Dative variation in Chinese is much more complicated than in English. In addition to the two word orders that exist in English (2a, 2b), it is also common for direct object to appear before the verb, as in a BA construction or a topicalized sentence (2c). Besides, indirect object can also precede the verb, as shown in 2d. Another dimension of variation is in the use of coverbs *gei* and *ba*, both of which can be optional (2b, 2c; see Li and Thompson, 1981 for a detailed discussion on this), or replaced by other morphemes (*zhu, yu, jiang*, etc).

(2) a. John song-le shu gei Mary.
John give-ASP book to Mary
John gave one/some book(s) to Mary.
→ V DO IO

b. John song (gei) Mary yiben shu.
John gave (to) Mary one book
John gave Mary a book.
→ V IO DO

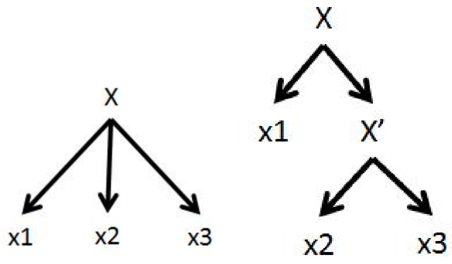
c. John ba shu song (gei) Mary, (ba)
John BA book gave (to) Mary (BA)

jiu song (gei) Kate.
wine gave (to) Kate
John gave the book(s) to Mary and gave the wine to Kate.
→ DO V IO

d. Ta meiren fa-le yiben shu.
He everyone allocated one book
He gave everyone a book.
→ IO V DO

For the purpose of the current study, we will ignore the existence (hence also the variation) of *gei* and *ba*, and concentrate on the variation in the relative order of V, DO and IO. In addition, our corpus search shows that sentences in the form of IO V DO are the least frequent (<9%) and mostly limited to a small set of verbs (mostly *fa* "to allocate" and *banfa* "to award"), so we drop this category from the current study. Thus the three remaining word order variants are: DO V IO, V DO IO, and V IO DO.

Generally speaking, there are two ways of modeling a variation phenomenon involving three variants. One way is to assume that the three variants are equally dissimilar from one another and the selection process is just to pick one out of three (Fig. 1a). The other approach is to assume a hierarchical structure: two of the variants are more similar to each other than they are to the third one and thus form a subcategory first before they join the third variant (Fig. 1b). In the selection process, the user first selects the subcategory (i.e. **x1** or **x'** in Fig 1b), and depending on which subcategory is chosen, they might need to make a second choice between two end nodes (i.e. **x2** and **x3**).



(a) (b)

Figure 1: Two possible schemas

We argue that the variation among the three word order variants in the current study is better modeled by a schema like Fig 1b, for both theoretical and methodological reasons. First, V DO IO and V IO DO are structurally more similar to each other than they are to DO V IO. Both V DO IO and V IO DO are in canonical word order of Chinese but the form DO V IO features the preposing (or topicalization) of the DO, whether or not the BA morpheme is present. Object preposing also exists outside ditransitive sentences (e.g. 3). Previous research has associated object preposing with the disposal meaning of the verb phrase, and the definiteness, givenness and weight of the object NP (Li and Thompson, 1981; Liu, 2007).

(3) a. Wo ba fan chi wan le.
I BA rice eat finish SEP
I have finished the rice.

b. Ta zhe dianying kan-le henduo bian.
he this movie saw many time
He has watched this movie for many times.

There is also a methodological motivation for adopting a hierarchical schema. Though it is not impossible to model a categorical variation with more than two variants (using multinomial logistic regression), binary variation is much easier to model and the interpretation of the results is more straightforward (this is especially true when random effects are present).

In view of the above, we propose the schema in Fig 2 for modeling the current variation phenomenon. We refer to sentences in the form of DO V IO as preverbal ditransitive sentences (since DO is before the verb), while both V DO IO and V IO DO are postverbal ditransitives. The distinction between the latter two forms regards whether DO is before or after IO, therefore one is termed as pre-IO and the other post-IO. Compared with the upper-level preverbal-postverbal distinction, the lower-level variation is much less studied in the literature (though see Liu, 2006 for a relevant discussion).

Corresponding to the schema in Fig 2, we constructed two separate models, one for the upper-level variation ("upper model") and the other for the lower-level variation ("lower model").

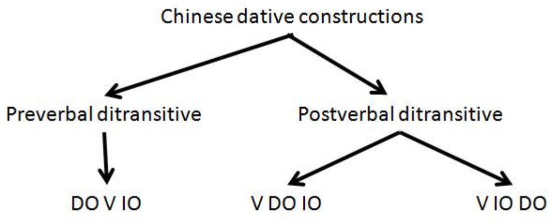


Figure 2: A two-level schema for Chinese dative variation

2 Methodology

2.1 Corpus and dataset

The data we use are from the Sinica Corpus of Modern Chinese (v3.1; Huang et al., 1995). We first compiled a list of 36 verbs that could be used ditransitively (see Appendix A) and then extracted from the corpus all sentences containing these words (n= 48,825 sentences). We then manually went through the sentences and selected those that (a) featured the ditransitive sense of the target verb, with both object NPs being overt, and (b) were in the form of any of the three form variants. 1,574 sentences remained after step (a) [1] and 1,433 after step (b) [2].

Further removal was conducted on verbs that were too sparse in the dataset. In each variation model, we removed verbs with fewer than two occurrences under either form variant. The final dataset for the upper model has 1149 sentences (of 20 verb types) while the dataset for the lower model has 801 sentences (of 14 verb types). The latter dataset is largely but not fully contained in the former due to the elimination of low-frequency verbs.

2.2 Data annotation

Similar to Bresnan et al.'s work on English, we annotated each data sentence for a wide range of features pertaining to the verb and the two NPs (see Appendix B for a complete list of annotated factors). Specifically, the verb was coded either as expressing a canonical transfer event, such as *ji* "to mail", or an extended transfer event, such as *jieshao* "to introduce". Semantic annotation of the two NPs is much trickier in Chinese than in English due to the lack of morphology. In practice, we used Bresnan et al.'s criteria for English, whenever applicable (e.g. accessibility, person, concreteness, animacy). In cases where the English rules did not apply (e.g. definiteness and number of bare NPs in Chinese), we developed working principles based on phrasal substitution. For example, if a bare NP can take a specifier like *yige/yizhi* "one" without changing sentence meaning, it is considered to be indefinite. Conversely, if a bare NP is better replaced with a full NP with a demonstrative *zhege* "this" or *nage* "that", it is coded as definite. Similar rules were used to assist annotating the number feature, using specifiers *yige/nage* "one"/"that" and *yixie/naxie* "some"/"those".

In addition to the factors in the English model, we also coded a set of structural features, including the presence of a following verb after the ditransitive construction, the presence of quantifiers/numerals in the NPs, and whether or not the ditransitive structure is embedded, nominalized, or relativized, etc. We suspect that since semantic features are often covert in Chinese words, it is possible that overt marking (e.g. the use of quantifiers/numerals) plays a more important role in conditioning surface form variation.

Finally we also included genre in the model. Sentences listed under the categories of dialogue and speech in the Sinica corpus were coded as "spoken" and the rest are coded as "written".

Altogether 24 factors were annotated and included in the statistical models as predictor variables. All variables are categorical except for the (log) length difference between DO and IO, which is numerical.

2.3 Statistical models

The statistical tool we use is mixed-effects logistic regression models. Compared with regular logistic regression models, mixed-effects models are more sophisticated in that they allow the user to specify factors that might introduce ran-

[1] A vast number of sentences were removed because the target verb was not used as a verb, or used with a different sense, or used as part of a different verb phrase, e.g. *fa* to allocate could also mean to bloom or be used in *fazhan* to develop, *faxian* to discover, etc.

[2] 141 sentences were removed because they were in the form of IO V DO.

dom variation in the dataset. In the current study, the datasets in both models contain sentences with different verbs. It is possible that different verbs have different intrinsic tendencies toward a certain word order variant.[3] Incorporating this piece of information into the model makes it more powerful and less affected by the unbalanced distribution of verb types. The mathematical formula of the mixed-effects logistic regression model is given below.

(4) Probability(V DO IO) = $\frac{1}{1+e^{-(\alpha_i + x\beta)}}$, where α_i is the verb-specific intercept of the verb v_i , x is a vector of predictors and β is a vector of corresponding coefficients.

Using the annotated datasets described in 2.2, we built an upper model and a lower model, corresponding to the schema in Fig 2. The general procedure of statistical analysis (which is the same for both models) is described as follows.

We first run the model with all 24 predictors, which will generate a coefficient and a p value for each predictor. Then we refit the model with only significant predictors (i.e. $p < 0.05$). The purpose of doing so is to filter out the noise in the model fit created by the large number of insignificant predictors. Only predictors that remain significant in the simplified model with largely unchanged coefficients are considered to be reliably significant.

Two model evaluation techniques are used to check the model results: cross-validation and separate analysis of high-frequency verbs. A potential problem in any statistical model is that it might overfit the data. After all, what we are interested in is the general probabilistic trends in dative variation, not the trends in a particular set of sentences featuring a particular set of verbs. A cross-validation test helps us evaluate the generalizability of model results by running the same model on a randomly sampled subset of the data. In doing so, it simulates the effect of having different datasets. In practice, we use two types of cross-validation procedures: one randomly samples sentences and the other samples verbs. Each procedure is executed on 100 randomly sampled subset of half the sentences/verbs. Only predictors with consistent performance over all iterations in both tests will be considered as stable.

Another concern in the model design is the effect of verb frequency. In the current dataset, one verb, i.e. *tigong* “to provide”, is extremely frequent. 37.3% of the sentences in the upper model and 50.9% in the lower model come from this verb. Though in theory, verb frequency is already taken care of by using mixed-effects models and running cross-validation on samples of the verb set, it is still necessary to test *tigong* separately from the rest of the verbs, due to its extremely high frequency. In the next section, we will report in detail the results from the two regression models.

3 Results

3.1 Upper model: predicting preverbal and postverbal variation

In the upper model, the distinction is between preverbal (DO V IO; coded as 1) and postverbal ditransitives (V DO IO and V IO DO; both coded as 0). The dataset in this model contains 1,149 sentences (of 20 verb types), with 379 preverbal and 770 postverbal. The distribution of the verbs is highly skewed. The most frequent verb is *tigong* “to provide” (n=428 tokens), followed by *song* “to send” (135) and *jiao* “to hand; to transfer” (117). The remaining 17 verbs have between 5 and 54 occurrences in the dataset.

10 out of 24 predictors in the full model are significant and most of them remain significant when the other 14 predictors are removed from the model. Table 1 below summarizes the results of the simplified model.

Judging from the signs of the coefficients in Table 1, a dative sentence is more likely to take the preverbal form (as opposed to the postverbal form) when (a) the verb expresses canonical transfer event, (b) DO is definite, plural, abstract and given in the previous context, with no quantifiers or numerals, (c) IO is not a pronoun and is not given in the previous context, and (d) DO is longer

[3] The same can be said about individual speakers, as some speakers might be more inclined to use certain forms than other speakers. However, since the sentences in the current datasets were sampled from a vast pool of speakers/writers (given the way the corpus is developed), individual differences among speakers is not considered in the current model.

Predictor	β	***p***
verb is canonical	1.71	0.03
DO is given	1.22	<0.001
DO is definite	4.89	<0.001
DO is plural	1.4	<0.001
DO is concrete	-1.13	0.004
quan/num in DO	-0.99	0.005
IO is pronoun	-1.64	<0.001
IO is given	-0.9	0.007
quan/num in IO	1.32	0.07 (n.s.)
Len(DO)-Len(IO)	0.53	0.002

Table 1: Fixed effects in the simplified upper model

than IO.

Table 2 shows the accuracy of the simplified model. If 0.5 is used as the cutoff probability, the model correctly predicts for (737+338)/1149=93.6% of the sentences. For comparison, the baseline accuracy is only 770/1049=-67% (i.e. by guessing postverbal every time). In other words, the model only needs to include 10 predictors to achieve an increase of around 39% (93.6-67)/67) in model accuracy.

		Predicted	
		preverbal	postverbal
observed	preverbal	338	41
	postverbal	33	737

Table 2: Prediction accuracy of the simplified upper model

Results from the two cross-validation tests confirm all the predictors regarding DO in Table 1, as well as the pronominality of IO and the length difference between DO and IO. Verb category and the givenness of IO do not survive the cross-validation tests.

Separate analysis of tigong shows that indeed, the extremely high-frequency verb exhibits vastly different patterns than other verbs. Only one predictor turns out to be significant for *tigong* sentences, that is, the definiteness of DO (β = 6.17, $p < 0.001$). A closer look at these sentences suggests that they are strongly biased toward postverbal word order, in that 400 out of 428 (95.4%) *tigong* sentences are postverbal (compared with the average level of 67% in all sentences). In other words, just by guessing postverbal every time, one is able to make the correct prediction for *tigong* over 95% of the time. Not surprisingly, there is little need for additional predictors. For non-*tigong* sentences, all factors in Table 1 are significant except for verb category and the presence of quantifiers/numerals in IO. Overall, the non-*tigong* model has an accuracy of 91.5% (baseline = 50.6%).

To sum up, we are confident to say that the semantic features of DO, as well as pronominality of IO and the length difference between the two objects, play important roles in conditioning the preverbal-postverbal variation. Knowing these factors boosts the model s predicting power by a great deal.

3.2 Lower model: predicting pre-IO and post-IO variation

In the lower model, the distinction is between pre-IO sentences (i.e. V DO IO; coded as 1) and post-IO sentences (i.e. V IO DO; coded as 0). The dataset consists of 801 sentences of 14 verb types, among which 161 are pre-IO and 640 are post-IO. The most frequent verb is again, *tigong* (n=408 tokens), followed by *dai* "to bring" (137) and *song* "to send" (89).

Table 3 below summarizes the results of the simplified version of the lower model (constructed in the same fashion as described in Section 3.1).

Predictor	β	***p***
DO is definite	1.59	0.006
DO is concrete	1.06	<0.001
DO is plural	-0.57	0.04
followed by a verb	2.29	<0.001
normalized verb phrase	1.36	0.13 (n.s.)
Len(DO) - Len(IO)	-1.37	<0.001

Table 3: Fixed effects in the simplified lower model

Compared to the upper model, fewer predictors are significant in the lower model. Everything else being equal, a postverbal ditransitive sentence is more likely to take the pre-IO form (V DO IO) if (a) DO is definite and concrete, (b) IO is singular, (c) DO is shorter than IO, and (d) the ditransitive construction is followed by another verb. The last point is illustrated in sentence 5a, which is adapted from a real sentence in the corpus. In 5a, the NP *women* "we" is both the recipient of the first verb *song* "to send" and the agent of the second verb *chi* "to eat". Thus, by using a pre-IO form, the NP *women* is in effect adjacent to the second verb chi, which might give an advantage in sentence processing. Notice though, if the other form (V IO DO) is used, the sentence is still grammatical (see 5b).

(5) a. Ta hai song xiaoye gei wo chi.
he also sent snacks to me eat
He also sent snacks for me to eat.

b. Ta hai song (gei) wo xiaoye chi.
he also sent (to) me snacks eat
He also sent me snacks to eat.

Overall the lower model is not as successful as the upper model. The prediction accuracy is 87.7% (baseline accuracy is 79.9%; see Table 4).

		Predicted	
		pre-IO	post-IO
observed	pre-IO	85	76
	post-IO	22	618

Table 4: Prediction accuracy of the simplified lower model

Moreover, cross-validation and the analysis of *tigong* show that only two factors, the presence of the following verb and length difference, are stable across subsets of the data. In fact, with length difference alone, the model generates correct predictions for 86.8% of the sentences (only 1% less than the accuracy reported in Table 4).

However, before we hastily conclude that length difference is the only thing that matters in the lower-level variation, it is important to point out that when the length factor is removed from the model, some predictors (such as the accessibility of DO) turn out to be significant and the model still manages to achieve an accuracy of 85.3%. Therefore, a more plausible explanation is that length difference is the strongest predictors for lower-level dative variation. Though the part of variation it accounts for can also be explained by other predictors, it is more effective in doing so. Therefore the existence of this variable tends to mask other predictors in the model.

4 Discussion

4.1 Comparing he two models

In the current study, we propose a two-level hierarchical schema for modeling the variation among three major word orders of Chinese dative sentences. On the upper level, there is a distinction between sentences with preverbal DOs and those with postverbal DOs. On the lower level, among postverbal sentences, there is a further distinction between pre-IO sentences (i.e. with prepositional phrases), and post-IO sentences (i.e. double object forms). This schema is promoted by structural as well as methodological concerns.

Our modeling results show that the two levels of variation are indeed characterized by different probabilistic patterns, which in turn provide evidence for our original proposal. As presented in Section 3, the upper-level distinction is mostly conditioned by the semantic features of the DO. However, in the lower-level variation, the two best predictors are length difference and the presence of a following verb. Overall, the upper-level model is more successful (accuracy = 93.6%, baseline = 67%) than the lower-level model (accuracy = 87.7%, baseline = 79.9%).

A more striking difference between the two models is that they exhibit weight effects in opposite directions. In both models, length difference between DO and IO plays an important role. Nevertheless, in the upper model, length difference has a positive sign (β = 0.53), meaning that the longer the DO is (compared to the IO), the more likely it is to prepose DO before the verb. Conversely, in the lower-level model, this factor has a negative sign (β = - 1.37), which means that the longer the DO is (compared to the IO), the less likely it is for DO to be before IO. That is to say, everything else being equal, if a DO is long, it will probably be preposed before the verb, but if it is already after the verb, then it will more likely be placed after IO, at the end of the construction.

The difference in directionality explains why it is only in the lower-level model that the weight effect overshadows other predictors. Features like pronominality, definiteness, and accessibility are inherently correlated with weight. Pronouns are shorter than full NPs; definite NPs tend to be shorter than indefinite NPs (which often take quantifiers and numerals); NPs that have appeared before tend to be in shorter forms than their first occurrences. In both models, a general trend is that NPs that are more prominent in the context (e.g. pronouns, definite NPs, NPs with antecedents) tend to occur earlier in the construction. Thus, in the lower model, the general trend of prominence is confluent with the short before long weight effect, but in the upper model, it is pulling away from the long before short weight effect. As a result, weight effect only masks semantic predictors in the lower model, not in the upper model.

4.2 Comparing with English dative variation

Compared with Bresnan et al.s models, the current results reveal a number of interesting differences between Chinese and English dative variation.

First, the variation phenomenon in Chinese involves at least one more major variant, that is, the preverbal word order, which significantly increases the complexity of the phenomenon. The fact that overall the English model has greater prediction accuracy than the Chinese models might have to do with the fact that the variation phenomenon is more complicated and harder to model in Chinese.

Second, dative variation in Chinese seems to be less sensitive to semantic features. If we only consider the lower-level variation in Chinese, which involves the same form variants as in English (i.e. V DO IO and V IO DO), the Chinese model is best predicted by the length difference between DO and IO and most other predictors are muted by the presence of this factor. In the English model, semantic features are still significant even when length difference is controlled.

Last but not least, as discussed at length in the previous section, the two levels of dative variation in Chinese exhibit weight effects in opposite directions. The English variation is also sensitive to weight, but only in the short before long direction, which is the same as the lower-level variation in Chinese.

5 Conclusion

In this work, we present a corpus-based statistical modeling study on Chinese dative variation. In doing so, we show that this new methodology, which combines corpus data and statistical modeling, is a powerful tool for studying complex variation phenomena in Chinese. The statistical models built in the current study achieve high accuracy in predicting surface forms in Chinese dative sentences. More importantly, the models unveil probabilistic tendencies in Chinese grammar that are otherwise hard to notice.

A remaining question in the current study is *why would Chinese dative variation exhibit weight effects in both directions*. The answer to this question awaits further investigation.

Acknowledgement

We would like to thank three anonymous reviewers for helpful comments on an earlier version of the paper. We owe special thanks to Joan Bresnan and her colleagues in the Spoken Syntax Lab at Stanford University, for sharing working manuals and for valuable discussions.

References

Bresnan, J. (2007). Is syntactic knowledge probabilistic? Experiments with the English dative alternation. In Featherston, S. and Sternefeld, W., editors, *Roots: Linguistics in search of its evidential base*, Studies in generative gramar, pages 77–96. Mouton de Gruyter, Berlin.

Bresnan, J., Cueni, A., Nikitina, T., and Baayen, H. (2007). Predicting the dative alternation. In Boume, G., Kraemer, I., and Zwarts, J., editors, *Cognitive foundations of interpretation*, pages 69–94. Royal Netherlands Academy of Science, Amsterdam.

Gahl, S. and Garnsey, S. (2004). Knowledge of grammar, knowledge of usage: Syntactic probabilities affect pronunciation variation. *Language*, 80(4):748–775.

Huang, C., Chen, K., Chang, L., and Hsu, H. (1995). An introduction to Academia Sinica Balanced Corpus. [in chinese]. In *Proceedings of ROCLING VIII*, pages 81–99.

Jurafsky, D. (2003). Probabilistic modeling in psycholinguistics: Linguistic comprehension and production. In Rens Bod, J. H. and Jannedy, S., editors, *Probabilistic Linguistics*, pages 39–96. MIT Press, Cambridge, Massachusetts.

Levy, R. (2008). Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–1177.

Li, C. N. and Thompson, S. A. (1981). *Mandarin Chinese: A functional reference grammar*. University of California Press, Berkeley.

Liu, F. (2006). Dative constructions in Chinese. *Language and Linguistics*, 7(4):863–904.

Liu, F. H. (2007). Word order variation and ba sentences in Chinese. *Studies in Language*, 31(3):649 – 682.

Manning, C. D. (2003). Probabilistic syntax. In Rens Bod, J. H. and Jannedy, S., editors, *Probabilistic Linguistics*, pages 289–341. MIT Press, Cambridge, Massachusetts.

Margetts, A. and Austin, P. (2007). Three participant events in the languages of the world: toward a cross-linguistic typology. *Linguistics*, 45(3):393–451.

Wasow, T. and Arnold, J. (2003). Post-verbal constituent ordering in english. In Rohdenburg, G. and Mondorf, B., editors, *Determinants of Grammatical Variation in English*, pages 119–154. Mouton.

Appendices

A Complete verb list [4]

song “to send”, *tigong* “to provide”, *jie* “to lend (to)”, *fu* “to pay”, *ban* “to award”, *banfa* “to award”, *zengsong* “to send (as a gift)”, *shang* “to award”, *jieshao* “to introduce”, *huan* “to return”, *fa* “to distribute/allocate”,*jiao* “to transfer”, *ji* “to mail”, *liu* “to leave (behind)”, *liuxia* “to leave (behind)”,*reng* “to throw”, *diu* “to throw”, *diuxia* “to throw (behind)”, *juan* “to donate”, *juanzeng* “to donate”, *juanxian* “to donate”, *bo* “to allocate”, *di* “to hand (to)”, *zu* “to rent (to)”, *fen* “to distribute”, *na* “to hand (to)”, *dai* “to bring”, *dailai* “to bring”, *jiao* “to teach”, *chuan* “to deliver”, *chuanran* “to pass around (a disease)”, *chuanda* “to deliver (a message)”, *chuansong* “to deliver” , *chuanshou* “to deliver (knowledge)”,*ci* “to give (as a reward)”, *pei* “to pay (compensation)”

B Predictors in the full model

Predictor	**Coding**
genre	1=spoken; 0=written
verb category	1=canonical transfer; 0=otherwise
definiteness of DO	1=definite; 0=indefinite
pronominality of DO	1=pronoun; 0=otherwise
number of DO	1=plural; 0=singular
person of DO	1=1st and 2nd person; 0=otherwise
concreteness of DO	1=concrete; 0=abstract
givenness of DO	1=given; 0=otherwise
quan/num in DO	1=yes; 0=no
definiteness of IO	1=definite; 0=indefinite
pronominality of IO	1=pronoun; 0=otherwise
number of IO	1=plural; 0=singular
person of IO	1=1st and 2nd person; 0=otherwise
concreteness of IO	1=concrete; 0=abstract
givenness of IO	1=given; 0=otherwise
followed by another verb	1=yes; 0=no
embedded under another verb	1=yes; 0=no
part of a copular sentence	1=yes; 0=no
adverbial phrase after the verb	1=yes; 0=no
particle after the verb	1=yes; 0=no
question form	1=yes; 0=no
sentence negation	1=yes; 0=no
relativization	1=yes; 0=no
nominalization	1=yes; 0=no
log(len(DO)- log(len(IO))	numerical

[4] The verb *gei* “to give” is not included in the list, because it has the same form as the coverb gei and therefore has different properties than other ditransitive verbs. Among other things, the verb *gei* cannot take the V DO IO form in Mandarin (e.g. **gei yiben shu gei wo* “give a book to me”).

Kernel Slicing: Scalable Online Training with Conjunctive Features

Naoki Yoshinaga
Institute of Industrial Science,
the University of Tokyo
ynaga@tkl.iis.u-tokyo.ac.jp

Masaru Kitsuregawa
Institute of Industrial Science,
the University of Tokyo
kitsure@tkl.iis.u-tokyo.ac.jp

Abstract

This paper proposes an efficient online method that trains a classifier with many conjunctive features. We employ kernel computation called *kernel slicing*, which explicitly considers conjunctions among frequent features in computing the polynomial kernel, to combine the merits of linear and kernel-based training. To improve the scalability of this training, we reuse the temporal margins of partial feature vectors and terminate unnecessary margin computations. Experiments on dependency parsing and hyponymy-relation extraction demonstrated that our method could train a classifier orders of magnitude faster than kernel-based online learning, while retaining its space efficiency.

1 Introduction

The past twenty years have witnessed a growing use of machine-learning classifiers in the field of NLP. Since the classification target of complex NLP tasks (*e.g.*, dependency parsing and relation extraction) consists of more than one constituent (*e.g.*, a head and a dependent in dependency parsing), we need to consider *conjunctive features*, i.e., conjunctions of primitive features that focus on the particular clues of each constituent, to achieve a high degree of accuracy in those tasks.

Training with conjunctive features involves a space-time trade-off in the way conjunctive features are handled. Linear models, such as log-linear models, explicitly estimate the weights of conjunctive features, and training thus requires a great deal of memory when we take higher-order conjunctive features into consideration. Kernel-based models such as support vector machines, on the other hand, ensure space efficiency by using the kernel trick to implicitly consider conjunctive features. However, training takes quadratic time in the number of examples, even with online algorithms such as the (kernel) perceptron (Freund and Schapire, 1999), and we cannot fully exploit ample 'labeled' data obtained with semi-supervised algorithms (Ando and Zhang, 2005; Bellare et al., 2007; Liang et al., 2008; Daumé III, 2008).

We aim at resolving this dilemma in training with conjunctive features, and propose online learning that combines the time efficiency of linear training and the space efficiency of kernel-based training. Following the work by Goldberg and Elhadad (2008), we explicitly take conjunctive features into account that frequently appear in the training data, and implicitly consider the other conjunctive features by using the polynomial kernel. We then improve the scalability of this training by a method called *kernel slicing*, which allows us to reuse the temporal margins of partial feature vectors and to terminate computations that do not contribute to parameter updates.

We evaluate our method in two NLP tasks: dependency parsing and hyponymy-relation extraction. We demonstrate that our method is orders of magnitude faster than kernel-based online learning while retaining its space efficiency.

The remainder of this paper is organized as follows. Section 2 introduces preliminaries and notations. Section 3 proposes our training method. Section 4 evaluates the proposed method. Section 5 discusses related studies. Section 6 concludes this paper and addresses future work.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1245–1253,
Beijing, August 2010

Algorithm 1 BASE LEARNER: KERNEL PA-I

```
INPUT: $\mathcal{T} = \{(\boldsymbol{x}, y)_t\}_{t=1}^{|\mathcal{T}|}, k : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}, C \in \mathbb{R}^+$
OUTPUT: $(\mathcal{S}_{|\mathcal{T}|}, \boldsymbol{\alpha}_{|\mathcal{T}|})$
 1: initialize: $\mathcal{S}_0 \leftarrow \varnothing, \boldsymbol{\alpha}_0 \leftarrow \varnothing$
 2: for $t = 1$ to $|\mathcal{T}|$ do
 3:     receive example $(\boldsymbol{x}, y)_t : \boldsymbol{x} \in \mathbb{R}^n, y \in \{-1, +1\}$
 4:     compute margin: $m_t(\boldsymbol{x}) = \sum_{\boldsymbol{s}_i \in \mathcal{S}_{t-1}} \alpha_i k(\boldsymbol{s}_i, \boldsymbol{x})$
 5:     if $\ell_t = \max\{0, 1 - y m_t(\boldsymbol{x})\} > 0$ then
 6:         $\tau_t \leftarrow \min\left\{C, \frac{\ell_t}{\|\boldsymbol{x}\|^2}\right\}$
 7:         $\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_{t-1} \cup \{\tau_t y\}, \mathcal{S}_t \leftarrow \mathcal{S}_{t-1} \cup \{\boldsymbol{x}\}$
 8:     else
 9:         $\boldsymbol{\alpha}_t \leftarrow \boldsymbol{\alpha}_{t-1}, \mathcal{S}_t \leftarrow \mathcal{S}_{t-1}$
10:     end if
11: end for
12: return $(\mathcal{S}_{|\mathcal{T}|}, \boldsymbol{\alpha}_{|\mathcal{T}|})$
```

2 Preliminaries

This section first introduces a passive-aggressive algorithm (Crammer et al., 2006), which we use as a base learner. We then explain fast methods of computing the polynomial kernel.

Each example $\boldsymbol{x}$ in a classification problem is represented by a *feature vector* whose element x_j is a value of a feature function, $f_j \in \mathcal{F}$. Here, we assume a binary feature function, $f_j(\boldsymbol{x}) \in \{0, 1\}$, which returns one if particular context data appear in the example. We say that feature f_j is *active* in example $\boldsymbol{x}$ when $x_j = f_j(\boldsymbol{x}) = 1$. We denote a binary feature vector, $\boldsymbol{x}$, as a set of active features $\boldsymbol{x} = \{f_j \mid f_j \in \mathcal{F}, f_j(\boldsymbol{x}) = 1\}$ for brevity; $f_j \in \boldsymbol{x}$ means that f_j is active in $\boldsymbol{x}$, and $|\boldsymbol{x}|$ represents the number of active features in $\boldsymbol{x}$.

2.1 Kernel Passive-Aggressive Algorithm

A passive-aggressive algorithm (PA) (Crammer et al., 2006) represents online learning that updates parameters for given labeled example $(\boldsymbol{x}, y)_t \in \mathcal{T}$ in each round t. We assume a binary label, $y \in \{-1, +1\}$, here for clarity. Algorithm 1 is a variant of PA (PA-I) that incorporates a kernel function, k. In round t, PA-I first computes a *(signed) margin* $m_t(\boldsymbol{x})$ of $\boldsymbol{x}$ by using the kernel function with support set $\mathcal{S}_{t-1}$ and coefficients $\boldsymbol{\alpha}_{t-1}$ (Line 4). PA-I then suffers a hinge-loss, $\ell_t = \max\{0, 1 - y m_t(\boldsymbol{x})\}$ (Line 5). If $\ell_t > 0$, PA-I adds $\boldsymbol{x}$ to $\mathcal{S}_{t-1}$ (Line 7). Hyperparameter C controls the aggressiveness of parameter updates.

The kernel function computes a dot product in $\mathbb{R}^{\mathcal{H}}$ space without mapping $\boldsymbol{x} \in \mathbb{R}^n$ to $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^{\mathcal{H}}$ ($k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}')$). We can implicitly consider (weighted) d or less order conjunctions of primitive features by using *polynomial kernel function* $k_d(\boldsymbol{s}, \boldsymbol{x}) = (\boldsymbol{s}^{\mathrm{T}}\boldsymbol{x} + 1)^d$. For example, given support vector $\boldsymbol{s} = (s_1, s_2)^{\mathrm{T}}$ and input example $\boldsymbol{x} = (x_1, x_2)^{\mathrm{T}}$, the second-order polynomial kernel returns $k_2(\boldsymbol{s}, \boldsymbol{x}) = (s_1x_1 + s_2x_2 + 1)^2 = 1 + 3s_1x_1 + 3s_2x_2 + 2s_1x_1s_2x_2$ ($\because s_i, x_i \in \{0, 1\}$). This function thus implies mapping $\boldsymbol{\phi}_2(\boldsymbol{x}) = (1, \sqrt{3}x_1, \sqrt{3}x_2, \sqrt{2}x_1x_2)^{\mathrm{T}}$.

Although online learning is generally efficient, the kernel spoils its efficiency (Dekel et al., 2008). This is because the kernel evaluation (Line 4) takes $\mathcal{O}(|\mathcal{S}_{t-1}||\boldsymbol{x}|)$ time and $|\mathcal{S}_{t-1}|$ increases as training continues. The learner thus takes the most amount of time in this margin computation.

2.2 Kernel Computation for Classification

This section explains fast, exact methods of computing the polynomial kernel, which are meant to test the trained model, $(\mathcal{S}, \boldsymbol{\alpha})$, and involve substantial computational cost in preparation.

2.2.1 Kernel Inverted

Kudo and Matsumoto (2003) proposed *polynomial kernel inverted* (PKI), which builds inverted indices $h(f_j) \equiv \{\boldsymbol{s} \mid \boldsymbol{s} \in \mathcal{S}, f_j \in \boldsymbol{s}\}$ from each feature f_j to support vector $\boldsymbol{s} \in \mathcal{S}$ to only consider support vector $\boldsymbol{s}$ relevant to given $\boldsymbol{x}$ such that $\boldsymbol{s}^{\mathrm{T}}\boldsymbol{x} \neq 0$. The time complexity of PKI is $\mathcal{O}(B \cdot |\boldsymbol{x}| + |\mathcal{S}|)$ where $B \equiv \frac{1}{|\boldsymbol{x}|}\sum_{f_j \in \boldsymbol{x}} |h(f_j)|$, which is smaller than $\mathcal{O}(|\mathcal{S}||\boldsymbol{x}|)$ if $\boldsymbol{x}$ has many rare features f_j such that $|h(f_j)| \ll |\mathcal{S}|$.

To the best of our knowledge, this is the only exact method that has been used to speed up margin computation in the context of kernel-based online learning (Okanohara and Tsujii, 2007).

2.2.2 Kernel Expansion

Isozaki and Kazawa (2002) and Kudo and Matsumoto (2003) proposed *kernel expansion*, which explicitly maps both support set $\mathcal{S}$ and given example $\boldsymbol{x} \in \mathbb{R}^n$ into $\mathbb{R}^{\mathcal{H}}$ by mapping $\boldsymbol{\phi}_d$ imposed by k_d:

$$m(\boldsymbol{x}) = \left(\sum_{\boldsymbol{s}_i \in \mathcal{S}} \alpha_i \boldsymbol{\phi}_d(\boldsymbol{s}_i)\right)^{\mathrm{T}} \boldsymbol{\phi}_d(\boldsymbol{x}) = \sum_{f_i \in \boldsymbol{x}^d} w_i,$$

where $\boldsymbol{x}^d \in \{0,1\}^{\mathcal{H}}$ is a binary feature vector in which $x_i^d = 1$ for $(\boldsymbol{\phi}_d(\boldsymbol{x}))_i \neq 0$, and $\boldsymbol{w}$ is a weight vector in the expanded feature space, $\mathcal{F}^d$. The weight vector $\boldsymbol{w}$ is computed from $\mathcal{S}$ and $\boldsymbol{\alpha}$:

$$\boldsymbol{w} = \sum_{\boldsymbol{s}_i \in \mathcal{S}} \alpha_i \sum_{k=0}^{d} c_d^k I_k(\boldsymbol{s}_i^d), \qquad (1)$$

where c_d^k is a squared coefficient of k-th order conjunctive features for d-th order polynomial kernel (*e.g.*, $c_2^0 = 1$, $c_2^1 = 3$, and $c_2^2 = 2$)[1] and $I_k(\boldsymbol{s}_i^d)$ is $\boldsymbol{s}_i^d \in \{0,1\}^{\mathcal{H}}$ whose dimensions other than those of k-th order conjunctive features are set to zero.

The time complexity of kernel expansion is $\mathcal{O}(|\boldsymbol{x}^d|)$ where $|\boldsymbol{x}^d| = \sum_{k=0}^{d} \binom{|\boldsymbol{x}|}{k} \propto |\boldsymbol{x}|^d$, which can be smaller than $\mathcal{O}(|\mathcal{S}||\boldsymbol{x}|)$ in usual NLP tasks ($|\boldsymbol{x}| \ll |\mathcal{S}|$ and $d \leq 4$).

2.2.3 Kernel Splitting

Since kernel expansion demands a huge memory volume to store the weight vector, $\boldsymbol{w}$, in $\mathbb{R}^{\mathcal{H}}$ ($\mathcal{H} = \sum_{k=0}^{d} \binom{|\mathcal{F}|}{k}$), Goldberg and Elhadad (2008) only explicitly considered conjunctions among features $f_C \in \mathcal{F}_C$ that commonly appear in support set $\mathcal{S}$, and handled the other conjunctive features relevant to rare features $f_R \in \mathcal{F} \setminus \mathcal{F}_C$ by using the polynomial kernel:

$$\begin{aligned} m(\boldsymbol{x}) &= m(\tilde{\boldsymbol{x}}) + m(\boldsymbol{x}) - m(\tilde{\boldsymbol{x}}) \\ &= \sum_{f_i \in \tilde{\boldsymbol{x}}^d} \tilde{w}_i + \sum_{\boldsymbol{s}_i \in \mathcal{S}_R} \alpha_i k'_d(\boldsymbol{s}_i, \boldsymbol{x}, \tilde{\boldsymbol{x}}), \quad (2) \end{aligned}$$

where $\tilde{\boldsymbol{x}}$ is $\boldsymbol{x}$ whose dimensions of rare features are set to zero, $\tilde{\boldsymbol{w}}$ is a weight vector computed with Eq. 1 for $\mathcal{F}_C^d$, and $k'_d(\boldsymbol{s}, \boldsymbol{x}, \tilde{\boldsymbol{x}})$ is defined as:

$$k'_d(\boldsymbol{s}, \boldsymbol{x}, \tilde{\boldsymbol{x}}) \equiv k_d(\boldsymbol{s}, \boldsymbol{x}) - k_d(\boldsymbol{s}, \tilde{\boldsymbol{x}}).$$

We can space-efficiently compute the first term of Eq. 2 since $|\tilde{\boldsymbol{w}}| \ll |\boldsymbol{w}|$, while we can quickly compute the second term of Eq. 2 since $k'_d(\boldsymbol{s}_i, \boldsymbol{x}, \tilde{\boldsymbol{x}}) = 0$ when $\boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{x} = \boldsymbol{s}_i^{\mathrm{T}} \tilde{\boldsymbol{x}}$; we only need to consider a small subset of the support set, $\mathcal{S}_R = \bigcup_{f_R \in \boldsymbol{x} \setminus \tilde{\boldsymbol{x}}} h(f_R)$, that has at least one of the rare features, f_R, appearing in $\boldsymbol{x} \setminus \tilde{\boldsymbol{x}}$ ($|\mathcal{S}_R| \ll |\mathcal{S}|$).

Counting the number of features examined, the time complexity of Eq. 2 is $\mathcal{O}(|\tilde{\boldsymbol{x}}^d| + |\mathcal{S}_R||\tilde{\boldsymbol{x}}|)$.

3 Algorithm

This section first describes the way kernel splitting is integrated into PA-I (Section 3.1). We then propose *kernel slicing* (Section 3.2), which enables us to reuse the temporal margins computed in the past rounds (Section 3.2.1) and to skip unnecessary margin computations (Section 3.2.2).

In what follows, we use PA-I as a base learner. Note that an analogous argument can be applied to other perceptron-like online learners with the additive weight update (Line 7 in Algorithm 1).

3.1 Base Learner with Kernel Splitting

A problem in integrating kernel splitting into the base learner presented in Algorithm 1 is how to determine $\mathcal{F}_C$, features among which we explicitly consider conjunctions, without knowing the final support set, $\mathcal{S}_{|\mathcal{T}|}$. We heuristically solve this by ranking feature f according to their frequency in the training data and by using the top-N frequent features in the training data as $\mathcal{F}_C$ ($= \{f \mid f \in \mathcal{F}, \textsc{rank}(f) \leq N\}$).[2] Since $\mathcal{S}_{|\mathcal{T}|}$ is a subset of the examples, this approximates the selection from $\mathcal{S}_{|\mathcal{T}|}$. We empirically demonstrate the validity of this approach in the experiments.

We then use $\mathcal{F}_C$ to construct a base learner with kernel splitting; we replace the kernel computation (Line 4 in Algorithm 1) with Eq. 2 where $(\mathcal{S}, \boldsymbol{\alpha}) = (\mathcal{S}_{t-1}, \boldsymbol{\alpha}_{t-1})$. To compute $m_t(\tilde{\boldsymbol{x}})$ by using kernel expansion, we need to additionally maintain the weight vector $\tilde{\boldsymbol{w}}$ for the conjunctions of common features that appear in $\mathcal{S}_{t-1}$.

The additive parameter update of PA-I enables us to keep $\tilde{\boldsymbol{w}}$ to correspond to $(\mathcal{S}_{t-1}, \boldsymbol{\alpha}_{t-1})$. When we add $\boldsymbol{x}$ to support set $\mathcal{S}_{t-1}$ (Line 7 in Algorithm 1), we also update $\tilde{\boldsymbol{w}}$ with Eq. 1:

$$\tilde{\boldsymbol{w}} \leftarrow \tilde{\boldsymbol{w}} + \tau_t y \sum_{k=0}^{d} c_d^k I_k(\tilde{\boldsymbol{x}}^d).$$

Following (Kudo and Matsumoto, 2003), we use a trie (hereafter, *weight trie*) to maintain conjunctive features. Each edge in the weight trie is labeled with a primitive feature, while each path

[1] Following Lemma 1 in Kudo and Matsumoto (2003), $c_d^k = \sum_{l=k}^{d} \binom{d}{l} \left(\sum_{m=0}^{k} (-1)^{k-m} \cdot m^l \binom{k}{m}\right)$.

[2] The overhead of counting features is negligible compared to the total training time. If we want to run the learner in a purely online manner, we can alternatively choose first N features that appear in the processed examples as $\mathcal{F}_C$.

represents a conjunctive feature that combines all the primitive features on the path. The weights of conjunctive features are retrieved by traversing nodes in the trie. We carry out an analogous traversal in updating the parameters of conjunctive features, while registering a new conjunctive feature by adding an edge to the trie.

The base learner with kernel splitting combines the virtues of linear training and kernel-based training. It reduces to linear training when we increase N to $|\mathcal{F}|$, while it reduces to kernel-based training when we decrease N to 0. The output is support set $\mathcal{S}_{|\mathcal{T}|}$ and coefficients $\boldsymbol{\alpha}_{|\mathcal{T}|}$ (optionally, $\tilde{\boldsymbol{w}}$), to which the efficient classification techniques discussed in Section 2.2 and the one proposed by Yoshinaga and Kitsuregawa (2009) can be applied.

Note on weight trie construction The time and space efficiency of this learner strongly depends on the way the weight trie is constructed. We need to address two practical issues that greatly affect efficiency. First, we traverse the trie from the rarest feature that constitutes a conjunctive feature. This rare-to-frequent mining helps us to avoid enumerating higher-order conjunctive features that have not been registered in the trie, when computing margin. Second, we use $\text{RANK}(f)$ encoded into a $\lceil \log_{128} \text{RANK}(f) \rceil$-byte string by using variable-byte coding (Williams and Zobel, 1999) as f's representation in the trie. This encoding reduces the trie size, since features with small $\text{RANK}(f)$ will appear frequently in the trie.

3.2 Base Learner with Kernel Slicing

Although a base learner with kernel splitting can enjoy the merits of linear and kernel-based training, it can simultaneously suffer from their demerits. Because the training takes polynomial time in the number of common features in $\boldsymbol{x}$ ($|\tilde{\boldsymbol{x}}^d| = \sum_{k=0}^{d} \binom{|\tilde{\boldsymbol{x}}|}{k} \propto |\tilde{\boldsymbol{x}}|^d$) at each round, we need to set N to a smaller value when we take higher-order conjunctive features into consideration. However, since the margin computation takes linear time in the number of support vectors $|\mathcal{S}_R|$ relevant to rare features $f_R \in \mathcal{F} \setminus \mathcal{F}_C$, we need to set N to a larger value when we handle a larger number of training examples. The training thereby slows down when we train a classifier with high-order conjunctive features and a large number of training examples.

We then attempt to improve the scalability of the training by exploiting a characteristic of labeled data in NLP. Because examples in NLP tasks are likely to be redundant (Yoshinaga and Kitsuregawa, 2009), the learner computes margins of examples that have many features in common. If we can reuse the 'temporal' margins of partial feature vectors computed in past rounds, this will speed up the computation of margins.

We propose *kernel slicing*, which generalizes kernel splitting in a purely feature-wise manner and enables us to reuse the temporal partial margins. Starting from the most frequent feature f_1 in $\boldsymbol{x}$ ($f_1 = \text{argmin}_{f \in \boldsymbol{x}} \text{RANK}(f)$), we incrementally compute $m_t(\boldsymbol{x})$ by accumulating a *partial margin*, $m_t^j(\boldsymbol{x}) \equiv m_t(\boldsymbol{x}_j) - m_t(\boldsymbol{x}_{j-1})$, when we add the j-th frequent feature f_j in $\boldsymbol{x}$:

$$m_t(\boldsymbol{x}) = m_t^0 + \sum_{j=1}^{|\boldsymbol{x}|} m_t^j(\boldsymbol{x}), \tag{3}$$

where $m_t^0 = \sum_{\boldsymbol{s}_i \in \mathcal{S}_{t-1}} \alpha_i k_d(\boldsymbol{s}_i, \varnothing) = \sum_i \alpha_i$, and $\boldsymbol{x}_j$ has the j most frequent features in $\boldsymbol{x}$ ($\boldsymbol{x}_0 = \varnothing$, $\boldsymbol{x}_j = \bigsqcup_{k=0}^{j-1} \{\text{argmin}_{f \in \boldsymbol{x} \setminus \boldsymbol{x}_k} \text{RANK}(f)\}$).

Partial margin $m_t^j(\boldsymbol{x})$ can be computed by using the polynomial kernel:

$$m_t^j(\boldsymbol{x}) = \sum_{\boldsymbol{s}_i \in \mathcal{S}_{t-1}} \alpha_i k'_d(\boldsymbol{s}_i, \boldsymbol{x}_j, \boldsymbol{x}_{j-1}), \tag{4}$$

or by using kernel expansion:

$$m_t^j(\boldsymbol{x}) = \sum_{f_i \in \boldsymbol{x}_j^d \setminus \boldsymbol{x}_{j-1}^d} \tilde{w}_i. \tag{5}$$

Kernel splitting is a special case of kernel slicing, which uses Eq. 5 for $f_j \in \mathcal{F}_C$ and Eq. 4 for $f_j \in \mathcal{F} \setminus \mathcal{F}_C$.

3.2.1 Reuse of Temporal Partial Margins

We can speed up both Eqs. 4 and 5 by reusing a temporal partial margin, $\delta_{t'}^j = m_{t'}^j(\boldsymbol{x})$ that had been computed in past round $t'(< t)$:

$$m_t^j(\boldsymbol{x}) = \delta_{t'}^j + \sum_{\boldsymbol{s}_i \in \mathcal{S}_j} \alpha_i k'_d(\boldsymbol{s}_i, \boldsymbol{x}_j, \boldsymbol{x}_{j-1}), \tag{6}$$

where $\mathcal{S}_j = \{\boldsymbol{s} \mid \boldsymbol{s} \in \mathcal{S}_{t-1} \setminus \mathcal{S}_{t'-1}, f_j \in \boldsymbol{s}\}$.

Algorithm 2 Kernel Slicing

```
INPUT: $\boldsymbol{x} \in 2^{\mathcal{F}}, \mathcal{S}_{t-1}, \boldsymbol{\alpha}_{t-1}, \mathcal{F}_C \subseteq \mathcal{F}, \boldsymbol{\delta} : 2^{\mathcal{F}} \mapsto \mathbb{N} \times \mathbb{R}$
OUTPUT: $m_t(\boldsymbol{x})$
 1: initialize: $\boldsymbol{x}_0 \leftarrow \varnothing, j \leftarrow 1, m_t(\boldsymbol{x}) \leftarrow m_t^0$
 2: repeat
 3:     $\boldsymbol{x}_j \leftarrow \boldsymbol{x}_{j-1} \sqcup \{\text{argmin}_{f \in \boldsymbol{x} \setminus \boldsymbol{x}_{j-1}} \text{RANK}(f)\}$
 4:     retrieve partial margin: $(t', \delta^j_{t'}) \leftarrow \boldsymbol{\delta}(\boldsymbol{x}_j)$
 5:     if $f_j \in \mathcal{F} \setminus \mathcal{F}_C$ or Eq. 7 is true then
 6:         compute $m^j_t(\boldsymbol{x})$ using Eq. 6 with $\delta^j_{t'}$
 7:     else
 8:         compute $m^j_t(\boldsymbol{x})$ using Eq. 5
 9:     end if
10:     update partial margin: $\boldsymbol{\delta}(\boldsymbol{x}_j) \leftarrow (t, m^j_t(\boldsymbol{x}))$
11:     $m_t(\boldsymbol{x}) \leftarrow m_t(\boldsymbol{x}) + m^j_t(\boldsymbol{x})$
12: until $\boldsymbol{x}_j \neq \boldsymbol{x}$
13: return $m_t(\boldsymbol{x})$
```

Eq. 6 is faster than Eq. 4,[3] and can even be faster than Eq. 5.[4] When $\text{RANK}(f_j)$ is high, $\boldsymbol{x}_j$ appears frequently in the training examples and $|\mathcal{S}_j|$ becomes small since t' will be close to t. When $\text{RANK}(f_j)$ is low, $\boldsymbol{x}_j$ rarely appears in the training examples but we can still expect $|\mathcal{S}_j|$ to be small since the number of support vectors in $\mathcal{S}_{t-1} \backslash \mathcal{S}_{t'-1}$ that have rare feature f_j will be small.

To compute Eq. 3, we now have the choice to choose Eq. 5 or 6 for $f_j \in \mathcal{F}_C$. Counting the number of features to be examined in computing $m^j_t(\boldsymbol{x})$, we have the following criteria to determine whether we can use Eq. 6 instead of Eq. 5:

$$1 + |\mathcal{S}_j||\boldsymbol{x}_{j-1}| \leq |\boldsymbol{x}^d_j \setminus \boldsymbol{x}^d_{j-1}| = \sum_{k=1}^{d} \binom{j-1}{k-1},$$

where the left- and right-hand sides indicate the number of features examined in Eq. 6 for the former and Eq. 5 for the latter. Expanding the right-hand side for $d = 2, 3$ and dividing both sides with $|\boldsymbol{x}_{j-1}| = j - 1$, we have:

$$|\mathcal{S}_j| \leq \left\{ \begin{array}{ll} 1 & (d = 2) \\ \frac{j}{2} & (d = 3) \end{array} \right. . \qquad (7)$$

If this condition is met after retrieving the temporal partial margin, $\delta^j_{t'}$, we can compute partial margin $m^j_t(\boldsymbol{x})$ with Eq. 6. This analysis reveals that we can expect little speed-up for the second-order polynomial kernel; we will only use Eq. 6 with third or higher-order polynomial kernel.

Algorithm 2 summarizes the margin computation with kernel slicing. It processes each feature $f_j \in \boldsymbol{x}$ in frequent-to-rare order, and accumulates partial margin $m^j_t(\boldsymbol{x})$ to have $m_t(\boldsymbol{x})$. Intuitively speaking, when the algorithm uses the partial margin, it only considers support vectors on each feature that have been added since the last evaluation of the partial feature vector, to avoid the repetition in kernel evaluation as much as possible.

3.2.2 Termination of Margin Computation

Kernel slicing enables another optimization that exploits a characteristic of online learning. Because we need an exact margin, $m_t(\boldsymbol{x})$, only when hinge-loss $\ell_t = 1 - y m_t(\boldsymbol{x})$ is positive, we can finish margin computation as soon as we find that the lower-bound of $y m_t(\boldsymbol{x})$ is larger than one.

When $y m_t(\boldsymbol{x})$ is larger than one after processing feature f_j in Eq. 3, we quickly examine whether this will hold even after we process the remaining features. We can compute a possible range of partial margin $m^k_t(\boldsymbol{x})$ with Eq. 4, having the upper- and lower-bounds, $\hat{k}'_d$ and $\check{k}'_d$, of $k'_d(\boldsymbol{s}_i, \boldsymbol{x}_k, \boldsymbol{x}_{k-1})$ $(= k_d(\boldsymbol{s}_i, \boldsymbol{x}_k) - k_d(\boldsymbol{s}_i, \boldsymbol{x}_{k-1}))$:

$$m^k_t(\boldsymbol{x}) \leq \hat{k}'_d \sum_{\boldsymbol{s}_i \in \mathcal{S}^+_k} \alpha_i + \check{k}'_d \sum_{\boldsymbol{s}_i \in \mathcal{S}^-_k} \alpha_i \qquad (8)$$

$$m^k_t(\boldsymbol{x}) \geq \check{k}'_d \sum_{\boldsymbol{s}_i \in \mathcal{S}^+_k} \alpha_i + \hat{k}'_d \sum_{\boldsymbol{s}_i \in \mathcal{S}^-_k} \alpha_i, \qquad (9)$$

where $\mathcal{S}^+_k = \{\boldsymbol{s}_i \,|\, \boldsymbol{s}_i \in \mathcal{S}_{t-1}, f_k \in \boldsymbol{s}_i, \alpha_i > 0\}$, $\mathcal{S}^-_k = \{\boldsymbol{s}_i \,|\, \boldsymbol{s}_i \in \mathcal{S}_{t-1}, f_k \in \boldsymbol{s}_i, \alpha_i < 0\}$, $\hat{k}'_d = (k+1)^d - k^d$ and $\check{k}'_d = 2^d - 1$ ($\because 0 \leq \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{x}_{k-1} \leq |\boldsymbol{x}_{k-1}| = k - 1$, $\boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{x}_k = \boldsymbol{s}_i^{\mathrm{T}} \boldsymbol{x}_{k-1} + 1$ for all $\boldsymbol{s}_i \in \mathcal{S}^+_k \cup \mathcal{S}^-_k$).

We accumulate Eqs. 8 and 9 from rare to frequent features, and use the intermediate results to estimate the possible range of $m_t(\boldsymbol{x})$ before Line 3 in Algorithm 2. If the lower bound of $y m_t(\boldsymbol{x})$ turns out to be larger than one, we terminate the computation of $m_t(\boldsymbol{x})$.

As training continues, the model becomes discriminative and given $\boldsymbol{x}$ is likely to have a larger margin. The impact of this termination will increase as the amount of training data expands.

[3] When a margin of $\boldsymbol{x}_j$ has not been computed, we regard $t' = 0$ and $\delta^j_{t'} = 0$, which reduces Eq. 6 to Eq. 4.

[4] We associate partial margins with partial feature sequences whose features are sorted by frequent-to-rare order, and store them in a trie (*partial margin trie*). This enables us to retrieve partial margin $\delta^j_{t'}$ for given $\boldsymbol{x}_j$ in $\mathcal{O}(1)$ time.

4 Evaluation

We evaluated the proposed method in two NLP tasks: dependency parsing (Sassano, 2004) and hyponymy-relation extraction (Sumida et al., 2008). We used labeled data included in open-source softwares to promote the reproducibility of our results.[5] All the experiments were conducted on a server with an Intel® Xeon™ 3.2 GHz CPU. We used a double-array trie (Aoe, 1989; Yata et al., 2009) as an implementation of the weight trie and the partial margin trie.

4.1 Task Descriptions

Japanese Dependency Parsing A parser inputs a sentence segmented by a *bunsetsu* (base phrase in Japanese), and selects a particular pair of bunsetsus (dependent and head candidates); the classifier then outputs label $y = +1$ (dependent) or -1 (independent) for the pair. The features consist of the surface form, POS, POS-subcategory and the inflection form of each bunsetsu, and surrounding contexts such as the positional distance, punctuations and brackets. See (Yoshinaga and Kitsuregawa, 2009) for details on the features.

Hyponymy-Relation Extraction A hyponymy relation extractor (Sumida et al., 2008) first extracts a pair of entities from hierarchical listing structures in Wikipedia articles (hypernym and hyponym candidates); a classifier then outputs label $y = +1$ (correct) or -1 (incorrect) for the pair. The features include a surface form, morphemes, POS and the listing type for each entity, and surrounding contexts such as the hierarchical distance between the entities. See (Sumida et al., 2008) for details on the features.

4.2 Settings

Table 1 summarizes the training data for the two tasks. The examples for the Japanese dependency parsing task were generated for a transition-based parser (Sassano, 2004) from a standard data set.[6] We used the dependency accuracy of the parser as model accuracy in this task. In the hyponymy-relation extraction task, we randomly chosen two sets of 10,000 examples from the labeled data for development and testing, and used the remaining examples for training. Note that the number of active features, $|\mathcal{F}^d|$, dramatically grows when we consider higher-order conjunctive features.

DATA SET	DEP	REL
$\lvert\mathcal{T}\rvert$	296,776	201,664
$(y = +1)$	150,064	152,199
$(y = -1)$	146,712	49,465
Ave. of $\lvert\boldsymbol{x}\rvert$	27.6	15.4
Ave. of $\lvert\boldsymbol{x}^2\rvert$	396.1	136.9
Ave. of $\lvert\boldsymbol{x}^3\rvert$	3558.3	798.7
$\lvert\mathcal{F}\rvert$	64,493	306,036
$\lvert\mathcal{F}^2\rvert$	3,093,768	6,688,886
$\lvert\mathcal{F}^3\rvert$	58,361,669	64,249,234

Table 1: Training data for dependency parsing (DEP) and hyponymy-relation extraction (REL).

We compared the proposed method, PA-I SL (Algorithm 1 with Algorithm 2), to PA-I KERNEL (Algorithm 1 with PKI; Okanohara and Tsujii (2007)), PA-I KE (Algorithm 1 with kernel expansion; viz., kernel splitting with $N = |\mathcal{F}|$), SVM (batch training of support vector machines),[7] and ℓ_1-LLM (stochastic gradient descent training of the ℓ_1-regularized log-linear model: Tsuruoka et al. (2009)). We refer to PA-I SL that does not reuse temporal partial margins as PA-I SL*. To demonstrate the impact of conjunctive features on model accuracy, we also trained PA-I without conjunctive features. The number of iterations in PA-I was set to 20, and the parameters of PA-I were averaged in an efficient manner (Daumé III, 2006). We explicitly considered conjunctions among top-N ($N = 125 \times 2^n; n \geq 0$) features in PA-I SL and PA-I SL*. The hyperparameters were tuned to maximize accuracy on the development set.

4.3 Results

Tables 2 and 3 list the experimental results for the two tasks (due to space limitations, Tables 2 and 3 list PA-I SL with parameter N that achieved the fastest speed). The accuracy of the models trained with the proposed method was better than ℓ_1-LLMs and was comparable to SVMs. The infe-

[5] The labeled data for dependency parsing is available from: http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/pecco/, and the labeled data for hyponymy-relation extraction is available from: http://nlpwww.nict.go.jp/hyponymy/.

[6] Kyoto Text Corpus Version 4.0: http://nlp.kuee.kyoto-u.ac.jp/nl-resource/corpus-e.html.

[7] http://chasen.org/~taku/software/TinySVM/

METHOD	d	ACC.	TIME	MEMORY
PA-I	1	88.56%	3s	55MB
ℓ_1-LLM	2	90.55%	340s	1656MB
SVM	2	90.76%	29863s	245MB
PA-I KERNEL	2	90.68%	8361s	84MB
PA-I KE	2	90.67%	41s	155MB
PA-I SL$^*_{N=4000}$	2	90.71%	**33s**	95MB
ℓ_1-LLM	3	90.76%	4057s	21,499MB
SVM	3	90.93%	25912s	243MB
PA-I KERNEL	3	90.90%	8704s	83MB
PA-I KE	3	90.90%	465s	993MB
PA-I SL$_{N=250}$	3	90.89%	**262s**	175MB

Table 2: Training time for classifiers used in dependency parsing task.

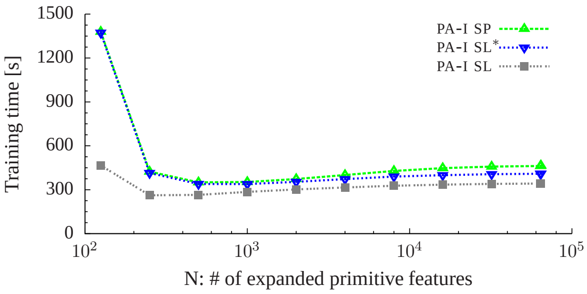


Figure 1: Training time for PA-I variants as a function of the number of expanded primitive features in dependency parsing task ($d = 3$).

rior accuracy of PA-I ($d = 1$) confirmed the necessity of conjunctive features in these tasks. The minor difference among the model accuracy of the three PA-I variants was due to rounding errors.

PA-I SL was the fastest of the training methods with the same feature set, and its space efficiency was comparable to the kernel-based learners. PA-I SL could reduce the memory footprint from 993MB[8] to 175MB for $d = 3$ in the dependency parsing task, while speeding up training.

Although linear training (ℓ_1-LLM and PA-I KE) dramatically slowed down when we took higher-order conjunctive features into account, kernel slicing alleviated deterioration in speed. Especially in the hyponymy-relation extraction task, PA-I SL took almost the same time regardless of the order of conjunctive features.

METHOD	d	ACC.	TIME	MEMORY
PA-I	1	91.75%	2s	28MB
ℓ_1-LLM	2	92.67%	136s	1683MB
SVM	2	92.85%	12306s	139MB
PA-I KERNEL	2	92.91%	1251s	54MB
PA-I KE	2	92.96%	27s	143MB
PA-I SL$^*_{N=8000}$	2	92.88%	**17s**	77MB
ℓ_1-LLM	3	92.86%	779s	14,089MB
SVM	3	93.09%	17354s	140MB
PA-I KERNEL	3	93.14%	1074s	49MB
PA-I KE	3	93.11%	103s	751MB
PA-I SL$_{N=125}$	3	93.05%	**17s**	131MB

Table 3: Training time for classifiers used in hyponymy-relation extraction task.

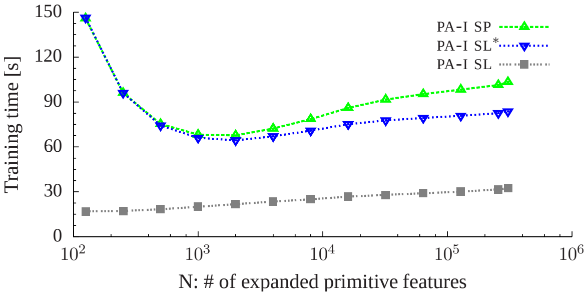


Figure 2: Training time for PA-I variants as a function of the number of expanded primitive features in hyponymy-relation extraction task ($d = 3$).

Figures 1 and 2 plot the trade-off between the number of expanded primitive features and training time with PA-I variants ($d = 3$) in the two tasks. Here, PA-I SP is PA-I with kernel slicing without the techniques described in Sections 3.2.1 and 3.2.2, viz., kernel splitting. The early termination of margin computation reduces the training time when N is large. The reuse of temporal margins makes the training time stable regardless of parameter N. This suggests a simple, effective strategy for calibrating N; we start the training with $N = |\mathcal{F}|$, and when the learner reaches the allowed memory size, we shrink N to $N/2$ by pruning sub-trees rooted by rarer features with $\text{RANK}(f) > N/2$ in the weight trie.

Figures 3 and 4 plot training time with PA-I variants ($d = 3$) for the two tasks as a function of the training data size. PA-I SP inherited the demerit of PA-I KERNEL which takes quadratic time in the number of examples, while PA-I SL took almost linear time in the number of examples.

[8] ℓ_1-LLM took much more memory than PA-I KE mainly because ℓ_1-LLM expands conjunctive features in the examples prior to training, while PA-I KE expands conjunctive features in each example on the fly during training. Interested readers may refer to (Chang et al., 2010) for this issue.

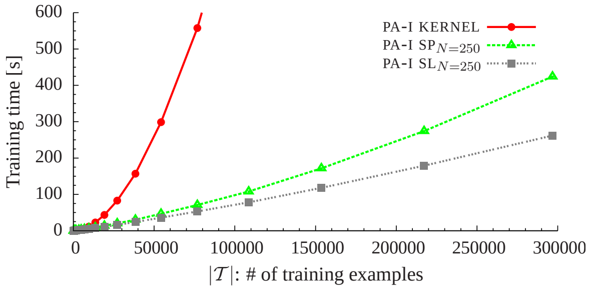


Figure 3: Training time for PA-I variants as a function of the number of training examples in dependency parsing task ($d = 3$).

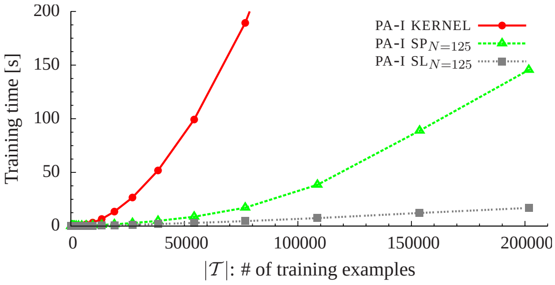


Figure 4: Training time for PA-I variants as a function of the number of training examples in hyponymy-relation extraction task ($d = 3$).

5 Related Work

There are several methods that learn 'simpler' models with fewer variables (features or support vectors), to ensure scalability in training.

Researchers have employed feature selection to assure space-efficiency in linear training. Wu et al. (2007) used frequent-pattern mining to select effective conjunctive features prior to training. Okanohara and Tsujii (2009) revised grafting for ℓ_1-LLM (Perkins et al., 2003) to prune useless conjunctive features during training. Iwakura and Okamoto (2008) proposed a boosting-based method that repeats the learning of rules represented by feature conjunctions. These methods, however, require us to tune the hyperparameter to trade model accuracy and the number of conjunctive features (memory footprint and training time); note that an accurate model may need many conjunctive features (in the hyponymy-relation extraction task, ℓ_1-LLM needed 15,828,122 features to obtain the best accuracy, 92.86%). Our method, on the other hand, takes all conjunctive features into consideration regardless of parameter N.

Dekel et al. (2008) and Cavallanti et al. (2007) improved the scalability of the (kernel) perceptron, by exploiting redundancy in the training data to bound the size of the support set to given threshold B ($\geq |\mathcal{S}_t|$). However, Orabona et al. (2009) reported that the models trained with these methods were just as accurate as a naive method that ceases training when $|\mathcal{S}_t|$ reaches the same threshold, B. They then proposed budget online learning based on PA-I, and it reduced the size of the support set to a tenth with a tolerable loss of accuracy. Their method, however, requires $\mathcal{O}(|\mathcal{S}_{t-1}|^2)$ time in updating the parameters in round t, which disables efficient training. We have proposed an orthogonal approach that exploits the data redundancy in evaluating the kernel to train the same model as the base learner.

6 Conclusion

In this paper, we proposed online learning with kernel slicing, aiming at resolving the space-time trade-off in training a classifier with many conjunctive features. The kernel slicing generalizes kernel splitting (Goldberg and Elhadad, 2008) in a purely feature-wise manner, to truly combine the merits of linear and kernel-based training. To improve the scalability of the training with redundant data in NLP, we reuse the temporal partial margins computed in past rounds and terminate unnecessary margin computations. Experiments on dependency parsing and hyponymy-relation extraction demonstrated that our method could train a classifier orders of magnitude faster than kernel-based learners, while retaining its space efficiency.

We will evaluate our method with ample labeled data obtained by the semi-supervised methods. The implementation of the proposed algorithm for kernel-based online learners is available from http://www.tkl.iis.u-tokyo.ac.jp/˜ynaga/.

Acknowledgment We thank Susumu Yata for providing us practical lessons on the double-array trie, and thank Yoshimasa Tsuruoka for making his ℓ_1-LLM code available to us. We are also indebted to Nobuhiro Kaji and the anonymous reviewers for their valuable comments.

References

Ando, Rie Kubota and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Aoe, Jun'ichi. 1989. An efficient digital search algorithm by using a double-array structure. *IEEE Transactions on Software Engineering*, 15(9):1066–1077.

Bellare, Kedar, Partha Pratim Talukdar, Giridhar Kumaran, Fernando Pereira, Mark Liberman, Andrew McCallum, and Mark Dredze. 2007. Lightly-supervised attribute extraction. In *Proc. NIPS 2007 Workshop on Machine Learning for Web Search*.

Cavallanti, Giovanni, Nicolò Cesa-Bianchi, and Claudio Gentile. 2007. Tracking the best hyperplane with a simple budget perceptron. *Machine Learning*, 69(2-3):143–167.

Chang, Yin-Wen, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. 2010. Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research*, 11:1471–1490.

Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Daumé III, Hal. 2006. *Practical Structured Learning Techniques for Natural Language Processing*. Ph.D. thesis, University of Southern California.

Daumé III, Hal. 2008. Cross-task knowledge-constrained self training. In *Proc. EMNLP 2008*, pages 680–688.

Dekel, Ofer, Shai Shalev-Shwartz, and Yoram Singer. 2008. The forgetron: A kernel-based perceptron on a budget. *SIAM Journal on Computing*, 37(5):1342–1372.

Freund, Yoav and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Goldberg, Yoav and Michael Elhadad. 2008. splitSVM: fast, space-efficient, non-heuristic, polynomial kernel computation for NLP applications. In *Proc. ACL-08: HLT, Short Papers*, pages 237–240.

Isozaki, Hideki and Hideto Kazawa. 2002. Efficient support vector classifiers for named entity recognition. In *Proc. COLING 2002*, pages 1–7.

Iwakura, Tomoya and Seishi Okamoto. 2008. A fast boosting-based learner for feature-rich tagging and chunking. In *Proc. CoNLL 2008*, pages 17–24.

Kudo, Taku and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proc. ACL 2003*, pages 24–31.

Liang, Percy, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proc. ICML 2008*, pages 592–599.

Okanohara, Daisuke and Jun'ichi Tsujii. 2007. A discriminative language model with pseudo-negative samples. In *Proc. ACL 2007*, pages 73–80.

Okanohara, Daisuke and Jun'ichi Tsujii. 2009. Learning combination features with L_1 regularization. In *Proc. NAACL HLT 2009, Short Papers*, pages 97–100.

Orabona, Francesco, Joseph Keshet, and Barbara Caputo. 2009. Bounded kernel-based online learning. *Journal of Machine Learning Research*, 10:2643–2666.

Perkins, Simon, Kevin Lacker, and James Theiler. 2003. Grafting: fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356.

Sassano, Manabu. 2004. Linear-time dependency analysis for Japanese. In *Proc. COLING 2004*, pages 8–14.

Sumida, Asuka, Naoki Yoshinaga, and Kentaro Torisawa. 2008. Boosting precision and recall of hyponymy relation acquisition from hierarchical layouts in Wikipedia. In *Proc. LREC 2008*, pages 2462–2469.

Tsuruoka, Yoshimasa, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for L1-regularized log-linear models with cumulative penalty. In *Proc. ACL-IJCNLP 2009*, pages 477–485.

Williams, Hugh E. and Justin Zobel. 1999. Compressing integers for fast file access. *The Computer Journal*, 42(3):193–201.

Wu, Yu-Chieh, Jie-Chi Yang, and Yue-Shi Lee. 2007. An approximate approach for training polynomial kernel SVMs in linear time. In *Proc. ACL 2007, Interactive Poster and Demonstration Sessions*, pages 65–68.

Yata, Susumu, Masahiro Tamura, Kazuhiro Morita, Masao Fuketa, and Jun'ichi Aoe. 2009. Sequential insertions and performance evaluations for double-arrays. In *Proc. the 71st National Convention of IPSJ*, pages 1263–1264. (In Japanese).

Yoshinaga, Naoki and Masaru Kitsuregawa. 2009. Polynomial to linear: efficient classification with conjunctive features. In *Proc. EMNLP 2009*, pages 1542–1551.

Discriminative Training for Near-Synonym Substitution

Liang-Chih Yu[1], Hsiu-Min Shih[2], Yu-Ling Lai[2], Jui-Feng Yeh[3] and Chung-Hsien Wu[4]
[1]Department of Information Management, Yuan Ze University
[2]Department of Mathematics, National Chung Cheng University
[3]Department of CSIE, National Chia-Yi University
[4]Department of CSIE, National Cheng Kung University
Contact: lcyu@saturn.yzu.edu.tw

Abstract

Near-synonyms are useful knowledge resources for many natural language applications such as query expansion for information retrieval (IR) and paraphrasing for text generation. However, near-synonyms are not necessarily interchangeable in contexts due to their specific usage and syntactic constraints. Accordingly, it is worth to develop algorithms to verify whether near-synonyms do match the given contexts. In this paper, we consider the near-synonym substitution task as a classification task, where a classifier is trained for each near-synonym set to classify test examples into one of the near-synonyms in the set. We also propose the use of discriminative training to improve classifiers by distinguishing positive and negative features for each near-synonym. Experimental results show that the proposed method achieves higher accuracy than both pointwise mutual information (PMI) and n-gram-based methods that have been used in previous studies.

1 Introduction

Near-synonym sets represent groups of words with similar meaning, which are useful knowledge resources for many natural language applications. For instance, they can be used for query expansion in information retrieval (IR) (Moldovan and Mihalcea, 2000; Bhogal et al., 2007), where a query term can be expanded by its near-synonyms to improve the recall rate. They can also be used in an intelligent thesaurus that can automatically suggest alternative words to avoid repeating the same word in the composing of text when there are suitable alternatives in its synonym set (Inkpen and Hirst, 2006; Inkpen, 2007). These near-synonym sets can be derived from manually constructed dictionaries such as WordNet (called synsets) (Fellbaum, 1998), EuroWordNet (Rodríguez et al., 1998), or clusters derived using statistical approaches (Lin, 1998).

Although the words in a near-synonym set have similar meaning, they are not necessarily interchangeable in practical use due to their specific usage and collocational constraints. Pearce (2001) presented an example of collocational constraints for the context " ____ coffee". In the given near-synonym set {strong, powerful}, the word "strong" is more suitable than "powerful" to fill the gap, since "powerful coffee" is an anti-collocation. Inkpen (2007) also presented several examples of collocations (e.g. ghastly mistake) and anti-collocations (e.g. ghastly error). Yu *et al.* (2007) described an example of the context mismatch problem for the context " ____ under the bay" and the near-synonym set {bridge, overpass, viaduct, tunnel} that represents the meaning of a physical structure that connects separate places by traversing an obstacle. The original word (target word) in the given context is "tunnel", and cannot be substituted by the other words in the same set since all the substitutions are semantically implausible. Accordingly, it is worth to develop algorithms to verify whether near-synonyms do match the given contexts. Applications can benefit from this ability to provide more effective services. For instance, a writing support system can assist users to select an alternative word that best fits a given context from a list of near-synonyms.

In measuring the substitutability of words, the co-occurrence information between a target word

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1254–1262,
Beijing, August 2010

(the gap) and its context words is commonly used in statistical approaches. Edmonds (1997) built a lexical co-occurrence network from 1989 Wall Street Journal to determine the near-synonym that is most typical or expected in a given context. Inkpen (2007) used the pointwise mutual information (PMI) formula to select the best near-synonym that can fill the gap in a given context. The PMI scores for each candidate near-synonym are computed using a larger web corpus, the Waterloo terabyte corpus, which can alleviate the data sparseness problem encountered in Edmonds' approach. Following Inkpen's approach, Gardiner and Dras (2007) also used the PMI formula with a different corpus (the Web 1T 5-gram corpus) to explore whether near-synonyms differ in attitude.

Yu *et al.* (2007) presented a method to compute the substitution scores for each near-synonym based on n-gram frequencies obtained by querying Google. A statistical test is then applied to determine whether or not a target word can be substituted by its near-synonyms. The dataset used in their experiments are derived from the OntoNotes copus (Hovy et al., 2006; Pradhan et al., 2007), where each near-synonym set corresponds to a *sense pool* in OntoNotes. Another direction to the task of near-synonym substitution is to identify the senses of a target word and its near-synonyms using word sense disambiguation (WSD), comparing whether they were of the same sense (McCarthy, 2002; Dagan et al., 2006). Dagan *et al.* (2006) described that the use of WSD is an indirect approach since it requires the intermediate sense identification step, and thus presented a sense matching technique to address the task directly.

In this paper, we consider the near-synonym substitution task as a classification task, where a classifier is trained for each near-synonym set to classify test examples into one of the near-synonyms in the set. However, near-synonyms share more common context words (features) than semantically dissimilar words in nature. Such similar contexts may decrease classifiers' ability to discriminate among near-synonyms. Therefore, we propose the use of a supervised discriminative training technique (Ohler et al., 1999; Kuo and Lee, 2003; Zhou and He, 2009) to improve classifiers by distinguishing positive and negative features for each near-synonym. To our best knowledge, this is the first study that uses discriminative training for near-synonym or lexical substitution. The basic idea of discriminative training herein is to adjust feature weights according to the minimum classification error (MCE) criterion. The features that contribute to discriminating among near-synonyms will receive a greater positive weight, whereas the noisy features will be penalized and might receive a negative weight. This re-weighting scheme helps increase the separation of the correct class against its competing classes, thus improves the classification performance.

Sentence: This will make the _______ message easier to interpret.

Original word: error

Near-synonym set: {error, mistake, oversight}

Figure 1. Example of a near-synonym set and a sentence to be evaluated.

The proposed supervised discriminative training is compared with two unsupervised methods, the PMI-based (Inkpen, 2007) and n-gram-based (Yu *et al.*, 2007) methods. The goal of the evaluation is described as follows. Given a near-synonym set and a sentence with one of the near-synonyms in it, the near-synonym is deleted to form a gap in this sentence. Figure 1 shows an example. Each method is then applied to predict an answer (best near-synonym) that can fill the gap. The possible candidates are all the near-synonyms (including the original word) in the given set. Ideally, the correct answers should be provided by human experts. However, such data is usually unavailable, especially for a large set of test examples. Therefore, we follow Inkpen's experiments to consider the original word as the correct answer. The proposed methods can then be evaluated by examining whether they can restore the original word by filling the gap with the best near-synonym.

The rest of this work is organized as follows. Section 2 describes the PMI and n-gram-based methods for near-synonym substitution. Section 3 presents the discriminative training technique. Section 4 summarizes comparative results. Conclusions are finally drawn in Section 5.

2 Unsupervised Methods

2.1 PMI-based method

The mutual information can measure the co-occurrence strength between a near-synonym and the words in a given context. A higher mutual information score indicates that the near-synonym fits well in the given context, thus is more likely to be the correct answer. The pointwise mutual information (Church and Hanks, 1991) between two words *x* and *y* is defined as

$$PMI(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}, \tag{1}$$

where $P(x,y) = C(x,y)/N$ denotes the probability that x and y co-occur; $C(x,y)$ is the number of times *x* and *y* co-occur in the corpus, and *N* is the total number of words in the corpus. Similarly, $P(x) = C(x)/N$, where *C*(*x*) is the number of times *x* occurs in the corpus, and $P(y) = C(y)/N$, where *C*(*y*) is the number of times *y* occurs in the corpus. Therefore, (1) can be re-written as

$$PMI(x,y) = \log_2 \frac{C(x,y)\cdot N}{C(x)\cdot C(y)}. \tag{2}$$

Inkpen (2007) computed the PMI scores for each near-synonym using the Waterloo terabyte corpus and a context window of size 2*k* (*k*=2). Given a sentence *s* with a gap, $s = ...w_1...w_k \underline{\qquad} w_{k+1}...w_{2k}...$, the PMI score for a near-synonym NS_i to fill the gap is defined as

$$PMI(NS_j,s) = \sum\nolimits_{i=1}^{k} PMI(NS_j,w_i) + \sum\nolimits_{i=k+1}^{2k} PMI(NS_j,w_i). \tag{3}$$

The near-synonym with the highest score is considered as the answer. In this paper, we use the Web 1T 5-gram corpus to compute PMI scores, the same as in (Gardiner and Dras, 2007). The frequency counts *C*(·) are retrieved from this corpus in the same manner within the 5-gram boundary.

2.2 N-gram-based method

The n-grams can capture contiguous word associations in given contexts. Given a sentence with a gap, the n-gram scores for each near-synonym are computed as follows. First, all possible n-grams containing the gap are extracted from the sentence. Each near-synonym then fills the gap to compute a normalized frequency according to

$$Z(ngram^i_{NS_j}) = \frac{\log\left(C(ngram^i_{NS_j})+1\right)}{\log C(NS_j)}, \tag{4}$$

where $C(ngram^i_{NS_j})$ denotes the frequency of an n-gram containing a near-synonym, $C(NS_j)$ denotes the frequency of a near-synonym, and $Z(ngram^i_{NS_j})$ denotes the normalized frequency of an n-gram, which is used to reduce the effect of high frequencies on measuring n-gram scores. All of the above frequencies are retrieved from the Web 1T 5-gram corpus.

The n-gram score for a near-synonym to fill the gap is computed as

$$NGRAM(NS_j,s) = \frac{1}{R}\sum_{i=1}^{R} Z(ngram^i_{NS_j}), \tag{5}$$

where $NGRAM(NS_j,s)$ denotes the n-gram score of a near-synonym, which is computed by averaging the normalized frequencies of all the n-grams containing the near-synonym, and *R* is the total number of n-grams containing the near-synonym. Again, the near-synonym with the highest score is the proposed answer. We herein use the 4-gram frequencies to compute n-gram scores as Yu *et al.* (2007) reported that the use of 4-grams achieved higher performance than the use of bigrams and trigrams.

3 Discriminative Training

3.1 Classifier

The supervised classification technique can also be applied to for near-synonym substitution. Each near-synonym in a set corresponds to a class. The classifiers for each near-synonym set are built from the labeled training data, i.e., a collection of sentences containing the near-synonyms. Such training data is easy to obtain since it requires no human annotation. The training data used herein is collected by extracting the 5-grams containing the near-synonyms from the Web 1T 5-gram corpus. The features used for training are the words occurring in the context of the near-synonyms in the 5-grams.

	Example		
	1	2	3
$g_1(x,M)=\cos(x,m_1)$	0.9*	0.6*	0.8
$g_2(x,M)=\cos(x,m_2)$	0.3	0.5	0.3*
$g_3(x,M)=\cos(x,m_3)$	0.2	0.4	0.1
$\max_{j\neq k} g_i(x,M)=$	0.3	0.5	0.8
$d_k(x,M)=$	-0.6	-0.1	0.5
$l_k(x,M)=$ (γ=5)	0.047	0.378	0.924

Table 1. Examples of correct classification and misclassification. * denotes the scores of the correct class.

For each near-synonym set, an $F \times K$ feature-class matrix, denoted as *M*, is created for classification. The rows represent the *F* distinct words (features) and the columns represent the *K* near-synonyms (classes) in the same set. Each entry in the matrix, m_{ij}, represents a weight of word *i* respect to near-synonym *j*, which is computed as the number of times word *i* appears in the contexts of near-synonym *j* divided by the total number of context words of near-synonym *j*. This frequency-based weight can then be transformed into a probabilistic form, i.e., divided by the sum of the weights of word *i* respect to all near-synonyms. Each test sentence is also transformed into an *F*-dimensional feature vector. Let $x=[x_1,...,x_i,...,x_F]$ denote the feature vector of an input sentence. The classification is performed by computing the cosine similarity between *x* and the column vectors (near-synonyms) in the matrix, defined as

$$\begin{aligned} NS_{\hat{j}} &= \arg\max_j \cos(x,m_j) \\ &= \arg\max_j \frac{x \bullet m_j}{\|x\|\|m_j\|} \qquad (6) \\ &= \arg\max_j \frac{\sum_{i=1}^{F} x_i m_{ij}}{\sqrt{\sum_{i=1}^{F} x_i^{\,2}}\sqrt{\sum_{i=1}^{F} m_{ij}^{\,2}}}, \end{aligned}$$

where m_j is the *j*-th column vector in the matrix *M*. The near-synonym with the highest cosine similarity score, $NS_{\hat{j}}$, is the predicted class of the classifier.

3.2 Minimum classification error criterion

According to the decision rule of the classifier, a classification error will occur when the near-synonym with the highest cosine score is not the correct class. Table 1 shows some examples, where Example 3 is an example of misclassification. On the other hand, although Example 2 is a correct classification, it might be an ambiguous case to classifiers since the scores are close among classes. Therefore, if we can increase the separation of the correct class from its competing ones, then the classification performance can be improved accordingly. This can be accomplished by adjusting the feature weights of the matrix *M* that have direct influence on the computation of cosine scores. The discriminative training performs the adjustment in the training phase according to the minimum classification error criterion. The detailed steps are as follows.

Given an input vector *x*, the classifier computes the cosine scores between *x* and each class using (6). The discriminant function for a class can thus be defined as the cosine measure; that is,

$$g_j(x,M)=\cos(x,m_j). \qquad (7)$$

where *j* denotes a class in *K*. Since the correct class of each input vector is known in the training phase, we can determine whether or not the input vector is misclassified by comparing the discriminant function (cosine score) of the correct class against its competing classes. In the case of misclassification, the cosine score of the correct class will be smaller than the competing cosine scores. Let *k* be the correct class of *x*, the misclassification function can be defined as

$$d_k(x,M)=-\,g_k(x,M)\;+\;G_k(x,M), \qquad (8)$$

where $g_k(x,M)$ is the discriminant function for the correct class *k*, and $G_k(x,M)$ is the anti-discriminant function that represents the other $K-1$ competing classes, defined as

$$G_k(x,M)=\left[\frac{1}{K-1}\sum_{j\neq k} g_j(x,M)^{\eta}\right]^{\frac{1}{\eta}}, \qquad (9)$$

When $\eta=1$, the anti-discriminant function $G_k(x,M)$ is the average of all the competing cosine scores. With the increase of η, $G_k(x,M)$ is gradually dominated by the biggest competing class. In the extreme case, i.e., $\eta \to \infty$, the anti-discriminant function becomes

$$G_k(x,M) = \max_{j \neq k} g_j(x,M). \quad (10)$$

The misclassification function in (8) can thus be rewritten as

$$d_k(x,M) = -g_k(x,M) + \max_{j \neq k} g_j(x,M), \quad (11)$$

In this case, the classification error is determined by comparing the discriminant function of the correct class against that of the biggest competing class. Obviously, $d_k(x,M) > 0$ implies a classification error. For instance, in Example 3, the discriminant function for the correct class is $g_2(x,M) = 0.3$, and that of the biggest competing class is $\max(g_1(x,M), g_3(x,M)) = 0.8$, thus the classification error is $d_k(x,M) = 0.5$. On the other hand, the classification error will be a negative value for correct classifications, as shown in Example 1 and 2.

Intuitively, a greater classification error also results in a greater loss. We herein use the sigmoid function as the class loss function; that is,

$$l_k(x,M) = l(d_k) = \frac{1}{1+\exp^{-\gamma d_k}}, \quad (12)$$

where γ is a constant that controls the slope of the sigmoid function. The sigmoid function maps the values of classification error within the range of 0 to 1. For correct classifications, a greater separation of the correct class from the biggest competing class leads to a greater negative value of d_k, i.e., a smaller classification error, resulting in a smaller loss tends asymptotically to 0 (Example 1), whereas a moderate loss is yielded if the separation was close (Example 2). For the cases of misclassification, a greater separation leads to a greater positive value of d_k, i.e., a greater classification error, resulting in a greater loss tends asymptotically to 1 (Example 3). The overall loss of the entire training set can then be estimated as

$$L(M) = \frac{1}{Q}\sum_{k=1}^{K}\sum_{x \in C_k} l_k(x,M), \quad (13)$$

where C_k denotes the set of training vectors of class *k*, and *Q* is the total number of vectors in the training set. The goal now is to minimize the loss. According to the above discussions on the three examples, to minimize the loss is to minimize the classification error, and to improve the separation of the correct class against its competing classes. This can be accomplished by adjusting the feature weights of the matrix *M* to distinguish positive and negative features for each class. We herein adopt a gradient descent method such as the generalized probabilistic descent (GPD) algorithm (Katagiri et al., 1998) to update the matrix *M*. The detailed steps are as follows.

Let the feature weights of the matrix *M* be the parameter set to be adjusted. The adjustment is performed iteratively according to the following update formula.

$$M^{(t+1)} = M^{(t)} - \varepsilon_t \nabla l_k(x^{(t)}, M^{(t)}), \quad (14)$$

where *t* denotes the *t*-th iteration, ε_t is an adjustment step of a small positive real number, and $\nabla l_k(x^{(t)}, M^{(t)})$ is the gradient of the loss function, which is computed by the following two parts

$$\nabla l_k(x^{(t)}, M^{(t)}) = \frac{\partial l_k}{\partial d_k}\frac{\partial d_k(x^{(t)}, M^{(t)})}{\partial m_{ij}}, \quad (15)$$

where

$$\frac{\partial l_k}{\partial d_k} = \gamma l_k(d_k)(1 - l_k(d_k)), \quad (16)$$

and from (7), (8), and (9),

$$\frac{\partial d_k(x^{(t)}, M^{(t)})}{\partial m_{ij}} = \begin{cases} -x_i, & \text{if } j = k \\ x_i \dfrac{G_k(x^{(t)},M) g_j(x^{(t)},M)^{\eta-1}}{\sum_{j \neq k} g_j(x^{(t)},M)^{\eta}}, & \text{if } j \neq k \end{cases}, \quad (17)$$

where x_i is an element of the input vector *x*. By replacing $\nabla l_k(x_t, M_t)$ in (14) with the two parts in (15), at each iteration each feature weight m_{ij} in *M* is adjusted by

$$m_{ij}^{(t+1)} = \begin{cases} m_{ij}^{(t)} + \varepsilon_t \dfrac{\partial l_k}{\partial d_k} x_i, & \text{if } j = k \\ m_{ij}^{(t)} - \varepsilon_t \dfrac{\partial l_k}{\partial d_k} x_i \dfrac{G_k(x^{(t)},M) g_j(x^{(t)},M)^{\eta-1}}{\sum_{j \neq k} g_j(x^{(t)},M)^{\eta}}, & \text{if } j \neq k \end{cases}. \quad (18)$$

The weight x_i represents whether or not a dimension word occurs in an input sentence. A zero weight indicates that the dimension word does not occur in the input sentence, thus the corresponding dimension of each column vector will not be adjusted. On the contrary, the corresponding dimension of the column vector of the correct class ($j = k$) is adjusted by adding a value, while those of the competing classes ($j \neq k$) are adjusted by subtracting a value from them. After a sequence of adjustments over the training set, the positive and negative features can be distinguished by adjusting their weights that result in a greater positive or negative value for each of them. The separation of the correct class against its competing ones can thus be increased.

The weight adjustment in (18) is in proportion to the adjustment step ε_t and the slope of the sigmoid function $\partial l_k / \partial d_k$. The adjustment step ε_t can be determined empirically. As (16) shows, the slope $\partial l_k / \partial d_k$ converges asymptotically to zero as the classification error d_k approaches to a very large (or small) value. This leads to a small weight adjustment. For instance, the weight adjustment in Example 1 is small due to a small value of d_k, or, say, due to a large separation between the correct class and its competing ones. This is reasonable because classifiers often perform well in such cases. Similarly, the weight adjustment in Example 3 (misclassification) is also small due to a large value of d_k. A greater adjustment is not employed because such a large separation is difficult to be reduced significantly. Additionally, over-adjusting some features may introduce negative effects on other useful features in the matrix. Therefore, discriminative training is more effective on the cases with a moderate value of d_k, like Example 2. Such cases usually fall within the decision boundary and tend to be confusing to classifiers. Hence, improving the separation on such cases helps significantly improve the classification performance.

4 Experimental Results

4.1 Experiment setup

1) Data: The near-synonym sets used for experiments included the seven sets (Exp1) and the eleven sets (Exp2) used in the previous studies (Edmonds, 1997; Inkpen, 2007; Gardiner and Dras, 2007), as shown in Table 2. The Web 1T 5-gram corpus was used to build classifiers, where the corpus was randomly split into a training set, a development set, and a test set with an 8:1:1 ratio. For efficiency consideration, we randomly sampled up to 100 5-grams from the test set for each near-synonym. This sampling procedure was repeated five times for evaluation of the classifiers.

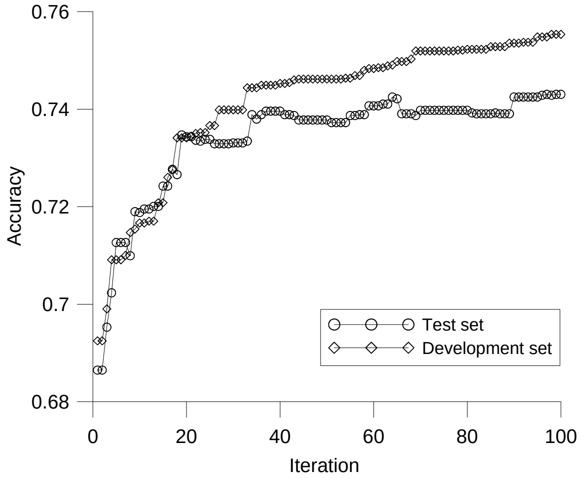


Figure 2. The change of classification accuracy during discriminative training.

2) Classifiers: The classifiers were implemented using PMI, n-grams, and discriminative training (DT) methods, respectively.

PMI: Given a near-synonym set and a test 5-gram with a gap, the PMI scores for each near-synonym were calculated using (3), where the size of the context window *k* was set to 2. The frequency counts between each near-synonym and its context words were retrieved from the training set.

NGRAM: For each test 5-gram with a gap, all possible 4-grams containing the gap were first extracted (excluding punctuation marks). The averaged 4-gram scores for each near-synonym were then calculated using (5). Again, the frequency counts of the 4-grams were retrieved from the training set.

DT: For each near-synonym set, the matrix *M* was built from the training set. Each 5-gram in the development set was taken as input to iteratively compute the cosine score, loss, classification error, respectively, and finally to adjust the feature weights of *M*. The parameters of DT included η for the anti-discriminative function, γ

No.	Near-synonym set	No. of cases	Accuracy (%)			
			NGRAM	PMI	COS	DT
1.	difficult, hard, tough	300	58.60	61.67	60.13	63.13
2.	error, mistake, oversight	300	68.47	78.33	77.20	79.20
3.	job, task, duty	300	68.93	70.40	74.00	75.67
4.	responsibility, burden, obligation, commitment	400	69.80	66.95	68.75	69.55
5.	material, stuff, substance	300	70.20	67.93	71.07	75.13
6.	give, provide, offer	300	58.87	66.47	64.13	68.27
7.	settle, resolve	200	69.30	68.10	77.10	84.10
	Exp1	**2,100**	**66.33**	**68.50**	**69.94**	**72.89**
1.	benefit, advantage, favor, gain, profit	500	71.44	69.88	69.44	71.36
2.	low, gush, pour, run, spout, spurt, squirt, stream	800	65.45	65.00	68.68	71.08
3.	deficient, inadequate, poor, unsatisfactory	400	65.65	69.40	70.35	74.35
4.	afraid, aghast, alarmed, anxious, apprehensive, fearful, frightened, scared, terror-stricken*	789	49.84	44.74	47.00	49.33
5.	disapproval, animadversion*, aspersion*, blame, criticism, reprehension*	300	72.80	79.47	80.00	82.53
6.	mistake, blooper, blunder, boner, contretemps*, error, faux pas*, goof, slip, solecism*	618	62.27	59.61	68.41	71.65
7.	alcoholic, boozer*, drunk, drunkard, lush, sot	433	64.90	80.65	77.88	84.34
8.	leave, abandon, desert, forsake	400	65.85	66.05	69.35	74.35
9.	opponent, adversary, antagonist, competitor, enemy, foe, rival	700	58.51	59.51	63.31	67.14
10.	thin, lean, scrawny, skinny, slender, slim, spare, svelte, willowy*, wiry	734	57.74	61.99	55.72	64.58
11.	lie, falsehood, fib, prevarication*, rationalization, untruth	425	57.55	63.58	69.46	74.21
	Exp2	**6,099**	**61.69**	**63.32**	**65.15**	**69.26**

Table 2. Accuracy of classifiers on Exp1 (7 sets) and Exp2 (11 sets). The words marked with * are excluded from the experiments because their 5-grams are very rare in the corpus.

for the sigmoid function, and ε_t for the adjustment step. The settings, $\eta = 25$, $\gamma = 35$, and $\varepsilon_t = 10^{-3}$, were determined by performing DT for several iterations through the training set. These setting were used for the following experiments.

3) Evaluation metric: The answers proposed by each classifier are the near-synonyms with the highest score. The correct answers are the near-synonyms originally in the gap of the test 5-grams. The performance is measure by the accuracy, which is defined as the number of correct answers made by each classifier, divided by the total number of test 5-grams.

In the following sections, we first demonstrate the effect of DT on classification performance, followed by the comparison of the classifiers.

4.2 Evaluation on discriminative training

This experiment is to investigate the performance change during discriminative training. Figure 2 shows the accuracy at the first 100 iterations for both development set and test set, with the 8th set in Exp2 as an example. The accuracy increased rapidly in the first 20 iterations, and stabilized after the 40th iteration. The discriminative training is stopped until the accuracy has not been changed over 30 iterations or the 300th iteration has been reached.

Features	Near-synonym set		
	mistake	error	oversight
made	**0.076**	-0.004	-0.005
biggest	**0.074**	-0.001	-0.004
message	-0.004	**0.039**	-0.010
internal	0.001	**0.026**	-0.001
supervision	-0.001	-0.006	**0.031**
audit	-0.002	-0.003	**0.028**

Table 3. Example of feature weights after discriminative training.

4.3 Comparative results

Table 2 shows the comparative results of the classification accuracy for the 18 near-synonym sets (Exp1 + Exp2). The accuracies for each near-synonym set were the average accuracies of the five randomly sampled test sets. The cosine measure without discrimination training (COS) was also considered for comparison. The results show that NGRAM performed worst among the four classifiers. The major reason is that not all 4-grams of the test examples can be found in the corpus. Instead of contiguous word associations used by NGRAM, PMI considers the words in the contexts independently to select the best synonyms. The results show that PMI achieved better performance than NGRAM. The two supervised methods, COS and DT, outperformed the two unsupervised methods, NGRAM and PMI. As indicated in the bold numbers, using the supervised method alone (without DT), COS yielded higher average accuracy by 5% and 2% over NGRAM and PMI, respectively, on Exp1, and by 6% and 3%, respectively, on Exp2. When DT was employed, the average accuracy was further improved by 4% and 6% on Exp1 and Exp2, respectively, compared with COS.

The use of DT can improve the classification accuracy mainly because it can adjust the feature weights iteratively to improve the separation between the correct class and its competing ones, which helps tackle the ambiguous test examples that fall within the decision boundary. Table 3 presents several positive and negative features for the near-synonym set {mistake, error, oversight}. The feature weights were adjusted according to their contributions to discriminating among the near-synonyms in the set. For instance, the features "made" and "biggest" both received a positive value for the class "mistake", and a negative value for the competing classes "error" and "oversight". These positive and negative weights help distinguish useful features from noisy ones for classifier training. On the other hand, if the feature weights were evenly distributed among the classes, these features would not be unlikely to contribute to the classification performance.

Exp1	Rank 1	Rank 2	Diff.
NGRAM	66.33%	79.35%	+19.63%
PMI	68.50%	88.99%	+29.91%
COS	69.94%	89.93%	+28.58%
DT	72.89%	91.06%	+24.93%
Exp2	Rank 1	Rank 2	Diff.
NGRAM	61.69%	68.48%	+11.01%
PMI	63.32%	79.11%	+24.94%
COS	65.15%	80.52%	+23.59%
DT	69.26%	82.86%	+19.64%

Table 4. Accuracy of Rank 1 and Rank 2 for each classifier.

4.4 Accuracy of Rank 1 and Rank 2

The accuracy presented in Table 2 was computed based on the classification results at Rank 1, i.e., a test sample was considered correctly classified only if the near-synonym with the highest score was the word originally in the gap of the test sample. Similarly, the accuracy at Rank 2 can be computed by considering the top two near-synonyms proposed by each classifier. That is, if either the near-synonym with the highest or the second highest score was the correct answer, the test sample was considered correctly classified. Table 4 shows the accuracy of Rank 1 and Rank 2 for each classifier. The results show that the improvement of Rank 1 accuracy on Exp1 was about 20 to 30 percentage points, and was 25.76 in average. For Exp2, the average improvement was 19.80 percentage points

5 Conclusion

This work has presented the use of discriminative training for near-synonym substitution. The discriminative training can improve classification performance by iteratively re-weighting the positive and negative features for each class. This helps improve the separation of the correct class against its competing ones, making classifiers more effective on ambiguous cases close to the decision boundary. Experimental results show that the supervised discriminative training technique achieves higher accuracy than the two unsupervised methods, the PMI-based and n-gram-based methods. The availability of a large labeled training set also encourages the use of the proposed supervised method.

Future work will investigate on the use of multiple features for discriminating among near-synonyms. For instance, the predicate-argument structure, which can capture long-distance information, will be combined with currently used local contextual features to boost the classification performance. More experiments will also be conducted to evaluate classifiers' ability to rank multiple answers.

References

J. Bhogal, A. Macfarlane, and P. Smith. 2007. A Review of Ontology based Query Expansion. *Information Processing & Management*, 43(4):866-886.

K. Church and P. Hanks. 1991. Word Association Norms, Mutual Information and Lexicography. *Computational Linguistics*, 16(1):22-29.

I. Dagan, O. Glickman, A. Gliozzo, E. Marmorshtein, and C. Strapparava. 2006. Direct Word Sense Matching for Lexical Substitution. In *Proc. of COLING/ACL-06*, pages 449-456.

P. Edmonds. 1997. Choosing the Word Most Typical in Context Using a Lexical Co-occurrence Network. In *Proc. of ACL-97*, pages 507-509.

C. Fellbaum. 1998. WordNet: An Electronic Lexical Database. MIT Press, Cambridge, MA.

M. Gardiner and M. Dras. 2007. Exploring Approaches to Discriminating among Near-Synonyms, In *Proc. of the Australasian Technology Workshop*, pages 31-39.

E. H. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. OntoNotes: The 90% Solution. In *Proc. of HLT/NAACL-06*, pages 57–60.

D. Inkpen. 2007. Near-Synonym Choice in an Intelligent Thesaurus. In *Proc. of NAACL/HLT-07*, pages 356-363.

D. Inkpen and G. Hirst. 2006. Building and Using a Lexical Knowledge-base of Near-Synonym Differences. *Computational Linguistics*, 32(2):1-39.

S. Katagiri, B. H. Juang, and C. H. Lee. 1998. Pattern Recognition Using a Family of Design Algorithms based upon the Generalized Probabilistic Descent Method, *Proc. of the IEEE*, 86(11):2345-2373.

H. K. J. Kuo and C. H. Lee. 2003. Discriminative Training of Natural Language Call Routers, *IEEE Trans. Speech and Audio Processing*, 11(1):24-35.

D. Lin. 1998. Automatic Retrieval and Clustering of Similar Words. In *Proc. of ACL/COLING-98*, pages 768-774.

D. McCarthy. 2002. Lexical Substitution as a Task for WSD Evaluation. In *Proc. of the SIGLEX/SENSEVAL Workshop on Word Sense Disambiguation at ACL-02*, pages 109-115.

D. Moldovan and R. Mihalcea. 2000. Using Wordnet and Lexical Operators to Improve Internet Searches. *IEEE Internet Computing*, 4(1):34-43.

U. Ohler, S. Harbeck, and H. Niemann. 1999. Discriminative Training of Language Model Classifiers, In *Proc. of Eurospeech-99*, pages 1607-1610.

D. Pearce. 2001. Synonymy in Collocation Extraction. In *Proc. of the Workshop on WordNet and Other Lexical Resources at NAACL-01*.

S. Pradhan, E. H. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2007. OntoNotes: A Unified Relational Semantic Representation. In *Proc. of the First IEEE International Conference on Semantic Computing (ICSC-07)*, pages 517-524.

H. Rodríguez, S. Climent, P. Vossen, L. Bloksma, W. Peters, A. Alonge, F. Bertagna, and A. Roventint. 1998. The Top-Down Strategy for Building EeuroWordNet: Vocabulary Coverage, Base Concepts and Top Ontology, *Computers and the Humanities*, 32:117-159.

L. C. Yu, C. H. Wu, A. Philpot, and E. H. Hovy. 2007. OntoNotes: Sense Pool Verification Using Google N-gram and Statistical Tests, In *Proc. of the OntoLex Workshop at the 6th International Semantic Web Conference (ISWC-07).*

D. Zhou and Y. He. 2009. Discriminative Training of the Hidden Vector State Model for Semantic Parsing, *IEEE Trans. Knowledge and Data Engineering*, 21(1):66-77.

Estimating Linear Models for Compositional Distributional Semantics

Fabio Massimo Zanzotto[1]
(1) Department of Computer Science
University of Rome "Tor Vergata"
zanzotto@info.uniroma2.it

Ioannis Korkontzelos
Department of Computer Science
University of York
johnkork@cs.york.ac.uk

Francesca Fallucchi[1,2]
(2) Università Telematica
"G. Marconi"
f.fallucchi@unimarconi.it

Suresh Manandhar
Department of Computer Science
University of York
suresh@cs.york.ac.uk

Abstract

In distributional semantics studies, there is a growing attention in compositionally determining the distributional meaning of word sequences. Yet, compositional distributional models depend on a large set of parameters that have not been explored. In this paper we propose a novel approach to estimate parameters for a class of compositional distributional models: the additive models. Our approach leverages on two main ideas. Firstly, a novel idea for extracting compositional distributional semantics examples. Secondly, an estimation method based on regression models for multiple dependent variables. Experiments demonstrate that our approach outperforms existing methods for determining a good model for compositional distributional semantics.

1 Introduction

Lexical distributional semantics has been largely used to model word meaning in many fields as computational linguistics (McCarthy and Carroll, 2003; Manning et al., 2008), linguistics (Harris, 1964), corpus linguistics (Firth, 1957), and cognitive research (Miller and Charles, 1991). The fundamental hypothesis is the distributional hypothesis (DH): "similar words share similar contexts" (Harris, 1964). Recently, this hypothesis has been operationally defined in many ways in the fields of physicology, computational linguistics, and information retrieval (Li et al., 2000; Pado and Lapata, 2007; Deerwester et al., 1990).

Given the successful application to words, distributional semantics has been extended to word sequences. This has happened in two ways: (1) via the reformulation of DH for specific word sequences (Lin and Pantel, 2001); and (2) via the definition of compositional distributional semantics (CDS) models (Mitchell and Lapata, 2008; Jones and Mewhort, 2007). These are two different ways of addressing the problem.

Lin and Pantel (2001) propose the *pattern distributional hypothesis* that extends the distributional hypothesis for specific patterns, i.e. word sequences representing partial verb phrases. Distributional meaning for these patterns is derived directly by looking to their occurrences in a corpus. Due to data sparsity, patterns of different length appear with very different frequencies in the corpus, affecting their statistics detrimentally. On the other hand, compositional distributional semantics (CDS) propose to obtain distributional meaning for sequences by composing the vectors of the words in the sequences (Mitchell and Lapata, 2008; Jones and Mewhort, 2007). This approach is fairly interesting as the distributional meaning of sequences of different length is obtained by composing distributional vectors of single words. Yet, many of these approaches have a large number of parameters that cannot be easily estimated.

In this paper we propose a novel approach to es-

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1263–1271,
Beijing, August 2010

timate parameters for additive compositional distributional semantics models. Our approach leverages on two main ideas. Firstly, a novel way for extracting compositional distributional semantics examples and counter-examples. Secondly, an estimation model that exploits these examples and determines an equation system that represents a regression problem with multiple dependent variables. We propose a method to estimate a solution of this equation system based on the Moore-Penrose pseudo-inverse matrices (Penrose, 1955).

The rest of the paper is organised as follows: Firstly, we shortly review existing compositional distributional semantics (CDS) models (Sec. 2). Then we describe our model for estimating CDS models parameters (Sec. 3). In succession, we introduce a way to extract compositional distributional semantics examples from dictionaries (Sec. 4). Then, we discuss the experimental set up and the results of our linear CDS model with estimated parameters with respect to existing CDS models (Sec. 5).

2 Models for compositional distributional semantics (CDS)

A CDS model is a function $\odot$ that computes the distributional vector of a sequence of words s by combining the distributional vectors of its component words $\mathsf{w}_1 \dots \mathsf{w}_n$. Let $\odot(\mathsf{s})$ be the distributional vector describing s and $\vec{w}_i$ the distributional vectors describing its component word w_i. Then, the CDS model can be written as:

$$\odot(\mathsf{s}) = \odot(\mathsf{w}_1 \dots \mathsf{w}_n) = \vec{w}_1 \odot \dots \odot \vec{w}_n \qquad (1)$$

This generic model has been fairly studied and many different functions have been proposed and tested.

Mitchell and Lapata (2008) propose the following general CDS model for 2-word sequences $\mathsf{s} = \mathsf{xy}$:

$$\odot(\mathsf{s}) = \odot(\mathsf{xy}) = f(\vec{x}, \vec{y}, R, K) \qquad (2)$$

where $\vec{x}$ and $\vec{y}$ are respectively the distributional vectors of x and y, R is the particular syntactic and/or semantic relation connecting x and y, and, K represents the amount of background knowledge that the vector composition process takes into account. Two specialisations of the general CDS model are proposed: the *basic additive* model and the *basic multiplicative* model.

	vector dimensions				
	between	gap	process	social	two
contact	< 11,	0,	3,	0,	11 >
x: *close*	< 27,	3,	2,	5,	24 >
y: *interaction*	< 23,	0,	3,	8,	4 >

Table 1: Example of distributional frequency vectors for the triple $t = (\vec{contact}, \vec{close}, \vec{interaction})$

The *basic additive* model (BAM) is written as:

$$\odot(\mathsf{s}) = \alpha\vec{x} + \beta\vec{y} \qquad (3)$$

where α and β are two scalar parameters. The simplistic parametrisation is $\alpha = \beta = 1$. For example, given the vectors $\vec{x}$ and $\vec{y}$ of Table 1, $\odot_{BAM}(\mathsf{s}) =< 50, 3, 5, 13, 28 >$.

The *basic multiplicative* model (BMM) is written as:

$$s_i = x_i y_i \qquad (4)$$

where s_i, x_i, and y_i are the i-th dimensions of the vectors $\odot(\mathsf{s})$, $\vec{x}$, and $\vec{y}$, respectively. For the example of Table 1, $\odot_{BMM}(\mathsf{s}) =< 621, 0, 6, 40, 96 >$.

Erk and Padó (2008) look at the problem in a different way. Let the general distributional meaning of the word w be $\vec{w}$. Their model computes a different vector $\vec{w}_{\mathsf{s}}$ that represents the specific distributional meaning of w with respect to s, i.e.:

$$\vec{w}_{\mathsf{s}} = \oslash(\mathsf{w}, \mathsf{s}) \qquad (5)$$

In general, this operator gives different vectors for each word w_i in the sequence s, i.e. $\oslash(\mathsf{w}_i, \mathsf{s}) \neq \oslash(\mathsf{w}_j, \mathsf{s})$ if $i \neq j$. It also gives different vectors for a word w_i appearing in different sequences s_k and s_l, i.e. $\oslash(\mathsf{w}_i, \mathsf{s}_k) \neq \oslash(\mathsf{w}_i, \mathsf{s}_l)$ if $k \neq l$.

The model of Erk and Padó (2008) was designed to *disambiguate* the distributional meaning of a word w in the context of the sequence s. However, substituting the word w with the semantic head h of s, allows to compute the distributional meaning of sequence s as shaped by the word that is governing the sequence (c.f. Pollard and Sag (1994)). For example, the distributional meaning of the word sequence *eats mice* is governed by the verb *eats*. Following this model, the distributional vector $\odot(\mathsf{s})$ can be written as:

$$\odot(\mathsf{s}) \approx \oslash(\mathsf{h}, \mathsf{s}) \tag{6}$$

The function $\oslash(\mathsf{h}, \mathsf{s})$ explicitly uses the relation R and the knowledge K of the general equation 2, being based on the notion of selectional preferences. We exploit the model for sequences of two words s=xy where the two words are related with an oriented syntactic relation r (e.g. r=adj_modifier). For making the syntactic relation explicit, we indicate the sequence as: $\mathsf{s} = \mathsf{x} \overset{r}{\leftarrow} \mathsf{y}$.

Given a word w, the model has to keep track of its selectional preferences. Consequently, each word w is represented with a triple:

$$(\vec{w}, R_{\mathsf{w}}, R_{\mathsf{w}}^{-1}) \tag{7}$$

where $\vec{w}$ is the distributional vector of the word w, R_{w} is the set of the vectors representing the direct selectional preferences of the word w, and R_{w}^{-1} is the set of the vectors representing the indirect selectional preferences of the word w. Given a set of syntactic relations $\mathcal{R}$, the set R_{w} and R_{w}^{-1} contain respectively a selectional preference vector $R_{\mathsf{w}}(r)$ and $R_{\mathsf{w}}(r)^{-1}$ for each $r \in \mathcal{R}$. Selectional preferences are computed as in Erk (2007). If x is the semantic head of sequence s, then the model can be written as:

$$\odot(\mathsf{s}) = \oslash(\mathsf{x}, \mathsf{x} \overset{r}{\leftarrow} \mathsf{y}) = \vec{x} \odot R_{\mathsf{y}}(r) \tag{8}$$

Otherwise, if y is the semantic head:

$$\odot(\mathsf{s}) = \oslash(\mathsf{y}, \mathsf{x} \overset{r}{\leftarrow} \mathsf{y}) = \vec{y} \odot R_{\mathsf{x}}^{-1}(r) \tag{9}$$

$\odot$ is in both cases realised using BAM or BMM. We will call these models: *basic additive* model with selectional preferences (BAM-SP) and basic multiplicative model with selectional preferences (BMM-SP).

Both Mitchell and Lapata (2008) and Erk and Padó (2008) experimented with few empirically estimated parameters. Thus, the general additive CDS model has not been adequately explored.

3 Estimating Additive Compositional Semantics Models from Data

The generic *additive* model sums the vectors $\vec{x}$ and $\vec{y}$ in a new vector $\vec{z}$:

$$\odot(\mathsf{s}) = \vec{z} = A\vec{x} + B\vec{y} \tag{10}$$

where A and B are two square matrices capturing the relation R and the background knowledge K of equation 2. Writing matrices A and B by hand is impossible because of their large size. Estimating these matrices is neither a simple classification learning problem nor a simple regression problem. It is a regression problem with multiple dependent variables. In this section, we propose our model to solve this regression problem using a set of training examples E.

The set of training examples E contains triples of vectors $(\vec{z}, \vec{x}, \vec{y})$. $\vec{x}$ and $\vec{y}$ are the two distributional vectors of the words x and y. $\vec{z}$ is the expected distributional vector of the composition of $\vec{x}$ and $\vec{y}$. Note that for an ideal perfectly performing CDS model we can write $\vec{z} = \odot(\mathsf{xy})$. However, in general the expected vector $\vec{z}$ is not guaranteed to be equal to the composed one $\odot(\mathsf{xy})$. Figure 1 reports an example of these triples, i.e., $t = (\vec{contact}, \vec{close}, \vec{interaction})$, with the related distributional vectors. The construction of E is discussed in section 4.

In the rest of the section, we describe how the regression problem with multiple dependent variables can be solved with a linear equation system and we give a possible solution of this equation system. In the experimental section, we refer to our model as the estimated additive model (EAM).

3.1 Setting the linear equation system

The matrices A and B of equation 10 can be joined in a single matrix:

$$\vec{z} = \begin{pmatrix} A & B \end{pmatrix} \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} \tag{11}$$

For the triple t of table 1, equation 11 is:

$$\vec{contact} = \begin{pmatrix} A & B \end{pmatrix} \begin{pmatrix} \vec{close} \\ \vec{interaction} \end{pmatrix} \tag{12}$$

and it can be rewritten as:

$$\begin{pmatrix} 11 \\ 0 \\ 3 \\ 0 \\ 11 \end{pmatrix} = \begin{pmatrix} A_{5\times 5} & B_{5\times 5} \end{pmatrix} \begin{pmatrix} 27 \\ 3 \\ 2 \\ 5 \\ 24 \\ 23 \\ 0 \\ 3 \\ 8 \\ 4 \end{pmatrix} \tag{13}$$

Focusing on matrix $\begin{pmatrix} AB \end{pmatrix}$, we can transpose the matrices as follows:

$$\begin{aligned} \vec{z}^T &= \left(\begin{pmatrix} A & B \end{pmatrix} \begin{pmatrix} \vec{x} \\ \vec{y} \end{pmatrix} \right)^T \\ &= \begin{pmatrix} \vec{x}^T & \vec{y}^T \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} \end{aligned} \tag{14}$$

Matrix $\begin{pmatrix} \vec{x}^T & \vec{y}^T \end{pmatrix}$ is known and matrix $\begin{pmatrix} A^T \\ B^T \end{pmatrix}$ is to be estimated.

Equation 14 is the prototype of our final equation system. The larger the matrix $\begin{pmatrix} AB \end{pmatrix}$ to be estimated, the more equations like 14 are needed. Given set E that contains n triples $(\vec{z}, \vec{x}, \vec{y})$, we can write the following system of equations:

$$\begin{pmatrix} \vec{z}_1^T \\ \vec{z}_2^T \\ \vdots \\ \vec{z}_n^T \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} \vec{x}_1^T & \vec{y}_1^T \end{pmatrix} \\ \begin{pmatrix} \vec{x}_2^T & \vec{y}_2^T \end{pmatrix} \\ \vdots \\ \begin{pmatrix} \vec{x}_n^T & \vec{y}_n^T \end{pmatrix} \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} \tag{15}$$

The vectors derived from the triples can be seen as two matrices of n rows, Z and $\begin{pmatrix} XY \end{pmatrix}$ related to $\vec{z}_i^T$ and $\begin{pmatrix} \vec{x}_i^T & \vec{y}_i^T \end{pmatrix}$, respectively. The overall equation system is then the following:

$$Z = \begin{pmatrix} X & Y \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} \tag{16}$$

This equation system represents the constraints that matrices A and B have to satisfy in order to be a possible linear CDS model that can at least describe seen examples. We will hereafter call $\Lambda = \begin{pmatrix} A & B \end{pmatrix}$ and $Q = \begin{pmatrix} X & Y \end{pmatrix}$. The system in equation 16 can be simplified as:

$$Z = Q\Lambda^T \tag{17}$$

As Q is a rectangular and singular matrix, it is not invertible and the system in equation 16 has no solutions. It is possible to use the principle of Least Square Estimation for computing an approximation solution. The idea is to compute the solution $\widehat{\Lambda}$ that minimises the residual norm, i.e.:

$$\widehat{\Lambda}^T = \arg\min_{\Lambda^T} \|Q\Lambda^T - Z\|^2 \tag{18}$$

One solution for this problem is the **Moore-Penrose pseudoinverse** Q^+ (Penrose, 1955) that gives the following final equation:

$$\widehat{\Lambda}^T = Q^+ Z \tag{19}$$

In the next section, we discuss how the **Moore-Penrose pseudoinverse** is obtained using singular value decomposition (SVD).

3.2 Computing the pseudo-inverse matrix

The pseudo-inverse matrix can provide an approximated solution even if the equation system has no solutions. We here compute the **Moore-Penrose pseudoinverse** using singular value decomposition (SVD) that is widely used in computational linguistics and information retrieval for reducing spaces (Deerwester et al., 1990).

Moore-Penrose pseudoinverse (Penrose, 1955) is computed in the following way. Let the original matrix Q have n rows and m columns and be of rank r. The SVD decomposition of the original matrix Q is $Q = U\Sigma V^T$ where Σ is a square diagonal matrix of dimension r. Then, the pseudoinverse matrix that minimises the equation 18 is:

$$Q^+ = V\Sigma^+ U^T \tag{20}$$

where the diagonal matrix Σ^+ is the $r \times r$ transposed matrix of Σ having as diagonal elements the reciprocals of the singular values $\frac{1}{\delta_1}, \frac{1}{\delta_2}, ..., \frac{1}{\delta_r}$ of Σ.

Using SVD to compute the pseudo-inverse matrix allows for different approximations (Fallucchi and Zanzotto, 2009). The algorithm for computing the singular value decomposition is iterative (Golub and Kahan, 1965). Firstly derived dimensions have higher singular value. Then, dimension k is more informative than dimension $k' > k$. We can consider different values for k to obtain different SVD for the approximations Q_k^+ of the original matrix Q^+ in equation 20), i.e.:

$$Q_k^+ = V_{n\times k}\Sigma_{k\times k}^+ U_{k\times m}^T \tag{21}$$

where Q_k^+ is a matrix n by m obtained considering the first k singular values.

4 Building positive and negative examples

As explained in the previous section, estimating CDS models, needs a set of triples E, similar to triple t of table 1. This set E should contain positive examples in the form of triples $(\vec{z_i}, \vec{x_i}, \vec{y_i})$. Examples are positive in the sense that $\vec{z_i} = \odot(\mathsf{xy})$ for an ideal CDS. There are no available sets to contain such triples, with the exception of the set used in Mitchell and Lapata (2008) which is designed only for testing purposes. It contains similar and dissimilar pairs of sequences (s_1,s_2) where each sequence is a verb-noun pair (v_i,n_i). From the positive part of this set, we can only derive quadruples where $\odot(\mathsf{v}_1\mathsf{n}_1) \approx \odot(\mathsf{v}_2\mathsf{n}_2)$ but we cannot derive the ideal resulting vector of the composition $\odot(\mathsf{v}_i\mathsf{n}_i)$. Sets used to test multiword expression (MWE) detection models (e.g., (Schone and Jurafsky, 2001; Nicholson and Baldwin, 2008; Kim and Baldwin, 2008; Cook et al., 2008; Villavicencio, 2003; Korkontzelos and Manandhar, 2009)) are again not useful as containing only valid MWE that cannot be used to determine the set of training triples needed here.

As a result, we need a novel idea to build sets of triples to train CDS models. We can leverage on knowledge stored in dictionaries. In the rest of the section, we describe how we build the positive example set E and a control negative example set NE. Elements of the two sets are pairs (t,s) where t is a target word s is a sequence of words. t is the word that represent the distributional meaning of s in the case of E. Contrarily, t is totally unrelated to the distributional meaning of s in NE. The sets E and NE can be used both for training and for testing. In the testing phase, we can use these sets to determine whether a CDS model is good or not and to compare different CDS models.

4.1 Building Positive Examples using Dictionaries

Dictionaries as natural repositories of equivalent expressions can be used to extract positive examples for training and testing CDS models. The basic idea is the following: dictionary entries are *declarations* of equivalence. Words or, occasionally, multi-word expressions t are declared to be semantically similar to their definition sequences s. This happens at least for some sense of the defined words. We can then observe that t ≈ s. For example, we report some sample definitions of *contact* and *high life*:

target word (t)	definition sequence (s)
contact	close interaction
high life	excessive spending

In the first case, a word, i.e. *contact*, is semantically similar to a two-word expression, i.e. *close interaction*. In the second case, two two-word expressions are semantically similar.

Then, the pairs (t, s) can be used to model positive cases of compositional distributional semantics as we know that the word sequence s is compositional and it describes the meaning of the word t. The distributional meaning $\vec{t}$ of t is the expected distributional meaning of s. Consequently, the vector $\vec{t}$ is what the CDS model $\odot(\mathsf{s})$ should compositionally obtain from the vectors of the components $\vec{s_1} \dots \vec{s_m}$ of s. This way of extracting similar expressions has some interesting properties:

First property Defined words t are generally single words. Thus, we can extract stable and meaningful distributional vectors for these words and then compare them to the distributional vectors composed by CDS model. This is an important property as we cannot compare directly the distributional vector $\vec{s}$ of a word sequence s and the vector $\odot(\mathsf{s})$ obtained by composing its components. As the word sequence s grows in length, the reliability of the vector $\vec{s}$ decreases since the sequence s becomes rarer.

Second property Definitions s have a large variety of different syntactic structures ranging from simple structures as Adjective-Noun to more complex ones. This gives the possibility to train and test CDS models that take into account syntax. Table 2 represents the distribution of the more frequent syntactic structures in the definitions of WordNet[1] (Miller, 1995).

[1] Definitions were extracted from WordNet 3.0 and were parsed with the Charniak parser (Charniak, 2000)

Freq.	*Structure*
2635	(FRAG (PP (IN) (NP (DT) (JJ) (NN))))
833	(NP (DT) (JJ) (NN))
811	(NP (NNS))
645	(NP (NNP))
623	(S (VP (VB) (ADVP (RB))))
610	(NP (JJ) (NN))
595	(NP (NP (DT) (NN)) (PP (IN) (NP (NN))))
478	(NP (NP (DT) (NN)) (PP (IN) (NP (NNP))))
451	(FRAG (PP (IN) (NP (NN))))
419	(FRAG (RB) (ADJP (JJ)))
375	(S (VP (VB) (PP (IN) (NP (DT) (NN)))))
363	(S (VP (VB) (PP (IN) (NP (NN)))))
342	(NP (NP (DT) (NN)) (PP (IN) (NP (DT) (NN))))
341	(NP (DT) (JJ) (JJ) (NN))
330	(ADJP (RB) (JJ))
307	(NP (JJ) (NNS))
244	(NP (DT) (NN) (NN))
241	(S (NP (NN)) (NP (NP (NNS)) (PP (IN) (NP (DT) (NNP)))))
239	(NP (NP (DT) (JJ) (NN)) (PP (IN) (NP (DT) (NN))))

Table 2: Top 20 syntactic structures of WordNet definitions

4.2 Extracting Negative Examples from Word Etymology

In order to devise complete training and testing sets for CDS models, we need to find a sensible way to extract negative examples. An option is to randomly generate totally unrelated triples for the negative examples set, NE. In this case, due to data sparseness NE would mostly contain triples $(\vec{z}, \vec{x}, \vec{y})$ where it is expected that $\vec{z} \neq \odot(\mathsf{xy})$. Yet, these can be too generic and too loosely related to be interesting cases.

Instead we attempt to extract sets of negative pairs (t,s) comparable with the one used for building the training set E. The target word t should be a single word and s should be a sequence of words. The latter should be a sequence of words related by construction to t but the meaning of t and s should be unrelated.

The idea is the following: many words are etymologically derived from very old or ancient words. These words represent a collocation which is in general not related to the meaning of the target word. For example, the word *philosophy* derives from two Greek words *philos* (beloved) and *sophia* (wisdom). However, the use of the word *philosophy* in not related to the collocation *beloved wisdom*. This word has lost its original compositional meaning. The following table shows some more etymologically complex words along with the compositionally unrelated collocations:

target word	compositionally unrelated seq.
municipal	receive duty
octopus	eight foot

As the examples suggest, we are able to build a set NE with features similar to the features of N. In particular, each target word is paired with a related word sequence derived from its etymology. These etymologically complex words are unrelated to the corresponding compositional collocations. To derive a set NE with the above characteristics we can use dictionaries containing etymological information as Wiktionary[2].

5 Experimental evaluation

In the previous sections, we presented the estimated additive model (EAM): our approach to estimate the parameters of a generic additive model for CDS. In this section, we experiment with this model to determine whether it performs better than existing models: the basic additive model (BAM), the basic multiplicative model (BMM), the basic additive model with selectional preferences (BAM-SP), and the basic multiplicative model with selectional preferences (BMM-SP) (c.f. Sec. 2). In succession, we explore whether our estimated additive model (EAM) is better than any possible BAM obtained with parameter adjustment. In the rest of the section, we firstly give the experimental setup and then we discuss the experiments and the results.

5.1 Experimental setup

Our experiments aim to compare compositional distributional semantic (CDS) models $\odot$ with respect to their ability of detecting statistically significant difference between sets E and NE. In particular, the average similarity $sim(\vec{z}, \odot(\mathsf{xy}))$ for $(\vec{z}, \vec{x}, \vec{y}) \in E$ should be significantly different from $sim(\vec{z}, \odot(\mathsf{xy}))$ for $(\vec{z}, \vec{x}, \vec{y}) \in NE$. In this section, we describe the chosen similarity measure sim, statistical significance testing and construction details for the training and testing set.

Cosine similarity was used to compare the context vector $\vec{z}$ representing the target word z with the composed vector $\odot(\mathsf{xy})$ representing the context vector of sequence $\mathsf{x}\ \mathsf{y}$. Cosine similarity be-

[2] http://www.wiktionary.org

tween two vectors $\vec{x}$ and $\vec{y}$ of the same dimension is defined as:

$$sim(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{\|\vec{x}\|\;\|\vec{y}\|} \tag{22}$$

where $\cdot$ is the dot product and $\|\vec{a}\|$ is the magnitude of vector $\vec{a}$ computed the Euclidean norm.

To evaluate whether a CDS model distinguishes positive examples E from negative examples NE, we test if the distribution of similarities $sim(\vec{z}, \odot(\mathsf{xy}))$ for $(\vec{z},\vec{x},\vec{y}) \in E$ is statistically different from the distribution of the same similarities for $(\vec{z},\vec{x},\vec{y}) \in NE$. For this purpose, we used Student's t-test for two independent samples of different sizes. t-test assumes that the two distributions are Gaussian and determines the probability that they are similar, i.e., derive from the same underlying distribution. Low probabilities indicate that the distributions are highly dissimilar and that the corresponding CDS model performs well, as it detects statistically different similarities for the positive set E and the negative set NE.

Based on the null hypothesis that the means of the two samples are equal, $\mu_1 = \mu_2$, Student's t-test takes into account the sizes N, means M and variances s^2 of the two samples to compute the following value:

$$t = (M_1 - M_2)\sqrt[-1]{\frac{2(s_1^2+s_2^2)}{df * N_h}} \tag{23}$$

where $df = N_1 + N_2 - 2$ stands for the degrees of freedom and $N_h = 2(N_1^{-1} + N_2^{-1})^{-1}$ is the harmonic mean of the sample sizes. Given the statistic t and the degrees of freedom df, we can compute the corresponding p-value, i.e., the probability that the two samples derive from the same distribution. The null hypothesis can be rejected if the p-value is below the chosen threshold of statistical significance (usually 0.1, 0.05 or 0.01), otherwise it is accepted. In our case, rejecting the null hypothesis means that the similarity values of instances of E are significantly different from instances of NE, and that the corresponding CDS model perform well. p-value can be used as a performance ranking function for CDS models.

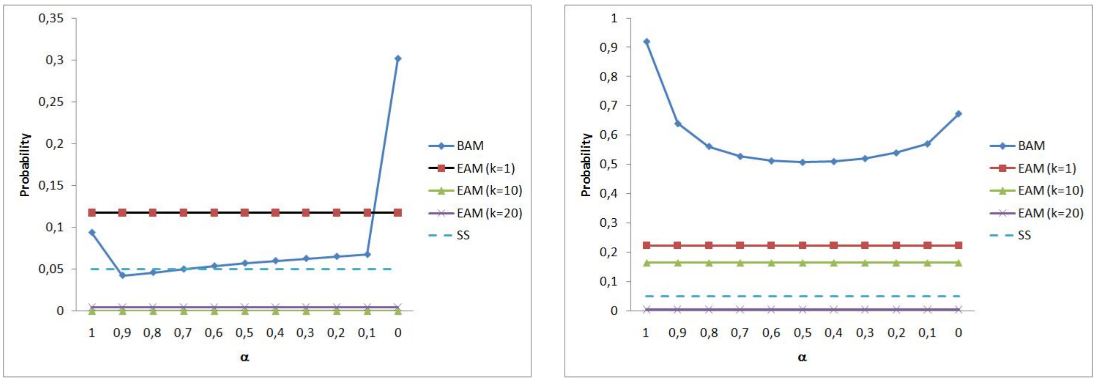
We constructed two sets of instances: (a) a set containing Adjective-Noun or Noun-Noun sequences (*NN* set); and (b) a set containing Verb-Noun sequences (*VN* set). Capturing different syntactic relations, these two sets can support that our results are independent from the syntactic relation between the words of each sequence. For each set, we used WordNet for extracting positive examples E and Wiktionary for extracting negative examples NE as described in Section 4. We obtained the following sets: (a) *NN* consists of 1065 word-sequence pairs from WordNet definitions and 377 pairs extracted from Wiktionary; and (b) *VN* consists of 161 word-sequence pairs from WordNet definitions and 111 pairs extracted from Wiktionary. We have then divided these two sets in two parts of 50% each, for training and testing. Instances of the training part of E have been used to estimate matrices A and B for model EAM, while the testing parts have been used for testing all models. Frequency vectors for all single words occurring in the above pairs were constructed from the British National Corpus using sentences as contextual windows and words as features. The resulting space has 689191 features.

	NN set	*VN* set
BAM	0.05690	0.50753
BMM	0.20262	0.37523
BAM-SP	0.42574	0.01710
BMM-SP	<1.00E-10	0.23552
EAM (k=20)	0.00431	0.00453

Table 3: Probability of confusing E and NE with different CDS models

5.2 Results and Analysis

The first set of experiments compares EAM with other existing CDS models: BAM, BMM, BAM-SP, and BMM-SP. Results are shown in Table 3. The table reports the p-value, i.e., the probability of confusing the positive set E and the negative set NE for all models. Lower probabilities characterise better models. Probabilities below 0.05 indicate that the model detects a statistically significant difference between sets E and NE. EAM has been computed with $k = 20$ different dimensions for the pseudo-inverse matrix. The two basic additive models (BAM and BAM-SP) have been computed for $\alpha = \beta = 1$.



NN set VN *set*

Figure 1: p-values of BAM with different values for parameter α (where $\beta = 1 - \alpha$) and of EAM for different approximations of the SVD pseudo-inverse matrix (k)

The first observation is that EAM models significantly separate positive from negative examples for both sets. This is not the case for any of the other models. Only, the selectional preferences based models in two cases have this property, but this cannot be generalised: BAM-SP on the *VN* set and BMM-SP on the *NN* set. In general, these models do not offer the possibility of separating positive from negative examples.

In the second set of experiments, we attempt to investigate whether simple parameter adjustment of BAM can perform better than EAM. Results are shown in figure 1. Plots show the basic additive model (BAM) with different values for parameter α (where $\beta = 1 - \alpha$) and EAM computed for different approximations of the SVD pseudo-inverse matrix (i.e., with different k). The x-axis of the plots represents parameter α and the y-axis represents the probability of confusing the positive set E and the negative set NE. The representation focuses on the performance of BAM with respect to different α values. The performance of EAM for different k values is represented with horizontal lines. Probabilities of different models are directly comparable. Line SS represents the threshold of statistical significance; the value below which the detected difference between the E and NE sets becomes statistically significant.

Experimental results show some interesting facts: While BAM for $\alpha > 0$ perform better than EAM computed with $k = 1$ in the NN set, they do not perform better in the VN set. EAM with $k = 1$ has 1 degree of freedom corresponding to 1 parameter, the same as BAM. The parameter of EAM is tuned on the training set, in contrast to α, the parameter of BAM. Increasing the number of considered dimensions, k of EAM, estimated models outperform BAM for all values of parameter α. Moreover, EAM detect a statistically significant difference between the E and the NE sets for $k \geq 10$ and $k = 20$ for the *NN* set and the *VN* set set, respectively. Simple parametrisation of a BAM does not outperform the proposed estimated additive model.

6 Conclusions

In this paper, we presented an innovative method to estimate linear compositional distributional semantics models. The core of our approach consists on two parts: (1) providing a method to estimate the regression problem with multiple dependent variables and (2) providing a training set derived from dictionary definitions. Experiments showed that our model is highly competitive with respect to state-of-the-art models for compositional distributional semantics.

References

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *proceedings of the 1st NAACL*, pages 132–139, Seattle, Washington.

Cook, Paul, Afsaneh Fazly, and Suzanne Stevenson. 2008. The VNC-Tokens Dataset. In *proceedings of the LREC Workshop: Towards a Shared Task for Multiword Expressions (MWE 2008)*, Marrakech, Morocco.

Deerwester, Scott C., Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

Erk, Katrin and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 897–906. Association for Computational Linguistics.

Erk, Katrin. 2007. A simple, similarity-based model for selectional preferences. In *proceedings of ACL*. Association for Computer Linguistics.

Fallucchi, Francesca and Fabio Massimo Zanzotto. 2009. SVD feature selection for probabilistic taxonomy learning. In *proceedings of the Workshop on Geometrical Models of Natural Language Semantics*, pages 66–73. Association for Computational Linguistics, Athens, Greece.

Firth, John R. 1957. *Papers in Linguistics.* Oxford University Press, London.

Golub, Gene and William Kahan. 1965. Calculating the singular values and pseudo-inverse of a matrix. *Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis*, 2(2):205–224.

Harris, Zellig. 1964. Distributional structure. In Katz, Jerrold J. and Jerry A. Fodor, editors, *The Philosophy of Linguistics*, New York. Oxford University Press.

Jones, Michael N. and Douglas J. K. Mewhort. 2007. Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114:1–37.

Kim, Su N. and Timothy Baldwin. 2008. Standardised evaluation of english noun compound interpretation. In *proceedings of the LREC Workshop: Towards a Shared Task for Multiword Expressions (MWE 2008)*, pages 39–42, Marrakech, Morocco.

Korkontzelos, Ioannis and Suresh Manandhar. 2009. Detecting compositionality in multi-word expressions. In *proceedings of ACL-IJCNLP 2009*, Singapore.

Li, Ping, Curt Burgess, and Kevin Lund. 2000. The acquisition of word meaning through global lexical co-occurrences. In *proceedings of the 31st Child Language Research Forum.*

Lin, Dekang and Patrick Pantel. 2001. DIRT-discovery of inference rules from text. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD-01).* San Francisco, CA.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval.* Cambridge University Press, Cambridge, UK.

McCarthy, Diana and John Carroll. 2003. Disambiguating nouns, verbs, and adjectives using automatically acquired selectional preferences. *Computational Linguistics*, 29(4):639–654.

Miller, George A. and Walter G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, VI:1–28.

Miller, George A. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Mitchell, Jeff and Mirella Lapata. 2008. Vector-based models of semantic composition. In *proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio. Association for Computational Linguistics.

Nicholson, Jeremy and Timothy Baldwin. 2008. Interpreting compound nominalisations. In *proceedings of the LREC Workshop: Towards a Shared Task for Multiword Expressions (MWE 2008)*, pages 43–45, Marrakech, Morocco.

Pado, Sebastian and Mirella Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

Penrose, Roger. 1955. A generalized inverse for matrices. In *Proceedings of Cambridge Philosophical Society.*

Pollard, Carl J. and Ivan A. Sag. 1994. *Head-driven Phrase Structured Grammar.* Chicago CSLI, Stanford.

Schone, Patrick and Daniel Jurafsky. 2001. Is knowledge-free induction of multiword unit dictionary headwords a solved problem? In Lee, Lillian and Donna Harman, editors, *proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 100–108.

Villavicencio, Aline. 2003. Verb-particle constructions and lexical resources. In *proceedings of the ACL 2003 workshop on Multiword expressions*, pages 57–64, Morristown, NJ, USA. Association for Computational Linguistics.

Grouping Product Features Using Semi-Supervised Learning with Soft-Constraints*

Zhongwu Zhai[†], **Bing Liu**[‡], **Hua Xu**[†] **and Peifa Jia**[†]

[†]State Key Lab of Intelligent Tech. & Sys.
Tsinghua National Lab for Info. Sci. and Tech.
Dept. of Comp. Sci. & Tech., Tsinghua Univ.
zhaizhongwu@gmail.com

[‡]Dept. of Comp. Sci.
University of Illinois at Chicago
liub@cs.uic.edu

Abstract

In opinion mining of product reviews, one often wants to produce a summary of opinions based on product features/attributes. However, for the same feature, people can express it with different words and phrases. To produce a meaningful summary, these words and phrases, which are domain synonyms, need to be grouped under the same feature group. This paper proposes a constrained semi-supervised learning method to solve the problem. Experimental results using reviews from five different domains show that the proposed method is competent for the task. It outperforms the original EM and the state-of-the-art existing methods by a large margin.

1 Introduction

One form of opinion mining in product reviews is to produce a feature-based summary (Hu and Liu, 2004a; Liu, 2010). In this model, product features are first identified, and positive and negative opinions on them are aggregated to produce a summary on the features. Features of a product are attributes, components and other aspects of the product, e.g., “picture quality”, “battery life” and “zoom” of a digital camera.

In reviews (or any writings), people often use different words and phrases to describe the same product feature. For example, “picture” and “photo” refer to the same feature for cameras. Grouping such synonyms is critical for effective opinion summary. Although WorldNet and other thesaurus dictionaries can help to some extent, they are far from sufficient due to a few reasons. First, many words and phrases that are not synonyms in a dictionary may refer to the same feature in an application domain. For example, “appearance” and “design” are not synonymous, but they can indicate the same feature, *design*. Second, many synonyms are domain dependent. For example, “movie” and “picture” are synonyms in movie reviews, but they are not synonyms in camera reviews as “picture” is more likely to be synonymous to “photo” while “movie” to “video”. Third, determining which expressions indicate the same feature can be dependent on the user’s application need. For example, in car reviews, internal design and external design can be regarded as two separate features, but can also be regarded as one feature, called “design”, based to the level of details that the user needs to study. In camera reviews, one may want to study battery as a whole (one feature), or as more than one feature, e.g., battery weight, and battery life. Due to this reason, in applications the user needs to be involved in synonym grouping.

Before going further, let us introduce two concepts, *feature group* and *feature expression*. Feature group (or *feature* for short) is the name of a feature (given by the user), while a feature expression of a feature is a word or phrase that actually appears in a review to indicate the feature. For example, a feature group could be named “picture quality”, but there are many possible expressions indicating the feature, e.g., “picture”, “photo”, “image”, and even the “picture quality” itself. All the feature expressions in a feature group signify the same feature.

Grouping feature expressions manually into suitable groups is time consuming as there are

*Supported by National Natural Science Foundation of China (Grant No: 60875073).

This work was done when the first author was visiting Bing Liu’s group at the University of Illinois at Chicago.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1272–1280,
Beijing, August 2010

often hundreds of feature expressions. This paper helps the user to perform the task more efficiently. To focus our research, we assume that feature expressions have been discovered from a review corpus by an existing system such as those in (Hu and Liu, 2004b; Popescu and Etzioni, 2005; Kim and Hovy, 2006; Kobayashi *et al.*, 2007; Mei *et al.*, 2007; Stoyanov and Cardie, 2008; Jin *et al.*, 2009; Ku *et al.*, 2009).

To reflect the user needs, he/she can manually label a small number of seeds for each feature group. The feature groups are also provided by the user based on his/her application needs. The system then assigns the rest of the feature expressions to suitable groups. To the best of our knowledge, this problem has not been studied in opinion mining (Pang and Lee, 2008).

The problem can be formulated as semi-supervised learning. The small set of seeds labeled by the user is the labeled data, and the rest of the discovered feature expressions are the unlabeled data. This is the transductive setting (Joachims, 1999) because the unlabeled set is used in learning and also in testing since our objective is to assign unlabeled expressions to the right feature groups.

Any semi-supervised learning method can be applied to tackle the problem. In this work, we use the Expectation-Maximization (EM) algorithm (Dempster *et al.*, 1977). Specifically, we use the naïve Bayesian EM formulation in (Nigam *et al.*, 2000), which runs a Bayesian classifier iteratively on the labeled and unlabeled data until the probabilities for the unlabeled data converge. When the algorithm ends, each unlabeled example is assigned a posterior probability of belonging to each group.

However, we can do better since the EM algorithm only achieves local optimal. What local optimal it achieves depends on the initialization, i.e., the initial seeds. We show that some prior knowledge can help provide a better initialization, and consequently generate better grouping results. Thus, we propose to create another set of data extracted from the unlabeled set based on two pieces of natural language knowledge:

1. Feature expressions sharing some common words are likely to belong to the same group, e.g., “battery life” and “battery power”.
2. Feature expressions that are synonyms in a dictionary are likely to belong to the same group, e.g., “movie” and “picture”.

We call these two pieces of prior knowledge *soft constraints* because they constrain the feature expressions to be in the same feature group. The constraints are soft (rather than hard) as they can be relaxed in the learning process. This relaxation is important because the above two constraints can result in wrong groupings. The EM algorithm is allowed to re-assign them to other groups in the learning process.

We call the proposed framework constrained semi-supervised learning. Since we use EM and soft constraints, we call the proposed method *SC-EM.* Clearly, the problem can also be attempted using some other techniques, e.g., topic modeling (e.g, LDA (Blei *et al.*, 2003)), or clustering using distributional similarity (Pereira *et al.*, 1993; Lin, 1998; Chen *et al.*, 2006; Sahami and Heilman, 2006). However, our results show that these methods do not perform as well.

The input to the proposed algorithm consists of: a set of reviews *R*, and a set of discovered feature expressions *F* from *R* (using an existing algorithm). The user labels a small set of feature expressions, i.e., assigning them to the user-specified feature groups. The system then assigns the rest of the discovered features to the feature groups. EM is run using the distributional (or surrounding words) contexts of feature expressions in review set *R* to build a naïve Bayesian classifier in each iteration.

Our evaluation was conducted using reviews from 5 different domains (insurance, mattress, vacuum, car and home-theater). The results show that the proposed method outperforms different variations of the topic modeling method LDA, *k*-means clustering, and the recent unsupervised feature grouping method mLSA.

In summary, this paper makes three main contributions:

1. It proposes a new sub-problem of opinion mining, i.e., grouping feature expressions in the context of semi-supervised learning. Although there are existing methods for solving the problem based on unsupervised learning, we argue that for practical use some form of supervision from the user is necessary to let the system know what the user wants.
2. An EM formulation is used to solve the problem. We augment EM with two soft constraints. These constraints help guide EM to

produce better solutions. We note that these constraints can be relaxed in the process to correct the imperfection of the constraints.

3. It is shown experimentally the new method outperforms the main existing state-of-the-art methods that can be applied to the task.

2 Related Work

This work is mainly related to existing research on synonyms grouping, which clusters words and phrases based on some form of similarity.

The methods for measuring word similarity can be classified into two main types (Agirre *et al.*, 2009): those relying on *pre-existing knowledge resources* (e.g., thesauri, or taxonomies) (Yang and Powers, 2005; Alvarez and Lim, 2007; Hughes and Ramage, 2007), and those based on *distributional properties* (Pereira *et al.*, 1993; Lin, 1998; Chen *et al.*, 2006; Sahami and Heilman, 2006; Pantel *et al.*, 2009).

In the category that relies on existing knowledge sources, the work of Carenini *et al.* (2005) is most related to ours. The authors proposed a method to map feature expressions to a given domain feature taxonomy, using several similarity metrics on WordNet. This work does not use the word distribution information, which is its main weakness because many expressions of the same feature are not synonyms in WordNet as they are domain/application dependent. Dictionaries do not contain domain specific knowledge, for which a domain corpus is needed.

Another related work is distributional similarity, i.e., words with similar meaning tend to appear in similar contexts (Harris, 1968). As such, it fetches the surrounding words as context for each term. Similarity measures such as *Cosine*, *Jaccard*, *Dice*, etc (Lee, 1999), can be employed to compute the similarities between the seeds and other feature expressions. To suit our need, we tested the *k*-means clustering with distributional similarity. However, it does not perform as well as the proposed method.

Recent work also applied topic modeling (e.g., LDA) to solve the problem. Guo *et al.* (2009) proposed a multilevel latent semantic association technique (called *mLSA*) to group product feature expressions, which runs LDA twice. However, *mLSA* is an unsupervised approach. For our evaluation, we still implemented the method and compared it with our SC-EM method.

Our work is also related to constrained clustering (Wagstaff *et al.*, 2001), which uses two forms of constraints, must-link and cannot-link. Must-links state that some data points must be in the same cluster, and cannot-links state that some data points cannot be in the same cluster. In (Andrzejewski *et al.*, 2009), the two constraints are added to LDA, called *DF-LDA*. We show that both these methods do not perform as well as our semi-supervised learning method *SC-EM.*

3 The Proposed Algorithm

Since our problem can be formulated as semi-supervised learning, we briefly describe the setting in our context. Given a set C of classes (our feature groups), we use L to denote the small set of labeled examples (labeled feature expressions or seeds), and U the set of unlabeled examples (unlabeled feature expressions). A classifier is built using L and U to classify every example in U to a class. Several existing algorithms can be applied. In this work, we use EM as it is efficient and it allows prior knowledge to be used easily. Below, we first introduce the EM algorithm that we use, and then present our augmented EM. The constraints and their conflict handling are discussed in Section 4.

3.1 Semi-Supervised Learning Using EM

EM is a popular iterative algorithm for maximum likelihood estimation in problems with missing data. In our case, the group memberships of the unlabeled expressions are considered missing because they come without group labels.

We use the EM algorithm based on naïve Bayesian classification (Nigam *et al.*, 2000). Although it is involved to derive, using it is simple. First, a classifier f is learned using only the labeled data L (Equations 1 and 2). Then, f is applied to assign a probabilistic label to each unlabeled example in U (see Equation 3). Next, a new classifier f is learned using both L and the newly probabilistically labeled unlabeled examples in U_{PL}, again using Equations 1 and 2. These last two steps iterate until convergence.

We now explain the notations in the Equations. Given a set of training documents D, each document d_i in D is considered as an ordered list of words. $w_{d_i,k}$ denotes the k^{th} word in d_i, where each word is from the vocabulary $V=\{w_1, w_{2,\dots,} w_{|V|}\}$. $C=\{c_1, c_{2,\dots,} c_{|C|}\}$ is the set of pre-defined

$$\mathrm{P}(w_t|c_j) = \frac{1 + \sum_{i=1}^{|D|} N_{ti}\mathrm{P}(c_j|d_i)}{|V| + \sum_{m=1}^{|V|}\sum_{i=1}^{|D|} N_{mi}\mathrm{P}(c_j|d_i)} \tag{1^1}$$

$$\mathrm{P}(c_j) = \frac{1 + \sum_{i=1}^{|D|} \mathrm{P}(c_j|d_i)}{|C| + |D|} \tag{2^1}$$

$$\mathrm{P}(c_j|d_i) = \frac{\mathrm{P}(c_j)\prod_{\mathrm{k}=1}^{|d_i|}\mathrm{P}(w_{d_i,k}|c_j)}{\sum_{\mathrm{r}=1}^{|C|}\mathrm{P}(c_r)\prod_{\mathrm{k}=1}^{|d_i|}\mathrm{P}(w_{d_i,k}|c_r)} \tag{3}$$

classes or groups. N_{ti} is the number of times the word w_t occurs in document d_i.

For our problem, the surrounding words contexts of the labeled seeds form *L*, while the surrounding words of the non-seed feature expressions form *U*. When EM converges, the classification labels of the unlabeled feature expressions give us the final grouping. Surrounding words contexts will be discussed in Section 5.

3.2 Proposed Soft-Constrained EM

Although EM can be directly applied to deal with our problem, we can do better. As we discussed earlier, EM only achieves local optimal based on the initialization, i.e., the labeled examples or seeds. We show that natural languages constraints can be used to provide a better initialization, i.e., to add more seeds that are likely to be correct, called *soft-labeled examples* or *soft seeds* (*SL*). Soft-labeled examples are handled differently from the original labeled examples in *L*. With the soft seeds, we have the proposed soft-constrained EM (called SC-EM).

Compared with the original EM, SC-EM has two main differences:

- Soft constraints are applied to *L* and *U* to produce a set *SL* of soft-labeled examples (or soft seeds) to initialize EM in addition to *L*. *SL* is thus a subset of *U*. The training set size is increased, which helps produce better results as our experimental results show.
- In the first iteration of EM, soft-labeled examples *SL* are treated in the same way as the labeled examples in *L*. Thus both *SL* and *L* are used as labeled examples to learn the initial classifier f_0. However, in the subsequent iterations, *SL* is treated in the same way as any examples in *U*. That is, the classifier f_x from each iteration *x* (including f_0) will predict *U*. After that, a new classifier is built using both *L* and U_{PL} (which is *U* with probabilistic labels). Clearly, this implies that the class labels of the examples in *SL* are allowed to change. That is also why we call *SL* the soft-labeled set in contrast to the hard-labeled set *L*, i.e., the examples in *L* will not change labels in EM. The reason that *SL* is allowed to change labels/classes is because the constraints can make mistakes. EM may be able to correct some of the mistakes.

```
Input:
-  Labeled examples $L$
-  Unlabeled examples $U$
Extract $SL$ from $U$ using constraints (Section 4);
Learn an initial naïve Bayesian classifier $f_0$ using $L \cup SL$ and Equations 1 and 2;
repeat
    // E-Step
    for each example $d_i$ in $U$ (including $SL$) do
        Using the current classifier $f_x$ to compute $P(c_j|d_i)$ using Equation 3.
    end
    // M-Step
    Learn a new naïve Bayesian classifier $f_x$ from $L$ and $U$ by computing $P(w_t|c_j)$ and $P(c_j)$ using Equations 1 and 2.
until the classifier parameters stabilize
Output: the classifier $f_x$ from the last iteration.
```

Figure 1. The proposed SC-EM algorithm

The detailed algorithm is given in Figure 1. The constraints are discussed in Section 4.

4 Generating SL Using Constraints

As mentioned earlier, two forms of constraints are used to induce the soft-labeled set *SL*. For easy reference, we reproduce them here:

1. Feature expressions sharing some common words are likely to belong to the same group.
2. Feature expressions that are synonyms in a dictionary are likely to belong to one group.

According to the number of words, feature expressions can be categorized into single-word expressions and phrase expressions. They are handled differently. The detailed algorithm is given in Figure 2. In the algorithm, *L* is the labeled set and *U* is the unlabeled set. *L*, in fact, consists of a set of sets, $L = \{L_1, L_2, \ldots, L_{|L|}\}$. Each L_i contains a set of labeled examples (feature expressions) of the i^{th} class (feature group). Similarly, the output set *SL* (the soft-labeled set) also consists of a set of sets, i.e., $SL = \{SL_1, SL_2, \ldots, SL_{|L|}\}$. Each SL_i is a set of soft-labeled examples (feature expressions) of the i^{th} class

[1] Laplace smoothing is used to prevent zero probabilities for infrequently occurring words.

(feature group). Thus L_i and SL_i correspond to each other as they represent the original labeled examples and the newly soft-labeled examples of the i^{th} class (or feature group) respectively.

The algorithm basically compares each feature expression u in U (line 1) with each feature expression e (line 4) in every labeled subset L_i (line 2) based on the above two constraints. If any of the constraints is satisfied (lines 5-17), it means that u is likely to belong to L_i (or the i^{th} class or feature group), and it is added to SL_i.

There are conflict situations that need to be resolved. That is, u may satisfy a constraint of more than one labeled sub-set L_i. For example, if u is a single word, it may be synonyms of feature expressions from more than one feature groups. The question is which group it is likely to belong. Further, u may be synonyms of a few single-word feature expressions in L_i. Clearly, u being a synonym of more than one word in L_i is better than it is only the synonym of one word in L_i. Similar problems also occur when u is an element of a feature expression phrase e.

To match u and e, there are a few possibilities. If both u and e are single words (lines 5-6), the algorithm checks if they are synonyms (line 7). The score in line 8 is discussed below. When one of u and e is a phrase, or both of them are phrases, we see whether they have shared words. Again, conflict situations can happen with multiple classes (feature groups) as discussed above. Note that in these cases, we do not use the synonym constraint, which does not help in our test.

Given these complex cases, we need to decide which class that u should be assigned to or should not be assigned to any class (as it does not meet any constraint). We use a score to record the level of satisfaction. Once u is compared with each e in every class, the accumulated score is used to determine which class L_i has the strongest association with u. The class j with the highest score is assigned to u. In other words, u is added to SL_j. Regarding the score value, synonyms gets the score of 1 (line 8), and intersection (shared words) gets the score equal to the size of the intersection (lines 10-17).

```
1  for each feature expression $u \in U$ do
2      for each feature group $L_i \in L$ do
3          score($L_i$) ← 0;
4          for each feature expression $e \in L_i$ do
5              if $u$ is a single word expression then
6                  if $e$ is a single word expression then
7                      if $u$ and $e$ are synonyms then
8                          score($L_i$) ← score($L_i$) + 1;
9                  else if $w \in e$ then  // $e$ is a phrase
10                     score($L_i$) ← score($L_i$) + 1
11             else  // $u$ is a phrase
12                 if $e$ is a single word expression then
13                     if $e \in u$ then  // $u$ is a phrase
14                         score($L_i$) ← score($L_i$) + 1
15                 else
16                     $s \leftarrow e \cap u$;
17                     score($L_i$) ← score($L_i$) + $|s|$
18     $u$ is added to $SL_j$ s.t. $\text{argmax}_{L_i}\, score(L_i)$
```

Figure 2. Generating the soft-labeled set SL

5 Distributional Context Extraction

To apply the proposed algorithm, a document d_i needs to be prepared for each feature expression e_i for naïve Bayesian learning. d_i is formed by aggregating the distributional context of each sentence s_{ij} in our corpus that contains the expression e_i. The context of a sentence is the surrounding words of e_i in a text window of [-t, t], including the words in e_i. Given a relevant corpus R, the document d_i for each feature expression e_i in L (or U) is generated using the algorithm in Figure 3. Stopwords are removed.

```
for each feature expression $e_i$ in $L$ (or $U$) do
    $S_i$ ← all sentences containing $e_i$ in $R$;
    for each sentence $s_{ij} \in S_i$ do
        $d_{ij}$ ← words in a window of [-$t$, $t$] on the left
                and right (including the words in $e_i$);
    $d_i$ ← words from all $d_{ij}$, $j$ = 1, 2, …, $|S_i|$;
                // duplicates are kept as it is not union
```

Figure 3. Distributional context extraction

For example, a feature expression from L (or U) is e_i = "*screen*" and there are two sentences in our corpus R that contain "*screen*"

s_{i1} = "*The LCD* ***screen*** *gives clear picture*".

s_{i2} = "*The picture on the* ***screen*** *is blur*"

We use the window size of [-3, 3]. Sentence s_{i1}, gives us d_{i1} = <LCD, screen, give, clear, picture> as a bag of words. "the" and "is" are removed as stopwords. s_{i2} gives us d_{i2} = <picture, screen, blur>. "on", "the" and "is" are removed as stopwords. Finally, we obtain the document d_i for feature expression e_i as a bag of words:

d_i = <LCD, screen, give, clear, picture,
picture, screen, blur>

6 Empirical Evaluation

This section evaluates the SC-EM algorithm and compares it with the main existing methods that can be applied to solve the problem.

6.1 Review Data Sets and Gold Standards

To demonstrate the generality of the proposed method, experiments were conducted using reviews from five domains: *Hometheater, Insurance, Mattress, Car* and *Vacuum*. All the data sets and the *gold standard* feature expressions and groups were from a company that provides opinion mining services. The details of the data sets and the gold standards are given in Table 1.

	Hometheater	Insurance	Mattress	Car	Vacuum
#Sentences	6355	12446	12107	9731	8785
#Reviews	587	2802	933	1486	551
#Feature expressions	237	148	333	317	266
#Feature groups	15	8	15	16	28

Table 1. Data sets and gold standards

6.2 Evaluation Measures

Since SC-EM is based on semi-supervised learning, we can use classification accuracy to evaluate it. We can also see it as clustering with initial seeds. Thus we also use clustering evaluation methods. Given gold standards, two popular clustering evaluation measures are *Entropy* and *Purity* (Liu, 2006). As *accuracy* is fairly standard, we will not discuss it further. Below, we briefly describe entropy and purity.

Given a data set *DS*, its gold partition is $G = \{g_1,\dots, g_j,\dots, g_k\}$, where *k* is the known number of clusters. The groups partition *DS* into *k* disjoint subsets, $DS_1,\dots, DS_i, \dots, DS_k$.

Entropy: For each resulting cluster, we can measure its entropy using Equation 4, where $\mathrm{P}_i(g_j)$ is the proportion of g_j data points in DS_i. The total entropy of the clustering (considering all clusters) is calculated by Equation 5.

$$entropy(DS_i) = -\sum\nolimits_{j=1}^{k} \mathrm{P}_i(g_j) log_2 \mathrm{P}_i(g_j) \qquad (4)$$

$$entropy_{total} = \sum\nolimits_{i=1}^{k} \frac{|DS_i|}{|DS|} entropy(DS_i) \qquad (5)$$

Purity: Purity measures the extent that a cluster contains only data from one gold-partition. Each cluster's purity is computed by Equation 6, and the total purity of the whole clustering is computed with Equation 7.

$$purity(DS_i) = \max_j \mathrm{P}_i(g_j) \qquad (6)$$

$$purity_{total} = \sum\nolimits_{i=1}^{k} \frac{|DS_i|}{|DS|} purity(DS_i) \qquad (7)$$

In testing, the unlabeled set *U* is also our test set. This is justified because our purpose is to assign unlabeled data to appropriate groups.

6.3 Baseline Methods and Settings

The proposed ***SC-EM*** method is compared with a set of existing methods, which can be categorized into unsupervised and semi-supervised methods. We list the *unsupervised* methods first.

LDA: *LDA* is a popular topic modeling method (see Section 2). Given a set of documents, it outputs groups of terms of different topics. In our case, each feature expression is a term, and the documents refer to the distributional contexts of each feature expressions (see Section 5).

mLSA: This is a state-of-the-art unsupervised method for solving the problem. It is based on LDA, and has been discussed in related work.

Kmeans: This is the *k*-means clustering method (MacQueen, 1966) based on distributional similarity with cosine as the similarity measure.

In the *semi-supervised* category, the methods are further classified into un-constrained, hard-constrained, and soft-constrained methods.

For the *un-constrained* subclass (no constraints are used), we have the following:

LDA*(*L, H*)**: This method is based on ***LDA, but the labeled examples *L* are used as seeds for each group/topic. All examples in *L* will always stay in the same topic. We call this hard initialization (*H*). *L* is handled similarly below.

DF-LDA*(*L, H*)**. *DF-LDA* is the *LDA* method (Andrzejewski *et al.*, 2009) that takes must-links and cannot-links. Our *L* set can be expressed as a combination of must-links and cannot-links. Unfortunately, only must-links can be used because the number of cannot-links is huge and crashes the system. For example, for the car data, the number of cannot-links is 194,400 for 10% labeled data (see Section 6.4) and for 20% it is 466,560,000. DF-LDA also has a parameter η controlling the link strength, which is set very high (=1000) to reflect the hard initialization. We did not use *DF-LDA* in the unsupervised subclass above as without constraints it reduces to ***LDA.

***Kmeans*(*L, H*)**: This method is based on *Kmeans*, but the clusters of the labeled seeds are fixed at the initiation and remain unchanged.

***EM*(*L, H*)**: This is the original EM for semi-supervised learning. Only the labeled examples are used as the initial seeds.

For the *hard-constrained* (*H*) subclass (our two constraints are applied and cannot be violated), we have the following methods (*LC* is *L* plus *SL* produced by the constraints (C):

***Rand*(*LC, H*):** This is an important baseline. It shows whether the constraints alone are sufficient to produce good results. That is, the final result is the expanded seeds *SL* plus the rest of *U* assigned randomly to different groups.

***LDA*(*LC, H*)**: It is similar to *LDA*(*L,H*), but both the initial seeds *L* and the expanded seeds *SL* are considered as labeled examples. They also stay in the same topics/groups in the process. Note that although *SL* is called a set of soft-labeled examples (seeds) in the proposed algorithm, they are treated as hard-labeled examples here just for experimental comparison.

***DF-LDA*(*LC, H*)**: This is *DF-LDA* with both *L* and *SL* expressed as must-links. Again, a large η (= 1000) is used to make sure that must-links for *L* and *SL* will not be violated.

***Kmeans*(*LC,H*)**: It is similar to *Kmeans*(*L,H*), but both *L* and *SL* stay in their assigned clusters.

***EM*(*LC, H*)**: It is similar to *SC-EM*, but *SL* is added to the labeled set *L*, and their classes are not allowed to change in the EM iterations.

For the *soft-constrained* (*S*) subclass, our two constraints can be violated. Initially, both the initial seeds *L* and the expanded seeds *SL* are considered as labeled data, but subsequently, only *L* is taken as the labeled data (i.e., staying in the same classes). The algorithm will re-estimate the label of each feature expression in *SL*. This subclass has the following methods:

***LDA*(*LC, S*)**: This is in contrast to *LDA*(*LC, H*). It allows the *SL* set to change topics/groups.

***Kmeans*(*LC, S*)**: This is in contrast to *Kmeans*(*LC, H*).

A soft *DF-LDA* is not included here because different η values give different results, and they are generally worse than *DF-LDA*(*LC, H*).

For all *LDA* based methods, the topic modeling parameters were set to their default values. The number of iteration is 1000. We used the *LDA* in MALLET[2], and modified it to suit different *LDA*-based methods except *DF-LDA*, which was downloaded from its authors' website[3]. We implemented *mLSA*, *Kmeans* and changed EM[4] to take soft seeds. For all *Kmeans* based methods, the distance function is the cosine similarity.

6.4 Evaluation Results

We now compare the results of *SC-EM* and the 14 baseline methods. To see the effects of different numbers of labeled examples (seeds), we experimented with 10%, 20%, 30%, 40%, and 50% of the feature expressions from the gold standard data as the labeled set *L*, and the rest as the unlabeled set *U*. All labeled data were selected randomly. For each setting, we run the algorithms 30 times and report the average results. Due to space limitations, we can only show the detailed *purity* (Pur), *entropy* (Ent) and *accuracy* (Acc) results for 30% as the labeled data (70% as unlabeled) in Table 2. For the other proportions of labeled data, we summarize them in Table 3. Each result in Table 3 is thus the average of the 5 data sets. All the results were obtained from the unlabeled set *U*, which was our test set. For entropy, the smaller the value is the better, but for purity and accuracy, the larger the better. For these experiments, we used the window size $t = 5$. Section 6.5 studies the effects of window sizes.

Tables 2 and 3 clearly show that the proposed algorithm (***SC-EM***) outperforms all 14 baseline methods by a large margin on every dataset. In detail, we observe the following:

- *LDA*, *mLSA* and *Kmeans* with no seeds (labeled data) perform the worst. Seeds help to improve the results, which is intuitive. Without seeds, *DF-LDA* is the same as *LDA*.
- *LDA* based methods seems to be the weakest. *Kmeans* based methods are slightly better, but EM based methods are the best. This clearly indicates that classification (EM) performs better than clustering. Comparing *DF-LDA* and *Kmeans*, their results are similar.
- For *LDA*, and *Kmeans*, hard-constrained methods (i.e., *LDA*(*L, H*), and *Kmeans*(*L, H*)) perform better than soft-constrained methods (i.e., *LDA*(*LC, S*) and *Kmeans*(*LC, S*)). This indicates that soft-constrained versions may change some correctly constrained expressions into wrong groups. However, for the EM based methods, the soft-constrained method (*SC-EM*) performs markedly better than the hard-constrained version (*EM*(*LC, H*)). This indicates that Bayesian classifier used in EM can take advantage of the soft constraints and correct some wrong assignments made by constraints. Much weaker results of *Rand*(*LC, H*) than *SC-EM* in different settings show that

[2] http://mallet.cs.umass.edu/

[3] http://pages.cs.wisc.edu/~andrzeje/research/df_lda.html

[4] http://alias-i.com/lingpipe/

Methods	Hometheater			Insurance			Mattress			Car			Vacuum		
	Acc	Pur	Ent	Acc	Pur	Ent	Acc	Pur	Ent	Acc	Pur	Ent	Acc	Pur	Ent
LDA	0.06	0.31	2.54	0.11	0.36	2.24	0.05	0.32	2.57	0.06	0.37	2.39	0.03	0.36	2.09
mLSA	0.06	0.31	2.53	0.14	0.38	2.19	0.06	0.34	2.55	0.09	0.37	2.40	0.03	0.37	2.11
Kmeans	**0.21**	**0.42**	**2.14**	**0.25**	**0.45**	**1.90**	**0.15**	**0.39**	**2.32**	**0.25**	**0.44**	**2.16**	**0.24**	**0.47**	**1.78**
LDA(L, H)	0.10	0.32	2.50	0.16	0.37	2.22	0.10	0.34	2.57	0.19	0.39	2.36	0.10	0.39	2.09
DF-LDA(L, H)	0.27	0.37	2.32	0.25	0.41	2.00	0.19	0.39	2.35	0.28	0.45	2.15	0.31	0.40	1.98
Kmeans(L, H)	0.20	0.42	2.12	0.25	0.43	1.92	0.17	0.42	2.26	0.27	0.48	2.04	0.20	0.48	1.76
EM(L, H)	**0.48**	**0.50**	**1.93**	**0.50**	**0.53**	**1.69**	**0.52**	**0.56**	**1.87**	**0.56**	**0.58**	**1.80**	**0.49**	**0.52**	**1.79**
Rand(CL, H)	0.41	0.46	2.07	0.40	0.46	1.94	0.40	0.47	2.07	0.34	0.41	2.31	0.39	0.52	1.59
LDA(CL, H)	0.44	0.50	1.96	0.42	0.48	1.89	0.42	0.49	1.97	0.44	0.52	1.87	0.43	0.55	1.48
DF-LDA(CL, H)	0.35	0.49	1.86	0.33	0.49	1.71	0.23	0.39	2.26	0.34	0.51	1.88	0.37	0.52	1.58
Kmeans(CL, H)	0.49	0.55	1.70	0.48	0.55	1.62	0.44	0.51	1.91	0.47	0.54	1.80	0.44	0.58	1.42
EM(CL, H)	**0.59**	**0.60**	**1.62**	**0.58**	**0.60**	**1.46**	**0.56**	**0.59**	**1.74**	**0.62**	**0.64**	**1.54**	**0.55**	**0.60**	**1.44**
LDA(CL, S)	0.24	0.35	2.44	0.27	0.40	2.14	0.23	0.37	2.44	0.27	0.41	2.33	0.23	0.41	2.01
Kmeans(CL, S)	0.33	0.46	2.04	0.34	0.45	1.90	0.25	0.43	2.20	0.29	0.47	2.07	0.37	0.50	1.68
SC-EM	**0.67**	**0.68**	**1.30**	**0.66**	**0.68**	**1.18**	**0.68**	**0.70**	**1.27**	**0.70**	**0.71**	**1.24**	**0.67**	**0.68**	**1.18**

Table 2. Comparison results (*L* = 30% of the gold standard data)

Methods	Acc					Pur					Ent				
	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%
LDA	0.07	0.07	0.06	0.06	0.08	0.33	0.33	0.34	0.35	0.38	2.50	2.44	2.37	2.28	2.11
mLSA	0.07	0.07	0.08	0.07	0.07	0.34	0.35	0.35	0.37	0.38	2.48	2.42	2.36	2.26	2.12
Kmeans	**0.22**	**0.23**	**0.22**	**0.22**	**0.22**	**0.42**	**0.43**	**0.44**	**0.44**	**0.46**	**2.16**	**2.11**	**2.06**	**1.98**	**1.86**
LDA(L, H)	0.10	0.10	0.13	0.14	0.15	0.34	0.34	0.36	0.37	0.39	2.48	2.43	2.35	2.25	2.11
DF-LDA(L, H)	0.23	0.25	0.26	0.27	0.30	0.41	0.40	0.41	0.41	0.44	2.23	2.23	2.16	2.10	1.94
Kmeans(L, H)	0.13	0.16	0.22	0.24	0.28	0.42	0.43	0.45	0.45	0.48	2.15	2.11	2.02	1.95	1.79
EM(L, H)	**0.35**	**0.44**	**0.51**	**0.55**	**0.58**	**0.43**	**0.49**	**0.54**	**0.57**	**0.61**	**2.22**	**1.99**	**1.81**	**1.65**	**1.49**
Rand(CL, H)	0.28	0.35	0.39	0.42	0.45	0.39	0.43	0.47	0.50	0.54	2.33	2.15	2.00	1.82	1.63
LDA(CL, H)	0.31	0.38	0.43	0.46	0.49	0.43	0.47	0.51	0.54	0.58	2.16	1.99	1.83	1.69	1.49
DF-LDA(CL, H)	0.32	0.33	0.33	0.34	0.36	0.49	0.50	0.48	0.48	0.48	1.90	1.85	1.86	1.83	1.82
Kmeans(CL, H)	0.33	0.41	0.46	0.49	0.52	0.47	0.51	0.55	0.57	0.61	1.98	1.82	1.69	1.56	1.42
EM(CL, H)	**0.44**	**0.54**	**0.58**	**0.61**	**0.64**	**0.49**	**0.57**	**0.61**	**0.64**	**0.67**	**1.98**	**1.72**	**1.56**	**1.40**	**1.25**
LDA(CL, S)	0.17	0.21	0.25	0.30	0.34	0.34	0.36	0.39	0.42	0.46	2.47	2.37	2.27	2.09	1.87
Kmeans(CL, S)	0.23	0.28	0.32	0.36	0.42	0.43	0.44	0.46	0.48	0.51	2.15	2.08	1.98	1.86	1.70
SC-EM	**0.45**	**0.58**	**0.68**	**0.75**	**0.81**	**0.50**	**0.61**	**0.69**	**0.76**	**0.82**	**1.95**	**1.56**	**1.24**	**0.94**	**0.69**

Table 3. Influence of the seeds' proportion (which reflects the size of the labeled set *L*)

constraints alone (i.e., synonyms and sharing of words) are far from sufficient. EM can improve it considerably.

- Comparing EM based methods, we can see that soft seeds in *SL* make a big difference for all data sets. *SC-EM* is clearly the best.
- As the number of labeled examples increases (from 10% to 50%), the results improve for every method (except those for *DF-LDA*, which does not change much).

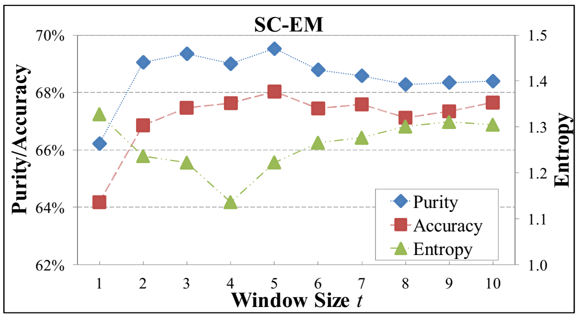


Figure 4. Influence of context window size

6.5 Varying the Context Window Size

We varied the text window size *t* from 1 to 10 to see how it impacts on the performance of *SC-EM*. The results are given in Figure 4 (they are averages of the 5 datasets). Again for purity and accuracy, the greater the value the better, while for entropy it is the opposite. It is clear that the window sizes of 2~6 produce similar good results. All evaluations reported above used *t* = 5.

7 Conclusion

This paper proposed the task of feature grouping in a semi-supervised setting. It argued that some form of supervision is needed for the problem because its solution depends on the user application needs. The paper then proposed to use the EM algorithm to solve the problem, which was improved by considering two soft constraints. Empirical evaluations using 5 real-life data sets show that the proposed method is superior to 14 baselines. In our future work, we will focus on further improving the accuracy.

References

Agirre E., E. Alfonseca, K. Hall, J. Kravalova, M. Pa ca and A. Soroa 2009. *A study on similarity and relatedness using distributional and WordNet-based approaches*. Proceedings of ACL.

Alvarez M. and S. Lim 2007. *A Graph Modeling of Semantic Similarity between Words*. Proceeding of the Conference on Semantic Computing.

Andrzejewski D., X. Zhu and M. Craven 2009. *Incorporating domain knowledge into topic modeling via Dirichlet forest priors*. Proceedings of ICML.

Blei D., A. Y. Ng and M. I. Jordan 2003. "*Latent Dirichlet Allocation*." JMLR 3: 993-1022.

Carenini G., R. Ng and E. Zwart 2005. *Extracting knowledge from evaluative text*. Proceedings of International Conference on Knowledge Capture.

Chen H., M. Lin and Y. Wei 2006. *Novel association measures using web search with double checking*. Proceedings of ACL.

Dempster A., N. Laird and D. Rubin 1977. "*Maximum likelihood from incomplete data via the EM algorithm*." Journal of the Royal Statistical Society 39(1): 1-38.

Guo H., H. Zhu, Z. Guo, X. Zhang and Z. Su 2009. *Product feature categorization with multilevel latent semantic association*. Proc. of CIKM.

Harris Z. S. 1968. *Mathematical structures of language*. New York, Interscience Publishers.

Hu M. and B. Liu 2004a. *Mining and summarizing customer reviews*. Proceedings of SIGKDD.

Hu M. and B. Liu 2004b. *Mining Opinion Features in Customer Reviews*. Proceedings of AAAI.

Hughes T. and D. Ramage 2007. *Lexical semantic relatedness with random graph walks*. EMNLP.

Jin W., H. Ho and R. Srihari 2009. *OpinionMiner: a novel machine learning system for web opinion mining and extraction*. Proceedings of KDD.

Joachims T. 1999. *Transductive inference for text classification using support vector machines*. Proceedings of ICML.

Kim S. and E. Hovy 2006. *Extracting opinions, opinion holders, and topics expressed in online news media text*. Proceedings of EMNLP.

Kobayashi N., K. Inui and Y. Matsumoto 2007. *Extracting aspect-evaluation and aspect-of relations in opinion mining*. Proceedings of EMNLP.

Ku L., H. Ho and H. Chen 2009. "*Opinion mining and relationship discovery using CopeOpi opinion analysis system*." Journal of the American Society for Information Science and Technology 60(7): 1486-1503.

Lee L. 1999. *Measures of distributional similarity*, Proceedings of ACL.

Lin D. 1998. *Automatic retrieval and clustering of similar words*, Proceedings of ACL.

Liu B. 2006. *Web data mining; Exploring hyperlinks, contents, and usage data*, Springer.

Liu B. 2010. *Sentiment Analysis and Subjectivity*. *Handbook of Natural Language Processing* N. Indurkhya and F. J. Damerau.

MacQueen J. 1966. *Some methods for classification and analysis of multivariate observations*. Proc. of Symposium on Mathematical Statistics and Probability.

Mei Q., X. Ling, M. Wondra, H. Su and C. Zhai 2007. *Topic sentiment mixture: modeling facets and opinions in weblogs*. Proceedings of WWW.

Nigam K., A. McCallum, S. Thrun and T. Mitchell 2000. "*Text classification from labeled and unlabeled documents using EM*." Machine Learning 39(2).

Pang B. and L. Lee 2008. "*Opinion mining and sentiment analysis*." Foundations and Trends in Information Retrieval 2(1-2): 1-135.

Pantel P., E. Crestan, A. Borkovsky, A. Popescu and V. Vyas 2009. *Web-scale distributional similarity and entity set expansion*. EMNLP.

Pereira F., N. Tishby and L. Lee 1993. *Distributional clustering of English words*. Proceedings of ACL.

Popescu A.-M. and O. Etzioni 2005. *Extracting Product Features and Opinions from Reviews*. EMNLP.

Sahami M. and T. Heilman 2006. *A web-based kernel function for measuring the similarity of short text snippets*. Proceedings of WWW.

Stoyanov V. and C. Cardie 2008. *Topic identification for fine-grained opinion analysis*. COLING.

Wagstaff K., C. Cardie, S. Rogers and S. Schroedl 2001. *Constrained k-means clustering with background knowledge*. In Proceedings of ICML.

Yang D. and D. Powers 2005. *Measuring semantic similarity in the taxonomy of WordNet*, Proceedings of the Australasian conference on Computer Science.

Forest-guided Supertagger Training

Yao-zhong Zhang [†] **Takuya Matsuzaki** [†] **Jun'ichi Tsujii**[†‡§]

† Department of Computer Science, University of Tokyo
‡ School of Computer Science, University of Manchester
§National Centre for Text Mining
{yaozhong.zhang, matuzaki, tsujii}@is.s.u-tokyo.ac.jp

Abstract

Supertagging is an important technique for deep syntactic analysis. A supertagger is usually trained independently of the parser using a sequence labeling method. This presents an inconsistent training objective between the supertagger and the parser. In this paper, we propose a forest-guided supertagger training method to alleviate this problem by incorporating global grammar constraints into the supertagging process using a CFG-filter. It also provides an approach to make the supertagger and the parser more tightly integrated. The experiment shows that using the forest-guided trained supertagger, the parser got an absolute 0.68% improvement from baseline in F-score for predicate-argument relation recognition accuracy and achieved a competitive result of 89.31% with a faster parsing speed, compared to a state-of-the-art HPSG parser.

1 Introduction

Deep syntactic analysis by lexicalized grammar parsing, which provides linguistic-rich information for many NLP tasks, has recently received more and more attention from the NLP community. To use a deep parser in real large-scale applications, speed is an important issue to take into consideration. Supertagging is one of the speed-up technique widely used for lexicalized grammar parsing. A supertagger is used to limit the number of plausible lexical entries fed to the parser, this can greatly reduce the search space for the parser. Supertagging was first proposed for Lexicalized Tree Adjoining Grammar (LTAG) (Bangalore and Joshi, 1999), and then successfully applied to Combinatory Categorial Grammar (CCG) (Clark, 2002) and Head-driven Phrase Structure Grammar (HPSG) (Ninomiya et al., 2006). In addition, supertags can also be used for other NLP tasks besides parsing, such as semantic role labeling (Chen and Rambow, 2003) and machine translation (Birch et al., 2007; Hassan et al., 2007) to utilize syntactic information in the supertags.

In lexicalized grammar parsing, supertagging is usually treated as a sequence labeling task independently trained from the parser. Previous research (Clark, 2002) showed that even a pointwise classifier not considering context edge features is effective when used as a supertagger. To make up for the insufficient accuracy as a single-tagger, more than one supertag prediction is reserved and the parser takes the burden of resolving the rest of the supertag ambiguities.

A non-trivial problem raised by the separate training of the supertagger is that the prediction score provided by the supertagger might not be suitable for direct use in the parsing process, since a separately trained supertagger that does not take into account grammar constraints has a training objective which is inconsistent with the parser. Although the scores provided by the supertagger can be ignored (e.g., in some CCG parsers), this may also discard some useful information for effective beam search and accurate disambiguation.

Based on this observation, we assume that considering global grammar constraints during the supertagger training process would make the supertagger and the parser more tightly integrated.

In this paper, we propose an on-line forest-guided training method for a supertagger to make the training objective of a supertagger more closely related to the parsing task. We implemented this method on a large-scale HPSG grammar. We used a CFG grammar to approximate the original HPSG grammar in the supertagging stage and applied best-first search to select grammar-satisfying supertag sequences for the parameter updating. The experiments showed that the HPSG parser is improved by considering structure constraints in the supertagging training process. For the standard test set (Penn Treebank Section 23), we accomplished an absolute 0.68% improvement from baseline in F-score for predicate-argument relation recognition and got a competitive result of 89.31% with a faster parsing speed, compared to a state-of-the-art HPSG parser.

The remainder of the paper is organized as follows: in section 2 we provide the necessary background regarding HPSG parsing. In section 3, we introduce the on-line forest-guided supertagger training method. Section 4 shows the experiment results and the related analysis. Section 5 compares the proposed approach with related work and section 6 presents our conclusions and future work.

2 Background

2.1 Statistical HPSG Parsing

HPSG (Pollard and Sag, 1994) is a lexicalist grammar framework. In HPSG, a large number of lexical entries are used to express word-specific characteristics, while only a small number of rule schemata are used to describe general construction rules. Typed feature structures named "signs" are used to represent both lexical entries and phrasal constituents. A classic efficient statistical HPSG parsing process is depicted in Figure 1. Given a word and part-of-speech sequence (w, p) as input, the first step (called "supertagging") in HPSG parsing is to assign possible lexical entries. In practice, for each word, more than one supertag is reserved for the parser. Then, the parser searches the given lexical entry space to construct a HPSG tree using the rule schemata to combine possible signs. Constituent-based methods and transition-based methods can be used for tree structure disambiguation. This parsing framework using supertagging is also used in other lexicalized grammars, such as LTAG and CCG.

2.2 HPSG Supertagging

Like other lexicalized grammar, the lexical entries defined in HPSG are referred to as "supertags". For example, the word "like" is assigned a lexical entry for transitive verbs in non-3rd person present form, which indicates that the head syntactic category of "like" is verb and it has an NP subject and an NP complement. With such fine-grained grammatical type distinctions, the number of supertags is very large. Compared to the 45 part-of-speech (POS) tags defined in the PennTreebank, the HPSG grammar we used contains 2,308 supertags. The large number and the complexity of the supertags makes supertagging harder than the POS tagging task.

Supertagging can be formulated as a sequence labeling task. Here, we follow the definition of Collins' perceptron (Collins, 2002). The training objective of supertagging is to learn the mapping from a POS-tagged word sentence $\boldsymbol{w} = (w_1/p_1, ..., w_n/p_n)$ to a sequence of supertags $\boldsymbol{s} = (s_1, ..., s_n)$. We use function $\boldsymbol{GEN(w)}$ to indicate all candidates of supertag sequences given input $\boldsymbol{w}$. Feature function $\boldsymbol{\Phi}$ maps a sample $(\boldsymbol{w}, \boldsymbol{s})$ to a point in the feature space R^d. $\boldsymbol{\theta}$ is the vector of feature weights. Given an input $\boldsymbol{w}$, the most plausible supertag sequence is found by the prediction function defined as follows:

$$F(\boldsymbol{w}) = \underset{\boldsymbol{s} \in \boldsymbol{GEN(w)}}{argmax} \; \theta \cdot \Phi(\boldsymbol{w}, \boldsymbol{s}) \qquad (1)$$

2.3 CFG-filtering

CFG-filtering (Kiefer and Krieger, 2000) is a technique to find a superset of (packed) HPSG parse trees that satisfy the constraints in a grammar. A CFG that approximates the original HPSG grammar is used for efficiently finding such trees without doing full-fledged HPSG parsing that is computationally demanding because the schema application involves unification operations among large feature structures (signs). The number of possible signs is infinite in general and hence

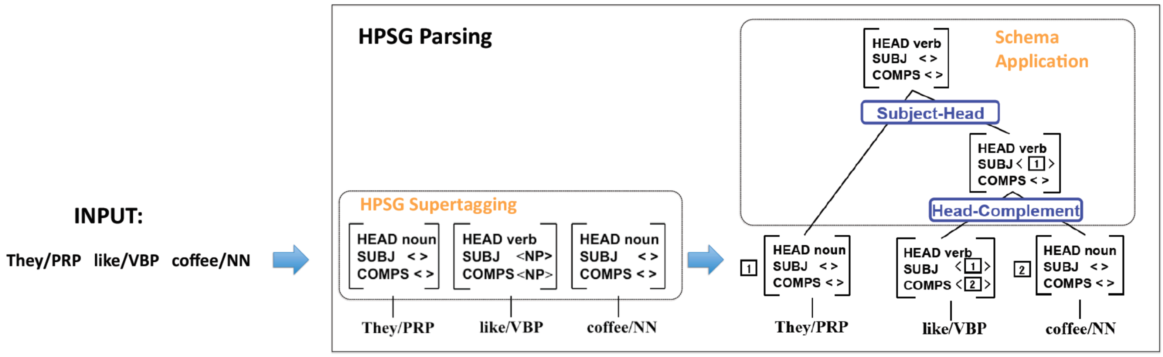

INPUT:
They/PRP like/VBP coffee/NN
HPSG Parsing
HPSG Supertagging
HEAD noun SUBJ < > COMPS < >
HEAD verb SUBJ <NP> COMPS <NP>
HEAD noun SUBJ < > COMPS < >
They/PRP
like/VBP
coffee/NN
Schema Application
HEAD verb SUBJ < > COMPS < >
Subject-Head
HEAD verb SUBJ < 1 > COMPS < >
Head-Complement
1
HEAD noun SUBJ < > COMPS < >
HEAD verb SUBJ < 1 > COMPS < 2 >
2
HEAD noun SUBJ < > COMPS < >
They/PRP
like/VBP
coffee/NN

Figure 1: HPSG parsing for the sentence “They like coffee.”

some features (e.g., the number agreement feature) are ignored in the approximating CFG so that the set of possible signs can be approximated by a finite set of non-terminal symbols in the CFG. By this construction, some illegal trees may be included in the set of trees licensed by the approximating CFG, but none of the well-formed trees (i.e., those satisfying all constraints in the grammar) are excluded by the approximation. We use the algorithm described by Kiefer and Krieger (2000) to obtain the approximating CFG for the original HPSG. The technical details regarding the algorithm can be found in Kiefer and Krieger (2000).

3 Forest-guided Training for Supertagging

3.1 Motivation

In lexicalized grammar parsing, a parser aims to find the most plausible syntactic structure for a given sentence based on the supertagging results. One efficient parsing approach is to use prediction scores provided by the supertagger. Usually, the supertagger is trained separately from the structure disambiguation in a later stage. This pipeline parsing strategy poses a potential problem in that the training objective of a supertagger can deviate from the final parser, if the global grammar constraints are not considered. For example, the supertag predictions for some words can contribute to high supertagging accuracy, but cause the parser to fail. Therefore, considering the global grammar constraints in the supertagging training stage can make the supertagger and the parser more tightly related, which will contribute towards the performance of the parser.

Algorithm 1: Forest-guided supertagger training

```
Input: Training Sample $(w_i, s_i)_{i=1,...,N}$,
       Number of iterations T
$\theta \leftarrow (0, ..., 0)$, $\theta_{sum} \leftarrow (0, ..., 0)$
for $iterNum \leftarrow$ 1 to T do
    for i $\leftarrow$ 1 to N do
        Generate supertag lattice using
        the point-wise classifier with current $\theta$
        Select $\hat{s}_i$ from the lattice
        which can construct a tree
        with largest sequence score
        if( No $\hat{s}_i$ satisfied grammar constraints)
        $\hat{s}_i \leftarrow \arg\max_{s \in \mathbf{GEN}(w_i)} \theta_i \cdot \Phi(w_i, s_i)$
        if $\hat{s}_i \neq s_i$ then
            $\theta_{i+1} \leftarrow \theta_i + \Phi(w_i, s_i) - \Phi(w_i, \hat{s}_i)$
            $\theta_{sum} \leftarrow \theta_{sum} + \theta_{i+1}$
Return: $\theta_{\text{sum}}/NT$
```

3.2 Training Algorithm

Based on the motivation above, we propose a forest-guided supertagger training method to make the supertagger more tightly integrated with the parser. This method is based on the averaged perceptron training algorithm. The training process is given in Algorithm 1.

The most important difference of the proposed algorithm compared to the traditional supertagger training method is that the current best-scored supertag sequence is searched only within the space of the supertag sequences that are allowed by the grammar. As for whether the grammar constraints are satisfied, we judge it by whether a possible syntactic tree can be constructed using the given supertag sequence. We do not require the constructed syntactic tree to be identical to the gold tree in the corpus. For this reason we call it "forest-guided".

In the forest-guided training of the supertagger, an approximating CFG is used to filter out the supertag sequences from which no well-formed tree can be built. It is implemented as a best-first CFG parser wherein the score of a constituent is the score of the supertag (sub-)sequence on the fringe of the constituent, which is calculated using the current value of the parameters. Note that the best-first parser can find the best-scored supertag sequence very efficiently given proper scoring for the candidate supertag set for each token; this is actually the case in the course of training except for the initial phase of the training, wherein the parameter values are not well-tuned. The efficiency is due to the sparseness of the approximating CFG (i.e., the production rule set includes only a tiny fraction of the possible parent-children combinations of symbols) and highest-scored supertags often have a well-formed tree on top of them.

As is clear from the above description, the use of CFG-filter in the forest-guided training of the supertagger is not essential but is only a subsidiary technique to make the training faster. The improvement by the forest-guided training should however depend on whether the CFG approximation is reasonably tight or not. Actually, we managed to obtain a manageable size out of a CFG grammar, which includes 80 thousand nonterminal symbols and 10 million rules, by eliminating only a small number of features (semantics, case and number agreement, and fine distinctions in nouns, adjectives and complementizers). We thus believe that the approximation is fairly tight.

This training algorithm can also be explained in a search-based learning framework (Hal Daumé III and Daniel Marcu, 2005). In this framework, the objective of learning is to optimize the $\boldsymbol{\theta}$ for the enqueue function to make the good hypotheses rank high in the search queue. The rank score $\boldsymbol{r}$ consists of two components: path score $\boldsymbol{g}$ and heuristic score $\boldsymbol{h}$. In the forest-guided training method, $\boldsymbol{r}$ can be rewritten as follows:

$$\begin{aligned} r &= g + h \\ &= \theta \cdot \Phi(x, \hat{y}) + 1_{[Tree(\hat{y})]} * Penalty \end{aligned} \quad (2)$$

The heuristic part $\boldsymbol{h}$ checks whether the supertag candidate sequence satisfies the grammar constraints: if no CFG tree can be constructed, -∞ penalty is imposed to the candidate sequence in the forest-guided training method.

4 Experiments

We mainly evaluated the proposed forest-guided supertagger training method on HPSG parsing. Supertagging accuracy[1] using different training methods was also investigated.

4.1 Corpus Description

The HPSG grammar used in the experiments is Enju version 2.3[2]. It is semi-automatically converted from the WSJ portion of PennTreebank (Miyao, 2006). The grammar consists of 2,308 supertags in total. Sections 02-21 were used to train different supertagging models and the HPSG parser. Section 22 and section 23 were used as the development set and the test set respectively. We evaluated the HPSG parser performance by labeled precision (LP) and labeled recall (LR) of predicate-argument relations of the parser's output as in previous works (Miyao, 2005). All experiments were conducted on an AMD Opteron 2.4GHz server.

Template Type	Template
Word	w_i,w_{i-1},w_{i+1}, w_{i-1}&w_i, w_i&w_{i+1}
POS	$p_i, p_{i-1}, p_{i-2}, p_{i+1}$, p_{i+2}, p_{i-1}&p_i, p_{i-2}&p_{i-1}, p_{i-1}&p_{i+1}, p_i&p_{i+1}, p_{i+1}&p_{i+2}
Word-POS	p_{i-1}&w_i, p_i&w_i, p_{i+1}&w_i

Table 1: Feature templates used for supertagging models.

[1] "UNK" supertags are ignored in evaluation as in previous works.

[2] http://www-tsujii.is.s.u-tokyo.ac.jp/enju/index.html

4.2 Baseline Models and Settings

We used a point-wise averaged perceptron (PW) to train a baseline supertagger. Point-wise classifiers have been reported to be very effective and with competitive results for the supertagging task (Clark, 2002; Zhang et al., 2009). The number of training iterations was set to 5. The features used in the supertaggers are described in Table 1. For comparison, these features are identical to the features used in the previous works (Matsuzaki et al., 2007; Ninomiya et al., 2007). To make the training efficient, we set the default chart size limit for the forest-guided supertagger training to be 20k by tuning it on the development set.

We combined the supertagger trained under forest-guidance with a supertagging-based HPSG parser (Matsuzaki et al., 2007) and evaluated the contribution of the improved supertagger training procedure for the final HPSG parsing by the accuracy of the predicate-argument relations output of the parser. The parser crucially depends on the supertagger's performance in that it outputs the first well-formed tree successfully constructed on the highest scored supertag sequence. The highest-scored supertag sequences are enumerated one by one in descending order in regards to their score. The enumeration is actually implemented as n-best parsing on the supertag candidates using an approximating CFG. The HPSG tree construction on a supertag sequence is done using a shift-reduce style parsing algorithm equipped with a classifier-based action selection mechanism.

The automatically assigned POS tags were given by a maximum entropy tagger with roughly 97% accuracy.

4.3 Supertagging Results

Although we mainly focused on improving the final HPSG parsing performance through the improved supertagger training, it is also very interesting to investigate the supertagger performance using different training methods. To evaluate the forest-guided training method for a supertagger, we also need to incorporate structure constraints in the test stage. To make fair comparisons, for the averaged perceptron trained supertagger we also add structure constraints in its testing. For simplicity, throughout this paper, we call the forest-guided trained supertagger "FT" in short, while the "PW" is used to represent the baseline point-wise averaged perceptron supertagger. "ME" is the re-implemented maximum entropy supertagger described in Matsuzaki et al. (2007).

	Model Name	**Acc%**
auto-POS	FT+CFG	**92.77**
	PW+CFG	92.47
	PW	91.14
	ME	91.45
gold-POS	FT+CFG	**93.98**
	PW+CFG	93.70
	PW	92.48
	ME	92.78

Table 2: Supertagging results in section 23. "FT" represents the forest-guided trained supertagger. "PW" is the baseline average perceptron trained supertagger. "ME" is the supertagger trained by using the maximum entropy method. "+CFG" indicates the use of the CFG-filter for the supertagger results. The accuracy of automatically assigned POS tags in this section is 97.39%.

For the PW supertagger, the performance was roughly 0.3% below the ME supertagger. Similar results were reported by Zhang et al. (2009), which used a Bayes point machine to reduce the gap between the averaged perceptron supertagger and the maximum entropy supertagger. Although we expected the ME supertagger using CFG-filter to give better results than the PW supertagger, implementing forest-guided supertagger training in a maximum entropy framework is different and more sophisticated than the current on-line training method. Considering that the performance of the PW supertagger and the ME supertagger were at a similar level, we chose the PW supertagger as our baseline.

We used a CFG-filter to incorporate global grammar constraints into both the training and the testing phase. Compared to the PW supertagger, the PW+CFG supertagger incorporated global grammar constraints only in the test phase, while for the FT+CFG supertagger, the global grammar constraints were incorporated both in

Iter NUM Training Method	1	2	3	4	5	Total Time
FT	6684s	4189s	3524s	3285s	3086s	≈ 5.8h
PW	99s	116s	117s	117s	117s	≈ 10 min
ME	/					≈ 3h

Table 3: Supertagger training time on section 02-21. "FT" and "PW" represent forest-guided training and point-wise averaged perceptron training separately. "ME" is the point-wise maximum entropy training reported in Matsuzaki et al. (2007).

the training and the testing stage. The supertagging accuracy for different models is shown in Table 2. Firstly, incorporating grammar constraints only in the testing phase (PW+CFG) gave an absolute 1.22% (gold POS) and 1.33% (auto POS) increase in F-score compared to the PW supertagger. Secondly, incorporating grammar constraints into both the training and the testing stage (FT+CFG) gave an additional 0.28% (gold POS) and 0.3% (auto POS) improvement over the PW+CFG supertagger with p-values 0.0018 (gold POS) and 0.0016 (auto POS).

This also indicates that the supertagger and the parser are closely related to each other. The original motivation for supertagging is using simple models to resolve lexical ambiguities, which can efficiently reduce the search space of the parser. A better supertagger can contribute to more efficient and more accurate lexicalized grammar parsing. Actually, a supertagger can act as a coarse parser for the whole parsing process as well, as long as the coarse parser is efficient. Since supertag disambiguation is highly constrained by the grammar, incorporating grammar constraints into supertagging (including training and testing) by using the CFG-filter can further improve the supertagging performance, as shown in Table 2.

As for the supertagger training time, incorporating grammar constraints inevitably increases the training time. As shown in Table 3, the total training time of forest-guided training (default settings, with chart size limited to 20k) was about 5.8 hours. For each iteration of the FT model, we find that the training time gradually decreases with each successive iteration. This hints that we can do better model initialization to further reduce the training time.

4.4 HPSG Parsing Results

We evaluated the HPSG parsers using different supertagger training methods. For the baseline HPSG parser, a CFG-filter is already incorporated to accelerate the parsing process. In the following experiments, we fed the parser all the possible supertag candidates with the prediction scores generated by the supertaggers. We controlled the upper bound of the chart size in the CFG-filter to make the parser more efficient.

Table 4 shows the results of the different parsing models. We first compared the baseline parsers using different supertaggers. The forest-guided supertagger improved the final FT parser's F-score by 0.68% (statistically significant) over the PW parser using the PW supertagger, which did not consider global grammar constraints during the supertagger training process. The parsing time of the FT parser was very close to that of the PW parser (108s vs. 106s), which was also efficient. The result empirically reflects that incorporating the global grammar constraints into the supertagger training process can refine supertag predicting scores, which become more consistent and compatible with the parser.

We also compared our results with a state-of-the-art HPSG parser using the same grammar. Enju (Miyao, 2005; Ninomiya et al., 2007) is a log-linear model based HPSG parser, which uses a maximum entropy model for the structure disambiguation. In contrast to our baseline parser, full HPSG grammar is directly used with CKY algorithm in the parsing stage. As for the parsing performance, our baseline PW parser using the PW supertagger was 0.23% below the Enju parser. However, by using the forest-guided trained supertagger, our improved FT parser per-

Parser	**UP**	**UR**	**LP**	**LR**	**F-score**	**Time** †
FT Parser	92.28	92.14	89.38	89.23	**89.31**	108s
PW Parser	91.88	91.63	88.75	88.51	88.63	106s
Enju 2.3	92.26	92.21	88.89	88.84	88.86	775s

Table 4: Parser performance on Section 23. “FT Parser” represents baseline parser which uses forest-guided trained supertagger. “PW Parser” represents the baseline parser which uses the point-wise averaged perceptron trained supertagger. (†) The time is the total time of both supertagging and parsing and it was calculated on all 2291 sentences of the Section 23.

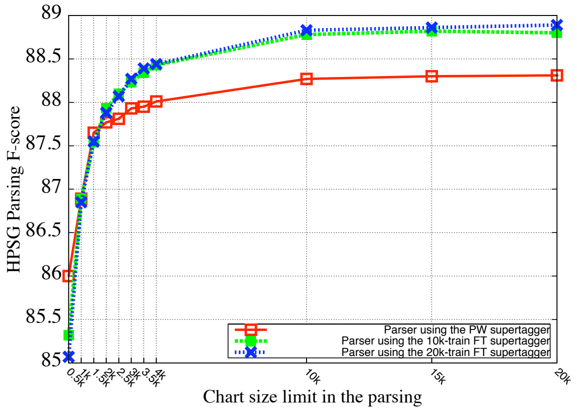


Figure 2: The F-score of the HPSG parsers on section 22 using different settings for the chart size limit in supertagger training and parsing.

formed 0.45% better than the Enju parser (default settings) in F-score. In addition, our shift-reduce style parser was faster than the Enju parser.

Beam size plays an important role for the forest-guided supertagger training method, since a larger beam size reduces the possibility of search errors. Precisely speaking, we control the beam size by limiting the number of edges in the chart in both the forest-guided supertagger training process and the final parsing. Figure 2 shows the results of setting different limits for the chart size during supertagger training and parsing on the development set. The X-axis represents the chart size limitation for the parsing. “10k-train” represents the chart size to be limited to 10k during FT supertagger training phase. A similar representation is used for “20k-train”. There is no tree structure search process for the baseline PW supertagger. We evaluated the F-score of the parsers using different supertaggers. As shown in Figure 2, when the chart size of the parser was more than 10k, the benefit of using forest-guided supertaggers were obvious (around an absolute 0.5% improvement in F-score, compared to the parser using the baseline PW supertagger). The performance of the parser using “10k-train” FT supertagger was already approaching to that of the parser using “20k-train” FT supertagger. When the chart size of the parser was less than 2000, the forest-guided supertaggers were not work. Similar to the results showed in previous research (Hal Daumé III and Daniel Marcu, 2005), it is better to use the same chart size limit in the forest-guided supertagger training and the final parsing.

5 Related Work

Since the supertagging technique is well known to drastically improve the parsing speed and accuracy, there is work concerned with tightly integrating a supertagger with a lexicalized grammar parser. Clark and Curran (2004) investigated a multi-tagger supertagging technique for CCG. Based on the multi-tagging technique, supertagger and parser are tightly coupled, in the sense that the parser requests more supertags if it fails. They (Clark and Curran, 2007) also used the perceptron algorithm to train a CCG parser. Different from their work, we focused on improving the performance of the deep parser by refining the training method for supertagging. Ninomiya et al. (2007) used the supertagging probabilities as a reference distribution for the log-linear model for HPSG, which aimed to consistently integrate supertagging into probabilistic HPSG parsing. Prins et al. (2001) trained a POS-tagger on an automatic parser-generated lexical entry corpus as a filter for Dutch HPSG parsing to improve the parsing speed and accuracy.

The existing work most similar to ours is Boullier (2003). He presented a non-statistical parsing-based supertagger for LTAG. Similar to his method, we used a CFG to approximate the original lexicalized grammar. The main difference between these two methods is that we consider the grammar constraints in the training phase of the supertagger, not only in the supertagging test phase and our main objective is to improve the performance of the final parser.

6 Conclusions and Future Work

In this paper, based on the observation that supertaggers are commonly trained separately from lexicalized parsers without global grammar constraints, we proposed a forest-guided supertagger training method to integrate supertagging more tightly with deep parsing. We applied this method to HPSG parsing and made further significant improvement for both supertagging (0.28%) and the HPSG parsing (0.68%) compared to the baseline. The improved parser also achieved a competitive result (89.31%) with a faster parsing speed, compared to a state-of-the-art HPSG parser.

For future work, we will try to weight the forest trees for the supertagger training and extend this method to other lexicalized grammars, such as LTAG and CCG.

Acknowledgments

We are grateful to the anonymous reviewers for their valuable comments. We also thank Goran Topic and Pontus Stenetorp for their help proofreading this paper. The first author was supported by The University of Tokyo Fellowship (UT-Fellowship). This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan).

References

Bangalore, Srinivas and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25:237–265.

Birch, Alexandra, Miles Osborne, and Philipp Koehn. 2007. CCG supertags in factored statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 9–16.

Boullier, P. 2003. Supertagging: A non-statistical parsing-based approach. In *In Proceedings IWPT-2003*, volume 3, pages 55–65.

Chen, John and Owen Rambow. 2003. Use of deep linguistic features for the recognition and labeling of semantic arguments. In *Proceedings of EMNLP-2003*, pages 41–48.

Clark, Stephen and James R. Curran. 2004. The importance of supertagging for wide-coverage ccg parsing. In *Proceedings of COLING-04*, pages 282–288.

Clark, S. and J.R. Curran. 2007. Perceptron training for a wide-coverage lexicalized-grammar parser. In *Proceedings of the Workshop on Deep Linguistic Processing*, pages 9–16.

Clark, Stephen. 2002. Supertagging for combinatory categorial grammar. In *Proceedings of the 6th International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+ 6)*, pages 19–24.

Collins, M. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP-2002*, pages 1–8.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *International Conference on Machine Learning (ICML)*, pages 169–176.

Hassan, Hany, Mary Hearne, and Andy Way. 2007. Supertagged phrase-based statistical machine translation. In *Proceedings of ACL-2007*, pages 288–295.

Kiefer, Bernd and Hans-Ulrich Krieger. 2000. A context-free approximation of head-driven phrase structure grammar. In *Proceedings of IWPT-2000*, pages 135–146.

Matsuzaki, Takuya, Yusuke Miyao, and Jun'ichi Tsujii. 2007. Efficient HPSG Parsing with Supertagging and CFG-filtering. In *Proceedings of IJCAI-07*, pages 1671–1676.

Miyao, Yusuke. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 83–90.

Miyao, Yusuke. 2006. *From Linguistic Theory to Syntactic Analysis: Corpus-Oriented Grammar Development and Feature Forest Model*. Ph.D. Dissertation, The University of Tokyo.

Ninomiya, Takashi, Yoshimasa Tsuruoka, Takuya Matsuzaki, and Yusuke Miyao. 2006. Extremely lexicalized models for accurate and fast HPSG parsing. In *Proceedings of EMNLP-2006*, pages 155–163.

Ninomiya, T., T. Matsuzaki, Y. Miyao, and J. Tsujii. 2007. A log-linear model with an n-gram reference distribution for accurate HPSG parsing. *In Proceedings of IWPT-2007*, pages 60–68.

Pollard, Carl and Ivan A. Sag. 1994. *Head-driven Phrase Structure Grammar*. University of Chicago / CSLI.

Prins, R. and G. Van Noord. 2001. Unsupervised Pos-Tagging Improves Parsing Accuracy And Parsing Efficiency. In *Proceedings of IWPT-2001*, pages 154–165.

Zhang, Yao-zhong, Takuya Matsuzaki, and Jun'ichi Tsujii. 2009. HPSG Supertagging: A Sequence Labeling View. In *Proceedings of IWPT-2009*, pages 210–213.

Entity Linking Leveraging

Automatically Generated Annotation

Wei Zhang[†] **Jian Su**[‡] **Chew Lim Tan**[†] **Wen Ting Wang**[‡]

[†]School of Computing
National University of Singapore
`{z-wei, tancl}`
`@comp.nus.edu.sg`

[‡] Institute for Infocomm Research
`{sujian, wwang}`
`@i2r.a-star.edu.sg`

Abstract

Entity linking refers entity mentions in a document to their representations in a knowledge base (KB). In this paper, we propose to use additional information sources from Wikipedia to find more name variations for entity linking task. In addition, as manually creating a training corpus for entity linking is labor-intensive and costly, we present a novel method to automatically generate a large scale corpus annotation for ambiguous mentions leveraging on their unambiguous synonyms in the document collection. Then, a binary classifier is trained to filter out KB entities that are not similar to current mentions. This classifier not only can effectively reduce the ambiguities to the existing entities in KB, but also be very useful to highlight the new entities to KB for the further population. Furthermore, we also leverage on the Wikipedia documents to provide additional information which is not available in our generated corpus through a domain adaption approach which provides further performance improvements. The experiment results show that our proposed method outperforms the state-of-the-art approaches.

1 Introduction

The named entity (NE) ambiguation has raised serious problems in many areas, including web people search, knowledge base population (KBP), and information extraction, because an entity (such as *Abbott Laboratories*, a diversified pharmaceuticals health care company) can be referred to by multiple mentions (e.g. "*ABT*" and "*Abbott*"), and a mention (e.g. "*Abbott*") can be shared by different entities (e.g. *Abbott Texas*: a city in United States; *Bud Abbott*, an American actor; and *Abbott Laboratories*, a diversified pharmaceutical health care company).

Both Web People Search (WePS) task (Artiles et al. 2007) and Global Entity Detection & Recognition task (GEDR) in Automatic Content Extraction 2008 (ACE08) disambiguate entity mentions by clustering documents with these mentions. Each cluster then represents a unique entity. Recently entity linking has been proposed in this field. However, it is quite different from the previous tasks.

Given a knowledge base, a document collection, entity linking task as defined by KBP-09[1] (McNamee and Dang, 2009) is to determine for each name string and the document it appears, which knowledge base entity is being referred to, or if the entity is a new entity which is not present in the reference KB.

Compared with GEDR and WePS, entity linking has a given entity list (i.e. the reference KB) to which we disambiguate the entity mentions. Moreover, in document collection, there are new entities which are not present in KB and can be used for further population. In fact, new entities with or without the names in KB cover more than half of testing instances.

[1] http://apl.jhu.edu/~paulmac/kbp.html

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1290–1298,
Beijing, August 2010

Entity linking has been explored by several researchers. Without any training data available, most of the previous work ranks the similarity between ambiguous mention and candidate entities through Vector Space Model (VSM). Since they always choose the entity with the highest rank as the answer, the ranking approaches hardly detect a situation where there may be a new entity that is not present in KB. It is also difficult to combine bag of words (BOW) with other features. For example, to capture the "category" information, the method of Cucerzan (2007) involves a complicated optimization issue and the approach has to be simplified for feasible computation, which compromises the accuracy. Besides unsupervised methods, some supervised approaches (Agirre et al. 2009, Li et al. 2009 and McNamee et al. 2009) also have been proposed recently for entity linking. However, the supervised approaches for this problem require large amount of training instances. But manually creating a corpus is labor-intensive and costly.

In this paper, we explore how to solve the entity linking problem. We present a novel method that can automatically generate a large scale corpus for ambiguous mentions leveraging on their unambiguous synonyms in the document collection. A binary classifier based on Support Vector Machine (SVM) is trained to filter out some candidate entities that are not similar to ambiguous mentions. This classifier can effectively reduce the ambiguities to the existing entities in KB, and it is very useful to highlight the new entities to KB for the further population. We also leverage on the Wikipedia documents to provide additional information which is not available in our generated corpus through a domain adaption approach which provides further performance improvements. Besides, more information sources for finding more variations also contribute to the overall 22.9% accuracy improvements on KBP-09 test data over baseline.

The remainder of the paper is organized as follows. Section 2 reviews related work for entity linking. In Section 3 we detail our algorithm including name variation and entity disambiguation. Section 4 describes the experimental setup and results. Finally, Section 5 concludes the paper.

2 Related Work

The crucial component of entity linking is the disambiguation process. Raphael et al. (2007) report a disambiguation algorithm for geography. The algorithm ranks the candidates based on the manually assigned popularity scores in KB. The class with higher popularity will be assigned higher score. It causes that the rank of entities would never change, such as Lancaster (California) would always have a higher rank than Lancaster (UK) for any mentions. However, as the popularity scores for the classes change over time, it is difficult to accurately assign dynamic popularity scores. Cucerzan (2007) proposes a disambiguation approach based on vector space model for linking ambiguous mention in a document with one entity in Wikipedia. The approach ranks the candidates and chooses the entity with maximum agreement between the contextual information extracted from Wikipedia and the context of a document, as well as the agreement among the category tags associated with the candidate entities. Nguyen and Cao (2008) refer the mentions in a document to KIM (Popov et al. 2004) KB. KIM KB is populated with over 40,000 named entities. They represent a mention and candidates as vectors of their contextual noun phrase and co-occurring NEs, and then the similarity is determined by the common terms of the vectors and their associated weights. For linking mentions in news articles with a Wikipedia-derived KB (KBP-09 data set), Varma et al. (2009) rank the entity candidates using a search engine. Han and Zhao (2009) rank the candidates based on BOW and Wikipedia semantic knowledge similarity.

All the related work above rank the candidates based on the similarity between ambiguous mention and candidate entities. However, the ranking approach hardly detects the new entity which is not present in KB.

Some supervised approaches also have been proposed. Li et al. (2009) and McNamee et al. (2009) train their models on a small manually created data set containing only 1,615 examples. But entity linking requires large training data. Agirre et al. (2009) use Wikipedia to construct their training data by utilizing Inter-Wikipedia links and the surrounding snippets of text. However, their training data is created from a different domain which does not work well in the targeted news article domain.

3 Approach

In this section we describe our two-stage approach for entity linking: name variation and entity disambiguation. The first stage finds variations for every entity in the KB and generates an entity candidate set for a given query. The second stage is entity disambiguation, which links an entity mention with the real world entity it refers to.

3.1 Name Variation

The aim for Name Variation is to build a Knowledge Repository of entities that contains vast amount of world knowledge of entities like name variations, acronyms, confusable names, spelling variations, nick names etc. We use Wikipedia to build our knowledge repository since Wikipedia is the largest encyclopedia in the world and surpasses other knowledge bases in its coverage of concepts and up-to-date content. We obtain useful information from Wikipedia by the tool named Java Wikipedia Library[2] (Zesch et al. 2008), which allows to access all information contained in Wikipedia.

Cucerzan (2007) extracts the name variations of an entity by leveraging four knowledge sources in Wikipedia: "entity pages", "disambiguation pages" "redirect pages" and "anchor text".

Entity page in Wikipedia is uniquely identified by its title – a sequence of words, with the first word always capitalized. The title of Entity Page represents an unambiguous name variation for the entity. A redirect page in Wikipedia is an aid to navigation. When a page in Wikipedia is redirected, it means that those set of pages are referring to the same entity. They often indicate synonym terms, but also can be abbreviations, more scientific or more common terms, frequent misspellings or alternative spellings etc. Disambiguation pages are created only for ambiguous mentions which denote two or more entities in Wikipedia, typically followed by the word "*disambiguation*" and containing a list of references to pages for entities that share the same name. This is more useful in extracting the abbreviations of entities, other possible names for an entity etc. Besides, both outlinks and inlinks in Wikipedia are associated with anchor texts that represent name variations for the entities.

Using these four sources above, we extracted name variations for every entity in KB to form the Knowledge Repository as Cucerzan's (2007) method. For example, the variation set for entity *E0272065* in KB is {*Abbott Laboratories, Abbott Nutrition, Abbott* …}. Finally, we can generate the entity candidate set for a given query using the Knowledge Repository. For example, for the query containing "*Abbott*", the entity candidate set retrieved is {*E0272065, E0064214* …}.

From our observation, for some queries the retrieved candidate set is empty. If the entity for the query is a new entity, not present in KB, empty candidate set is correct. Otherwise, we fail to identify the mention in the query as a variation, commonly because the mention is a misspelling or infrequently used name. So we propose to use two more sources "Did You Mean" and "Wikipedia Search Engine" when Cucerzan (2007) algorithm returns empty candidate set. Our experiment results show that both proposed knowledge sources are effective for entity linking. This contributes to a performance improvement on the final entity linking accuracy.

Did You Mean: The "*did you* mean" feature of Wikipedia can provide one suggestion for misspellings of entities. This feature can help to correct the misspellings. For example, "Abbot Nutrition" can be corrected to "Abbott Nutrition".

Wikipedia Search Engine: This key word based search engine can return a list of relevant entity pages of Wikipedia. This feature is more useful in extracting infrequently used name.

Algorithm 1 below presents the approach to generate the entity candidate set over the created Knowledge Repository. $\boldsymbol{Ref_E(s)}$ is the entity set indexed by mention $\boldsymbol{s}$ retrieved from Knowledge Repository. In Step 8, we use the longest common subsequence algorithm to measure the similarity between strings $\boldsymbol{s}$ and the title of the entity page with highest rank. More details about longest common subsequence algorithm can be found in Cormen et al. (2001).

[2] http://www.ukp.tu-darmstadt.de/software/JWPL

Algorithm 1 Candidate Set Generation

```
Input: mention $s$;
if $Ref_E(s)$ is empty
    $s'$ $\leftarrow$ Wikipedia“did you mean”Suggestion
    If s’ is not NULL
        s $\leftarrow$ $s'$
    else
        EntityPageList $\leftarrow$ WikipediaSearchEngine(s)
        EntityPage$\leftarrow$FirstPage of EntityPageList
        Sim=Similarity(s,EntityPage.title)
        if Sim > Threshold
            $s \leftarrow EntityPage.title$
        end if
    end if
end if
Output: $Ref_E(s)$;
```

3.2 Entity Disambiguation

The disambiguation component is to link the mention in query with the entity it refers to in candidate set. If the entity to which the mention refers is a new entity which is not present in KB, *nil* will be returned. In this Section, we will describe the method for automatic data creation, domain adaptation from Wikipedia data, and our supervised learning approach as well.

3.2.1 Automatic Data Creation

The basic idea is to take a document with an unambiguous reference to an entity E1 and replacing it with a phrase which may refer to E1, E2 or others.

Observation: Some full names for the entities in the world are unambiguous. This phenomenon also appears in the given document collection of entity linking. The mention “*Abbott Laboratories*” appearing at multiple locations in the document collection refers to the same entity “*a pharmaceuticals health care company*” in KB.

From this observation, our method takes into account the mentions in the Knowledge Repository associated with only one entity and we treat these mentions as unambiguous name. Let us take *Abbott Laboratories*-{*E0272065*} in the Knowledge Repository as an example. We first use an index and search tool to find the documents with unambiguous mentions. Such as, the mention “*Abbott Laboratories” occurs in* document *LDC2009T13* and *LDC2007T07* in the document collection. The chosen text indexing and searching tool is the well-known Apache Lucene information retrieval open-source library[3].

Next, to validate the consistency of NE type between entities in KB and in document, we run the retrieved documents through a Named Entity Recognizer, to tag the named entities in the documents. Then we link the document to the entity in KB if the document contains a named entity whose name exactly matches with the unambiguous mention and type (i.e. Person, Organization and Geo-Political Entity) exactly matches with the type of entity in KB. In this example, after Named Entity Recognition, “Abbott Laboratories” in document *LDC2009T13* is tagged as an Organization which is consistent with the entity type of *E0272065* in KB. We link the *“Abbott Laboratories”* occurring in *LDC2009T13* with entity *E0272065.*

Finally, we replace the mention in the selected documents with the ambiguous synonyms. For example, we replace the mention *“Abbott Laboratories”* in document *LDC2009T13* with *“Abbott”* where *Abbott-{E0064214, E0272065…}* is an entry in Knowledge Repository. *“Abbott”* is ambiguous, because it is referring not only to *E0272065*, but also to *E0064214* in Knowledge Repository. Then, we can get two instances for the created data set as Figure 1, where one is positive and the other is negative.

(Abbott, LDC2009T13) E0272065 +

(Abbott, LDC2009T13) E0064214 -

…

+ refer to - not refer to

Figure 1: An instance of the data set

However, from our studies, we realize some limitations on our training data. For example, as shown in Figure 1, the negative instance for *E0272065* and the positive instance for

[3] http://lucene.apache.org

E0064214 are not in our created data set. However, those instances exist in the current document collection. We do not retrieve them since there is no unambiguous mention for *E0064214* in the document collection.

To reduce the effect of this problem, we propose to use the Wikipedia data as well, since Wikipedia data has training examples for all the entities in KB. Articles in Wikipedia often contain mentions of entities that already have a corresponding article, and at least the first occurrence of the mentions of an entity in a Wikipedia article must be linked to its corresponding Wikipedia article, if such an article exists. Therefore, if the mention is ambiguous, the hyperlink is disambiguating it. Next, we will describe how to incorporate Wikipedia data.

Incorporating Wikipedia Data. The document collection for entity linking is commonly from other domains, but not Wikipedia. To benefit from Wikipedia data, we introduce a domain adaption approach (Daumé III, 2007) which is suitable for this work since we have enough "target" domain data. The approach is to augment the feature vectors of the instances. Denote by X the input space, and by Y the output space, in this case, X is the space of the real vectors $\varphi(\mathrm{x})$ for the instances in data set and $Y=\{+1,-1\}$ is the label. D^s is the Wikipedia domain dataset and D^t is our automatically created data set. Suppose for simplicity that $X=R^F$ for some $F > 0$ (R^F is the space of F-dimensions). The augmented input space will be defined by $\breve{X} = R^{3F}$. Then, define mappings $\varphi^s, \varphi^t : X \rightarrow \breve{X}$ for mapping the Wikipedia and our created data set respectively. These are defined as follows:

$$\varphi^{\mathrm{s}}(\mathrm{x}) =< \varphi(\mathrm{x}), \varphi(\mathrm{x}), 0 >$$

$$\varphi^{\mathrm{t}}(\mathrm{x}) =< \varphi(\mathrm{x}), 0, \varphi(\mathrm{x}) >$$

Where $\mathbf{0}=<0,0,\ldots,0> \in R^F$ is the zero vector. We use the simple linear kernel in our experiments. However, the following kernelized version can help us to gain some insight into the method. K denotes the dot product of two vectors. $K(x,x')=< \varphi\ (x),\ \varphi\ (x')>$. When the domain is the same, we get: $\breve{K}(x,x') =< \varphi\ (x), \varphi\ (x') > +< \varphi\ (x), \varphi\ (x') = 2K(x,x')$. When they are from different domains, we get: $\breve{K}(x,x') =< \varphi\ (x), \varphi\ (x') >= K(x,x')$. Putting this together, we have:

$$\breve{K} = \begin{cases} 2K(x,x') & \text{same domain} \\ K(x,x') & \text{diff. domain} \end{cases}$$

This is an intuitively pleasing result. Loosely speaking, this means that data points from our created data set have twice as much influence as Wikipedia points when making predictions about test data from document collection.

3.2.2 The Disambiguation Framework

To disambiguate a mention in document collection, the ranking method is to rank the entities in candidate set based on the similarity score. In our work, we transform the ranking problem into a classification problem: deciding whether a mention refers to an entity on an SVM classifier. If there are 2 or more than 2 candidate entities that are assigned *positive* label by the binary classifier, we will use the baseline system (explained in Section 4.2) to rank the candidates and the entity with the highest rank will be chosen.

In the learning framework, the training or testing instance is formed by (*query*, *entity*) pair. For Wikipedia data, (*query*, *entity*) is *positive* if there is a hyperlink from the article containing the mention in query to the entity, otherwise (*query*, *entity*) is *negative*. Our automatically created data has been assigned labels in Section 3.2.1. Based on the training instances, a binary classifier is generated by using particular learning algorithm. During disambiguation, (*query*, *entity*) is presented to the classifier which then returns a class label.

Each (*query*, *entity*) pair is represented by the feature vector using different features and similarity metrics. We chose the following three classes of features as they represent a wide range of information - lexical features, word-category pair, NE type - that have been proved to be effective in previous works and tasks. We now discuss the three categories of features used in our framework in details.

Lexical features. For Bag of Words feature in Web People Search, Artiles et al. (2009) illustrated that noun phrase and n-grams longer than 2 were not effective in comparison with token-based features and using bi-grams gives the best results only reaching recall 0.7. Thus, we use token-based features. The similarity metric we choose is cosine (using standard tf.idf weighting). Furthermore, we also take into account the co-occurring NEs and represent it in the form of token-based features. Then, the single cosine similarity feature is based on Co-occurring NEs and Bag of Words.

Word Category Pair. Bunescu (2007) demonstrated that word-category pairs extracted from the document and Wikipedia article are a good signal for disambiguation. Thus we also consider word-category pairs as a feature class, i.e., all *(w,c)* where *w* is a word from Bag of Words of document and *c* is a category to which candidate entity belongs.

NE Type. This feature is a single binary feature to guarantee that the type of entity in document (i.e. Person, Geo-Political Entity and Organization) is consistent with the type of entity in KB.

4 Experiments and Discussions

4.1 Experimental Setup

In our study**,** we use KBP-09 knowledge base and document collection for entity linking. In the current setting of KBP-09 Data, the KB has been generated automatically from Wikipedia. The KB contains 818,741 different entities. The document collection is mainly composed of newswire text from different press agencies. The collection contains 1.3 million documents that span from 1994 to the end of 2008. The test data has 3904 queries across three named entity types: Person, Geo-Political Entity and Organization. Each query contains a document with an ambiguous mention.

Wikipedia data can be obtained easily from the website[4] for free research use. It is available in the form of database dumps that are released periodically. In order to leverage various information mentioned in Section 3.1 to derive name variations, make use of the links in Wikipedia to generate our training corpus and get word category information for the disambiguation, we further get Wikipedia data directly from the website. The version we used in our experiments was released on Sep. 02, 2009. The automatically created corpus (around 10K) was used as the training data, and 30K training instances associated with the entities in our corpus was derived from Wikipedia.

For pre-processing, we perform sentence boundary detection and Chunking derived from Stanford parser (Klein and Manning, 2003), Named Entity Recognition using a SVM based system trained and tested on ACE 2005 with 92.5(P) 84.3(R) 88.2(F), and coreference resolution using a SVM based coreference resolver trained and tested on ACE 2005 with 79.5%(P), 66.7%(R) and 72.5%(F).

We select SVM as the classifier used in this paper since SVM can represent the stat-of-the-art machine learning algorithm. In our implementation, we use the binary SVM^{Light} developed by Joachims (1999). The classifier is trained with default learning parameters.

We adopt the measure used in KBP-09 to evaluate the performance of entity linking. This measure is micro-averaged accuracy: the number of correct link divided by the total number of queries.

4.2 Baseline Systems

We build the baseline using the ranking approach which ranks the candidates based on similarity between mention and candidate entities. The entity with the highest rank is chosen. Bag of words and co-occurring NEs are represented in the form of token-based feature vectors. Then tf.idf is employed to calculate similarity between feature vectors.

To make the baseline system with token-based features state-of-the-art, we conduct a series of experiments. Table 1 lists the performances of our token-based ranking systems. In our experiment, local tokens are text segments generated by a text window centered on the mention. We set the window size to 55, which is the value that was observed to give optimum performance for the disambiguation problem (Gooi and Allan, 2004). Full tokens and NE are all the tokens and named entities co-occurring in the text respectively. We notice that tokens of the full text as well as the co-occurring named entity produce the best baseline performance, which we use for the further experiment.

[4] http://download.wikipedia.org

	Micro-averaged Accuracy
local tokens	60.0
local tokens + NE	60.6
full tokens + NE	61.9

Table 1: Results of the ranking methods

4.3 Experiment and Result

As discussed in Section 3.1, we exploit two more knowledge sources in Wikipedia: “did you mean” (DYM) and “Wikipedia search engine” (SE) for name variation step. We conduct some experiments to compare our name variation method using Algorithm 1 in Section 3.1 with the name variation method of Cucerzan (2007). Table 2 shows the comparison results of different name variation methods for entity linking. The experiments results show that, in entity linking task, our name variation method outperforms the method of Cucerzan (2007) for both entity disambiguation methods.

Name Variation Approaches	Ranking Method	Our Disambiguation Method
Cucerzan (2007)	60.9	82.2
+DYM+SE	61.9	**83.8**

Table 2: Entity Linking Result for two name variation approaches. Column 1 used the baseline method for entity disambiguation step. Column 2 used our proposed entity disambiguation method.

Table 3 compares the performance of different methods for entity linking on the KBP-09 test data. Row 1 is the result for baseline system. Row 2 and Row 3 show the results training on Wikipedia data and our automatically data respectively. Row 4 is the result training on both Wikipedia and our created data using the domain adaptation method mentioned in Section 3.2.1. It shows that our method trained on the automatically generated data alone significantly outperforms baseline. Compared Row 3 with Row 2, our created data set serves better at training the classifier than Wikipedia data. This is due to the reason that Wikipedia is a different domain from newswire domain. By comparing Row 4 with Row 3, we find that by using the domain adaptation method in Section 3.2.1, our method for entity linking can be further improved by 1.5%. Likely, this is because of the limitation of the auto-generated corpus as discussed in Section 3.2.1. In another hand, Wikipedia can complement the missing information with the auto-generated corpus. So combining Wikipedia data with our generated data can achieve better result. Compared with baseline system using Cucerzan (2007) name variation method in Table 2, in total our proposed method achieves a significant 22.9% improvement.

	Micro-averaged Accuracy
Baseline	61.9
Wiki	79.9
Created Data	82.3
Wiki → Created Data	**83.8**

Table 3: Micro-averaged Accuracy for Entity Linking

To test the effectiveness of our method to deal with new entities not present in KB and existing entities in KB respectively, we conduct some experiments to compare with Baseline. Table 4 shows the performances of entity linking systems for existing entities (non-NIL) in KB and new entity (NIL) which is not present in KB. We can see that the binary classifier not only effectively reduces the ambiguities to the existing entities in KB, but also is very useful to highlight the new entities to KB for the further population. Note that, in baseline system, all the new entities are found by the empty candidate set of name variation process, while the disambiguation component has no contribution. However, our approach finds the new entities not only by the empty candidate set, but also leveraging on disambiguation component which also contributes to the performance improvement.

	non-NIL	NIL
Baseline	72.6	52.4
Wiki → Created Data	79.2	87.8

Table 4: Entity Linking on Existing and New Entities

Finally, we also compare our method with the top 5 systems in KBP-09. Among them, *Siel_093* (Varma et al. 2009) and *NLPR_KBP1* (Han and Zhao 2009) use similarity ranking approach; *Stanford_UBC2* (Agirre et al. 2009), *QUANTA1* (Li et al. 2009) and *hltcoe1* (McNamee et al. 2009) use supervised approach. From the results shown in Figure 2, we observe that our method outperforms all the top 5 systems and the baseline system of KBP-09. Specifically, our method achieves better result than both similarity ranking approaches. This is due to the limitations of the ranking approach which have been discussed in Section 2. We also observe that our method gets a 5% improvement over *Stanford_UBC2*. This is because they collect their training data from Wikipedia which is a different domain from document collection of entity linking, news articles in this case; while our automatic data generation method can create a data set from the same domain as the document collection. Our system also outperforms *QUANTA1* and *hltcoe1* because they train their model on a small manually created data set (1,615 examples), while our method can automatically generate a much larger data set.

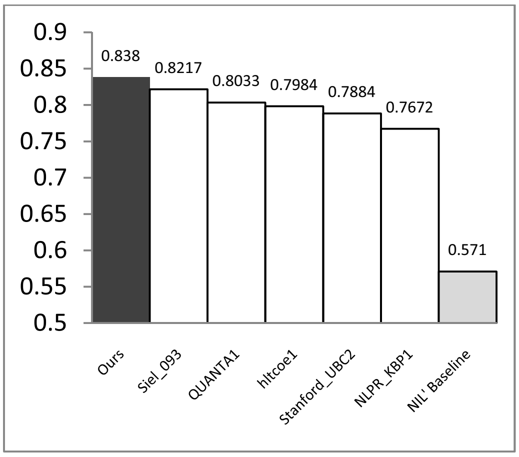


Figure 2: A comparison with KBP09 systems

5 Conclusion

The purpose of this paper is to explore how to leverage the automatically generated large scale annotation for entity linking. Traditionally, without any training data available, the solution is to rank the candidates based on similarity. However, it is difficult for the ranking approach to detect a new entity that is not present in KB, and it is also difficult to combine different features. In this paper, we create a large corpus for entity linking by an automatic method. A binary classifier is then trained to filter out KB entities that are not similar to current mentions. We further leverage on the Wikipedia documents to provide other information which is not available in our generated corpus through a domain adaption approach. Furthermore, new information sources for finding more variations also contribute to the overall 22.9% accuracy improvements on KBP-09 test data over baseline.

References

E. Agirre et al. Stanford-UBC at TAC-KBP. In *Proceedings of Test Analysis Conference 2009 (TAC 09).*

J. Artiles, J. Gonzalo, and S. Sekine. 2007. The semeval-2007 web evaluation: Establishing a benchmark for the web people search task. In *Proceeding of the Fourth International Work-shop on Semantic Evaluations (SemEval-2007).*

J. Artiles, E. Amigo and J. Gonzalo. 2009. The role of named entities in Web People Search. In *proceeding of the 47th Annual Meeting of the Association for Computational Linguistics.*

R. Bunescu. 2007. Learning for information extraction from named entity recognition and disambiguation to relation extraction. Ph.D thesis, University of Texas at Austin, 2007.

T. H. Cormen, et al. 2001. Introduction To Algorithms (Second Edition). *The MIT Press*, Page 350-355.

S. Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. *Empirical Methods in Natural Language Processing*, June 28-30, 2007.

H. Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics .*

C. H. Gooi and J. Allan. 2004. Cross-document coreference on a large scale corpus. In *proceedings of Human Language Technology Conference North American Association for Computational Linguistics Annual Meeting,* Boston, MA.

X. Han and J. Zhao. NLPR_KBP in TAC 2009 KBP Track: A Two-Stage Method to Entity Linking. In *Proceedings of Test Analysis Conference 2009 (TAC 09).*

T. Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

D. Klein and C. D. Manning. 2003. Fast Exact Inference with a Factored Model for Natural Language Parsing*. In Advances in Neural Information Processing Systems 15 (NIPS 2002), Cambridge, MA: MIT Press, pp. 3-10.*

F. LI et al. THU QUANTA at TAC 2009 KBP and RTE Track. In *Proceedings of Test Analysis Conference 2009 (TAC 09).*

P. McNamee and H. T. Dang. 2009. Overview of the TAC 2009 Knowledge Base Population Track. In *Proceedings of Test Analysis Conference 2009 (TAC 09).*

P. McNamee et al. HLTCOE Approaches to Knowledge Base Population at TAC 2009. In *Proceedings of Test Analysis Conference 2009 (TAC 09).*

H. T. Nguyen and T. H. Cao. 2008. Named Entity Disambiguation on an Ontology Enriched by Wikipedia. *2008 IEEE International Conference on Research, Innovation and Vision for the Future in Computing & Communication Technologies.*

B. Popov et al. 2004. KIM - a Semantic Platform for Information Extraction and Retrieval. In *Journal of Natural Language Engineering*, Vol. 10, Issue 3-4, Sep 2004, pp. 375-392, Cambridge University Press.

V. Raphael, K. Joachim and M. Wolfgang, 2007. Towards ontology-based disambiguation of geographical identifiers. In *Proceeding of the 16th WWW workshop on I3: Identity, Identifiers, Identifications, 2007.*

V. Varma et al. 2009. IIIT Hyderabad at TAC 2009. In *Proceedings of Test Analysis Conference 2009 (TAC 09).*

T. Zesch, C. Muller and I. Gurevych. 2008. Extractiong Lexical Semantic Knowledge from Wikipedia and Wiktionary. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, 2008.

A Rhetorical Syntax-Driven Model for Speech Summarization

Jian Zhang and Pascale Fung
Human Language Technology Center
Department of Electronic & Computer Engineering
Hong Kong University of Science & Technology (HKUST)
{zjustin, pascale}@ece.ust.hk

Abstract

We show a novel approach of parsing and reordering rhetorical syntax tree for extractive summarization of presentation speech. Our previous work showed (Fung et al., 2008) that rhetorical structures are embedded in this type of speech and that exploring this structure helps improve summarization quality. We further demonstrate that speakers do not follow the strict order of bullet points in the presentation slides, and that a re-ordering of these points occurs. We therefore propose a method of parsing presentation transcriptions into a rhetorical syntax tree and then re-order the leaf nodes to transform the speech transcriptions into an extractive summary, akin to a process of presentation slide generation. Chunking, parsing, and reordering are carried out by 28-class Hidden Markov Support Vector Machine(HMSVM) classifier trained from reference presentations and presentation slides. Using ROUGE-L F-measure we showed that our rhetorical syntax-driven model gives a 35.8% relative improvement over a binary summarizer with no rhetorical information, a 14.3% improvement over Rhetorical State Hidden Markov Model(RSHMM) (Fung et al., 2008), and a 4.3% improvement over our proposed model with no reordering.

1 Introduction

In this paper, we propose to improve extractive summarization of presentation speech using parsing and reordering of the salient points in the speech. Presentation speech includes classroom lectures, conference talks, business seminars, as well as political debates and parliamentary speech where the speaker gives a presentation according to some prepared slides containing bullet points. Some of the speech are transcribed into text, others might even be accompanied by short abstracts. Nevertheless, for learning and collaboration purposes, transcribed text is too long to read whereas short abstracts do not contain enough information (Teufel and Moens, 2002). Especially, in our conference presentation corpus, on average only about 40% bullet points of each transcription appears in the corresponding conference paper abstract. The accompanying presentation slides, on the other hand, are much better in summarizing the gists.

In recent years, more research has been conducted on exploring the hierarchical structure for better summarization performance (Fung et al., 2003; Murray et al., 2006; Sauper and Regina, 2009; Tatar et al., 2008). Unlike text documents, the structure of a spoken document is not immediately apparent in terms of its layout. However, researchers have shown that structural characteristics of a speech are clearly rendered by not just its linguistic features but also its acoustic features (Fung et al., 2008; Hirschberg and Nakatani, 1996; Nakatani et al., 1995). The hierarchical layout structure of Power Point slides enhances

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1299–1307,
Beijing, August 2010

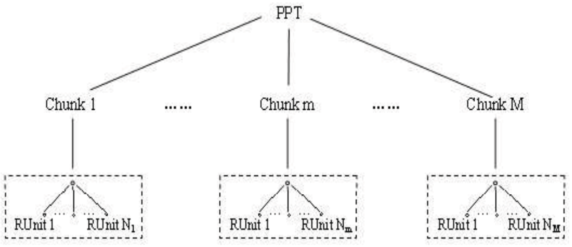


Figure 1: Rhetorical syntax tree

the understanding by the audience. In fact, they even provide a kind of extractive summarization that is superior in terms of informativeness than short abstracts. Unfortunately, presentation slides are not always made available to the audience or for the archive. In some cases, presentation slides consist of mostly figures and graphs, even videos, but without sufficient text bullet points to summarize the content. Meanwhile, there are significant amounts of presentation speech online (e.g. political speech, lectures and seminars) that can be rendered more useful if we can summarize them in a format similar to presentation slides.

Following previous research showing that modeling hierarchical structure indeed helps improve summarization performance, we are interested in going a step further in proposing a rhetorical syntax driven summarization model. Presentation speech is transcribed automatically by an ASR system, then "parsed" into a rhetorical syntax tree. The leaf nodes of the tree, representing actual utterances with rhetorical unit labels, are then "reordered" and organized into a target summary. In this paper, we also propose using a HMSVM (Altun et al., 2003) for the parser and the summarizer. HMSVM has the advantage of considering the interdependence between neighboring sentences. Reordering rules are automatically learned and candidate sequences generated, before they are scored by the final summarizer.

This paper is organized as follows: Section 2 describes our motivation, and the rhetorical structure characteristics in lecture speech. Section 3 details how to parse rhetorical structure of the lecture speech. Section 4 describes the reordering process. Section 5 then describes how to produce extractive summaries. We then describe the corpus, how to create reference summaries, the acoustic/prosodic, linguistic and discourse characteristics of lecture speech, baselines, and our in-house automatic speech recognition system are presented in Section 6. Section 7 presents the experiment results. Section 8 describes related work. We then conclude at the end of this paper.

2 Rhetorical Syntax Tree of Presentation Speech

Unlike conversational speech, lectures and presentations are planned. Lecture speakers follow a relatively rigid rhetorical structure at the document level: s/he starts with an overview of the topic to be presented, followed with the actual content with more detailed descriptions, and then concludes at the end. The speech is given in several coherent "chunks" corresponding to the talk outline. By looking at presentation slides as shown in Figure 1, we can clearly see the chunk boundaries, which always exist at slide transitions, delineate content changes. Each of the chunks, in turn, contains many coherent text spans, namely the rhetorical units. Each rhetorical unit contains one or more slides. We represent the rhetorical structure of presentations by a hierarchical text plan, or a rhetorical tree. Since lecture speeches are mostly based on presentation slides with main bullet points, the structural format of the presentation slides is a faithful representation of the document-level rhetorical structure of the speech. At the top level of the tree are the rhetorical chunks, and at the lower level child nodes are rhetorical units. Each of the chunks contains several rhetorical units, where each unit may contain a number of utterances corresponding to a list of bullet points in the slides.

We propose using the annotation labels commonly shared by most presentation speech for labeling rhetorical chunks as shown in the left column of Table 1. The chunk label definitions are derived from the general structure of presentations in the specific domain of our corpus, namely conference presentations. Note that whereas we chose to use 7 labels for presentation speech, label definitions are fairly obvious and easy to derive for other genres of speech. We use a machine-aided manual annotation method to label the training presentation speech data. Referring to the slides

Table 1: Rhetorical Chunk and Unit Description

Rhetorical Chunk	Rhetorical Unit
c_1 (Title)	title, author of the presentation
c_2 (Outline)	texture structure;
c_3 (Motivation)	aim; problem/phenomenon
c_4 (Related work)	rival/contrast; continuation
c_5 (Methodology)	solution/inventive step
c_6 (Experiment)	corpus description; detailed experimental setup
c_7 (Conclusion)	conclusion; future work

of each presentation, each sentence in the speech transcription is assigned a label corresponding to one of the 7 chunks defined in Table 1. First, all bullet point sentences in the slides are assigned a chunk label according to its section. Referring to our previous work, a Relaxed Dynamic Time Warping program (Zhang et al., 2008) is used to roughly align transcribed sentences to the corresponding slide bullet points. Chunk labels are also included in this alignment. Human inspection quickly corrects any alignment mistakes made by the program. We then label all the sentence of each type "c_i" rhetorical chunk as "c_i". For example, the sentences of a rhetorical chunk which describes "Methodology: solution/inventive step" is labeled as "c_5".

Rhetorical units are described in the right column of Table 1. The rhetorical units, as we explain below, are clustered automatically without explicit labels. These rhetorical units correspond more or less to the definitions in Table 1, without human effort. To obtain the reference rhetorical unit labels of each type of rhetorical chunks in the entire experiment corpus described in Section 6.1, we cluster all utterances that belong to the same chunk in all presentation speech into several rhetorical units by using modified k-means (MKM) clustering algorithm (Wilpon and Rabiner, 1984; Fung et al., 2003). MKM starts from the centroid of all utterances in one rhetorical chunk and splits the clusters top down until the sub-clusters stabilize. Each final cluster represents one rhetorical unit. The clustering algorithm is shown as follows.

Given all utterances within the same type of chunk from all presentation speech:
(1) **Compute** the centroid;
(2) **Assign** sentence feature vectors closest to each centroid to its cluster;
(3) **Update** each centroid feature vector using all sentence feature vectors assigned to each cluster;
(4) **Iterate** step(2) to step (4) until sentence feature vectors stop moving between clusters;
(5) **Stop** if clusters stabilizes, and **get** final clusters, else **goto** step (6);
(6) **Split** the cluster with largest intra-cluster distance into two by finding the pair of vectors as new centroids, and **repeat** steps (2) to step (5).

Using the above algorithm, we find out the following rhetorical unit clusterings of different kinds of rhetorical chunks on our experiment corpus: (1) two rhetorical units in "Title"(c_1) chunk:
"r_i:c_1, $i = 1, 2$";
(2) three rhetorical units in "Outline"(c_2) chunk and "Conclusion"(c_7) chunk:
"r_i:c_j, $i = 1, 2, 3; j = 2, 7$";
(3) five rhetorical units in "Motivation"(c_3) chunk,
"Related work"(c_4) chunk,
"Methodology"(c_5) chunk,
and "Experiment"(c_6) chunk:
"r_i:c_j, $i = 1, 2, 3, 4, 5; j = 3, 4, 5, 6$".

These rhetorical unit clusters correspond roughly to the definitions in the right hand column of Table 1, though no manual labeling is involved.

3 Parsing Presentation Speech

By using our rhetorical syntax-driven model, the process of parsing presentation speech can be described as follows. First we extract acoustic/prosodic features from presentation speech, and linguistic, discourse features from the ASR transcribed text. These features are described in Section 6.1. Next we parse the presentation speech into rhetorical units. Given the ASR transcription of a presentation speech, our task is to parse the transcription sentences into chunks and then into rhetorical units (leaf nodes) that roughly correspond to the rhetorical chunks and units in Table 1 according to their feature vectors. We consider the parsing process as a multi-class classification problem. Each rhetorical unit of each rhetorical chunk is represented by one class.

Considering that HMSVM (Altun et al., 2003) combines the advantages of maximum margin classifier and kernels with the elegance and efficiency of HMMs, and can effectively handle the dependency between neighboring sentences, we train a twenty-eight-class HMSVM classifier for parsing speech, with one class representing each rhetorical unit. As an example, the sentences labeled as "r_2:c_5" belong to the second rhetorical unit of "Methodology"(c_5) chunk. We have found that, by looking at our corpus of conference presentations, speakers indeed follow the "chunk order" of the slides they use. We add some constraints existing between "r_m:c_i" and "r_n:c_j" where "$i \neq j$" according to the "chunk order" of the slides. Function f_1 maps each given speech or transcription **y** to a rhetorical unit label sequence **z**. For example we want to learn a discriminant function $F_1 : \mathcal{Y} \times \mathcal{Z} \rightarrow \mathcal{R}$ over input/output pairs from which we produce a prediction by maximizing F_1 over the output variable for a given input **y**. The general form of our hypotheses f_1 is:

$$\mathbf{z}^* = f_1(\mathbf{y}; \mathbf{w}) = \underset{\mathbf{z} \in \mathcal{Z}}{\operatorname{argmax}} F_1(\mathbf{y}, \mathbf{z}; \mathbf{w}) \quad (1)$$

where **w** denotes a weighting parameter vector to learn.

We assume F_1 to be linear in some combined feature representation of inputs, described in Section 6.1, and outputs $\Psi(\mathbf{y}, \mathbf{z})$. We then get $F_1(\mathbf{y}, \mathbf{z}; \mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{y}, \mathbf{z}) \rangle$. Moreover, we apply a kernel function K over the joint input/output space such that:

$$K((\mathbf{y}, \mathbf{z}), (\bar{\mathbf{y}}, \bar{\mathbf{z}})) = \langle \Psi(\mathbf{y}, \mathbf{z}), \Psi(\bar{\mathbf{y}}, \bar{\mathbf{z}}) \rangle \quad (2)$$

Ψ can be written as a sum over the length of the sequence and decomposed as:

$$\Psi(\mathbf{O}, \mathbf{z}) = (\sum_{i=1}^{l(\mathbf{O})} \Psi_{\sigma,\tau}(\upsilon_i, \upsilon_{i+1}, \mathbf{O}, i))_{\sigma,\tau \in \gamma} \quad (3)$$

where γ is the rhetorical unit label set. $l(\mathbf{O})$ is the length of the observation sequence **O** in our case. Ψ is composed by mapping functions that depend only on labels at position i and $i+1$, **O** as well as i (Markov property).

We then rewrite F_1 using $\mathbf{w} = (\mathbf{w}_{\sigma,\tau})_{\sigma,\tau \in \gamma}$ as Equation 4.

$$\begin{aligned} F_1(\mathbf{O}, \mathbf{z}) &= \sum_{\sigma,\tau \in \gamma} \langle \mathbf{w}_{\sigma,\tau}, \sum_{i=1}^{l(\mathbf{O})} \Psi_{\sigma,\tau}(\upsilon_i, \upsilon_{i+1}, \mathbf{O}, i) \rangle \\ &= \sum_{i=1}^{l(\mathbf{O})} \underbrace{\sum_{\sigma,\tau \in \gamma} \langle \mathbf{w}_{\sigma,\tau}, \Psi_{\sigma,\tau}(\upsilon_i, \upsilon_{i+1}, \mathbf{O}, i) \rangle}_{=:g(\upsilon_i, \upsilon_{i+1}, \mathbf{O}, i)} \end{aligned} \quad (4)$$

In decoding process, using this decomposition (Altun et al., 2003) we can define

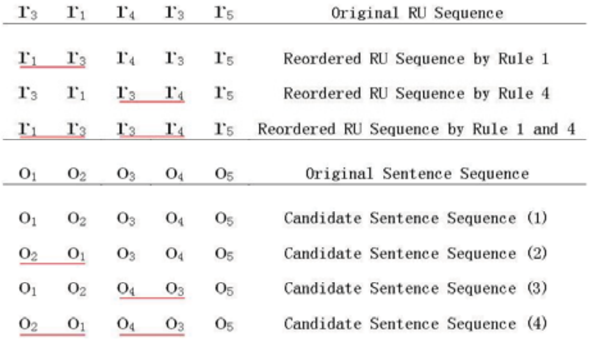
$$\begin{aligned} V(i, \upsilon) := \max_{\upsilon' \in \gamma}(V(i-1, \upsilon')) + g(\upsilon', \upsilon, \mathbf{O}, i-1) \\ when\ i > 1 \\ or\ := 0 \quad otherwise \end{aligned} \quad (5)$$

as the maximal score for all labels with label υ at position i. Using dynamic programming we compute $\max_{\upsilon \in \gamma} V(l(\mathbf{O}), \upsilon)$. The optimal label sequence is recovered by backtracking.

4 Reordering Rhetorical Unit Sequence

4.1 Extracting Reordering Rules

We found that, by looking at our corpus of conference presentations described in Section 6.1, presentation speakers do not always follow the bullet point order within a chunk. When demonstrating a current slide they may be already introducing the next slide. About 11% of the rhetorical units in the transcriptions are out of order vis-a-vis the corresponding bullet points in the presentation slide. As an integral part of our rhetorical syntax-driven summarization model, the rhetorical unit sequence and consequently the sentence sequence are reordered within a chunk. The extraction of reordering rules is based on the alignment between source rhetorical unit sequence in the speech transcription and target rhetorical unit sequence in the Power Point slide sentences. Each sentence is represented by its rhetorical unit label. For example, from the training set and development set in one of our held-out experiment settings described in Section 7, we extracted the following reordering rules: (1)$(r_3, r_1) \rightarrow (r_1, r_3)$;



r_3	r_1	r_4	r_3	r_5	Original RU Sequence
$\underline{r_1}$	$\underline{r_3}$	r_4	r_3	r_5	Reordered RU Sequence by Rule 1
r_3	r_1	$\underline{r_3}$	$\underline{r_4}$	r_5	Reordered RU Sequence by Rule 4
$\underline{r_1}$	$\underline{r_3}$	$\underline{r_3}$	$\underline{r_4}$	r_5	Reordered RU Sequence by Rule 1 and 4
o_1	o_2	o_3	o_4	o_5	Original Sentence Sequence
o_1	o_2	o_3	o_4	o_5	Candidate Sentence Sequence (1)
$\underline{o_2}$	$\underline{o_1}$	o_3	o_4	o_5	Candidate Sentence Sequence (2)
o_1	o_2	$\underline{o_4}$	$\underline{o_3}$	o_5	Candidate Sentence Sequence (3)
$\underline{o_2}$	$\underline{o_1}$	$\underline{o_4}$	$\underline{o_3}$	o_5	Candidate Sentence Sequence (4)

Figure 2: Candidate sentence sequences after applying reordering rules

(2)$(r_3, r_2) \rightarrow (r_2, r_3)$; (3)$(r_4, r_2) \rightarrow (r_2, r_4)$; (4)$(r_4, r_3) \rightarrow (r_3, r_4)$; (5)$(r_5, r_3) \rightarrow (r_3, r_5)$; (6)$(r_5, r_4) \rightarrow (r_4, r_5)$.

In each reordering rule, the left item represents rhetorical unit sequence of the transcription sentences while the right item represents rhetorical unit sequence of bullet points of the corresponding Power Point slides. From these reordering rules, we can see that the speakers in our corpus talk about content described by future bullet points (i.e in the subsequent rhetorical units), but never seem to repeat content from bullet points in the previous unit(s).

4.2 Applying Reordering Rules

Given a sentence sequence and its corresponding rhetorical unit sequence within each chunk, from left to right, with a shifting window of two, we search for the matching reordering rule and adjust the order of the sentences one matched rule at a time, yielding a set of at most 2^L sentence sequence candidates for each chunk where L equals to the length of the sentence. From our data, we found that there are at most 2 matched rules per sentence sequence. So including the original sequence, at most 4 candidate sequences are generated for each chunk.

A reordering example is shown in Figure 2. We apply the reordering rules on a sentence sequence "$(o_1, o_2, o_3, o_4, o_5)$" and the corresponding unit sequence "$(r_3, r_1, r_4, r_3, r_5)$". Four candidate reordered sentence sequence are produced. Without any reordering, we get "Candidate Sentence Sequence (1)". Using reordering Rule 1, we get "Candidate Sentence Sequence (2)".

5 Rhetorical Syntax-driven Summarization

Following sentence reordering, the extractive summarizer selects salient sentences from each chunk using a binary-class classifier. The classifier is run over all candidate sequences from Section 4 and the system selects the best sequence and its summary sentences according to the output probability of the classifier. The best sequence $\{o_1...o_i...o_k\}$ satisfies

$$\text{argmax} \sum_{i=1}^{k} \lg P(o_i \in summary\ sentence\ set | c_j) \qquad (6)$$

where c_j represents the rhetorical chunk c_j which has several candidate sequences, including the sequence $\{o_1...o_i...o_k\}$.
$P(o_i \in summary\ sentence\ set|c_j)$ is output probability of that the sentence o_i in the rhetorical chunk c_j is summary sentence.

Again, an HMSVM classifier is used at this stage. The sentence feature vector **o** now has its rhetorical unit label as an additional feature, to yield a new sentence feature vector $\hat{\mathbf{o}}$. For the sentence vector sequence **z** of each chunk, we label it by using the optimal function $F_2(\mathcal{Z}, \mathcal{V})$. The training stage is similar to that of training the HMSVM parser. The difference is that the HMSVMs for summarization are binary classifiers, while the HMSVM parser is a multi-class classifier.

6 Experimental Setup

6.1 Corpus

We use a lecture speech corpus containing wave files of 71 presentations recorded from different mandarin speakers at two technical conferences, together with well-formatted Power Point slides, manual transcriptions, and their associated audio data. Each presentation lasts about 15 minutes on average. The 71 presentations are split into 391 chunks, and each sentence is assigned a rhetorical chunk label, using the machine-aided human labeling method as described in Section 2. Each chunk has on average 4.3 rhetorical units. The reference rhetorical unit labels are created by using unsupervised MKM algorithm described in Section 2. Since the labeling process also yields an alignment path between transcription sentences and Power Point slide bullet points, we extract those sentences that have the highest alignment scores with the bullet points to form reference summary sentences, then corrected by five human subjects according to the rhetorical chunk descriptions in the right column of Table 1. We use the kappa coefficient (Krippendorff, 1980) for measuring stability of each annotator and reproducibility between each pair of annotators. The average kappa coefficient is higher than 0.85.

6.2 Features and Baselines

We use the discourse feature PossionNoun proposed in our previous work (Zhang et al., 2008) which is based on the following assumptions: first, if a sentence contains new noun words, it probably contains new information. The noun word's Poisson score varies according to its position. We use Poisson distribution to approximate the variation. Second, if a noun word occurs frequently, it is likely to be more important than other noun words, and the sentence with these high frequency noun words should be included in a summary. We also use the following acoustic and linguistic features for representing sentences. The acoustic features are: duration of the sentence, average syllable duration of the sentence, F0 and Energy min/max/mean/slope/range value of the sentence. The linguistic features are: sentence word count, TF/IDF of each word in the sentence, and the word identity in each sentence.

We use three alternate summarization models for comparison. One is a binary classifier without any rhetorical information, one class for summary sentence and the other for non-summary sentence. The second is RSHMM (Fung et al., 2008). The third is our rhetorical syntax tree without reordering. The two above are built by using acoustic, linguistic, and discourse features for representing the sentences. We apply rhetorical unit label as an additional feature for building our proposed models(with/without reordering).

Table 2: Summarization average performance of 6-fold cross validation experiment in ROUGE-L F-measure (F-measure)

Bianry	RSHMM	Without reordering	Syntax tree with reordering
.53(.52)	.63(.56)	.69(.61)	.72(.65)

(1)Binary: binary classifier as baseline
(2)RSHMM: Rhetorical State HMM proposed in our previous work (Fung et al., 2008)
(3)without reordering: rhetorical syntax tree based summarizer without reordering
(4)Syntax tree with reordering: rhetorical syntax tree based summarizer with reordering

6.3 Lecture Speech ASR Transcription System

We apply our rhetorical syntax-driven summarization model for ASR transcriptions. The database for building our in-house ASR system contains 29 hours of audio data from the technical conferences in our corpus. We choose approximately 21 hours of speech as the training data. The test data comprises of 12 presentations with approximately 3 hours of audio data. Our decoding system runs in multiple passes. Automatic segmentation is first performed on the lecture speech audio data. This is followed by bigram decoding with the gender independent (GI) acoustic model (AM) and lattice generation. Trigram and four-gram branches are created for AM adaptation through lattice expansion and rescoring. Re-decoding with both adapted AM and adapted language model (LM) is performed to produce 1-best results. System combination via recognizer output voting error reduction scheme (ROVER) (Fiscus, 1997) is employed by using character based alignment from the trigram and four-gram branch outputs. The final system obtains a recognition performance of 79.2% character accuracy.

7 Experimental Results

For evaluating different summarization systems, we perform 6-fold cross validation experiments, and two held-out experiments. There are many kinds of metrics for evaluating speech summarization performance (Zhu and Penn, 2005; Penn and Zhu, 2008). We choose ROUGE-L F-

Table 3: Summarization performance of held-out experiments in ROUGE-L F-measure (F-measure)

(A) on manual transcription

Binary	RSHMM	Without reordering	Syntax tree with reordering
.50(.48)	.61(.52)	.67(.59)	.69(.61)

(B) on ASR transcription

Binary	RSHMM	Without reordering	Syntax tree with reordering
.46(.43)	.59(.50)	.66(.57)	.68(.61)

(1)Binary: binary classifier as baseline
(2)RSHMM: Rhetorical State HMM proposed in our previous work (Fung et al., 2008)
(3)without reordering: rhetorical syntax tree based summarizer without reordering
(4)Syntax tree with reordering: rhetorical syntax tree based summarizer with reordering

measure (Lin, 2004) and F-measure (Van Rijsbergen, 1979) as evaluation metrics in our experiments. 11 documents from the 71 presentations are excluded as our development set. In the 6-fold cross validation experiments, we divide the remaining 60 presentations into six subsets of equal size. For each experiment, we use five subsets to train all models and the remaining subset for testing. The average performance of these 6-fold cross validation experiments is shown in Table 2.

Among these 60 presentations, 50 are randomly selected as training data for the two held-out experiments, while the remaining ten are used as test data. Table 3-(A) shows the result of the one held-out experiment on manual transcriptions of test data. Table 3-(B) shows the other held-out experiment on ASR transcriptions of test data.

From these results, we can see that the proposed rhetorical syntax-driven summarizer with reordering outperforms all other methods. Table 2 shows that our rhetorical syntax-driven model gives a 35.8% relative improvement over a binary summarizer with no rhetorical information, a 14.3% improvement over RSHMM (Fung et al., 2008), and a 4.3% improvement over our proposed model with no reordering. These findings suggest that our rhetorical syntax-driven summarization model built by using binary HMSVM classifier apply the sequence information of the sentences within the same rhetorical chunk for improving summarization performance because of the Markov property of HMSVM.

In the above experiments, we all use the rhetorical unit labels produced by our rhetorical structure parser for improving summarization performance. The average accuracy of the rhetorical structure parser is about 83.2%. When we use the reference rhetorical unit labels on our cross-validation experiments, the average summarization performance is 0.75 of ROUGE-L F-measure, a 4.2% improvement over that using the rhetorical unit labels produced by the rhetorical structure parser.

Although the overall performance on ASR transcriptions is worse than that of manual transcriptions, the performance is also satisfying. Furthermore, we also find that using only acoustic features our model obtains satisfying result, 0.65 of ROUGE-L F-measure, on 6-fold cross validation experiment.

8 Related Work

(Furui et al., 2008) has shown that feature-based extractive summarization is an approach that is efficient and more effective than MMR-based approach for lecture speech summarization.

(Marcu, 1997) described the first experiment that shows the concepts of rhetorical analysis and nuclearity can be used effectively for text summarization. (Fung et al., 2003) presented a stochastic HMM framework with modified K-means and segmental K-means algorithms for extractive text summarization. (Fung and Ngai, 2006) further presented a stochastic Hidden Markov Story Model for multilingual and multi-document summarization and proposed that monolingual documents recounting the same story (i.e., in the same topic) share a unique story flow (one story, one flow), and such a flow can be modeled by HMMs. (Barzilay and Lee, 2004) presented an unsupervised method for the induction of content models, which capture constraints on topic selection and organization for texts in a particular domain (Branavan et al., 2007) proposed a structured discriminative model for table-of-contents generation on written text that accounts for a wide range of phrase-based and collocation features. (Eisenstein and Barzilay, 2008) describes a novel Bayesian approach to unsupervised lexical cohesion driven topic segmentation.

Many researchers have suggested that rhetorical information also exists in spoken documents and efficient modeling of this information is helpful to the summarization task. (Tatar et al., 2008) and (AKITA and Kawahara, 2007) used the Hearst method (Hearst, 1997) to segment documents and detect topics for text summarization and topic adaptation of speech recognition systems for long speech archives respectively. (Hirohata et al., 2005) consider that humans tend to summarize presentations by extracting important sentences from introduction and conclusion sections, and further propose a summarization method based on this structural characteristic. They estimated the introduction and conclusion section boundaries based on the Hearst method (Hearst, 1997), using sentence cohesiveness which is measured by a cosine value between content word-frequency vectors, before performing summarization. Teufel and Moens (Teufel and Moens, 2002) also proposed "rhetorical status" as summary unit, but without hierarchical structure or reordering. Furthermore, many linguists believe that speech acoustics contribute to rhetorical and discourse structure. (Nakatani et al., 1995) provide empirical evidence that discourses can be segmented reliably, and that acoustic characteristics are used by speakers to convey linguistic structure at the discourse level in the English domain. There is a large amount of previous work seeking to demonstrate that acoustic prosodic profile of speech closely models its discourse or rhetorical structure (Halliday, 1967; Ladd, 1996; Hirschberg and Nakatani, 1996).

Our work described in this paper is closely related to (Marcu, 1997; Teufel and Moens, 2002) in that we propose using rhetorical units for summarization. Our method differs from (Teufel and Moens, 2002) in that we assume the relevance or saliency and function of certain text pieces can be determined by analyzing the full hierarchical structure of the text. Instead of annotating the training data with rhetorical labels manually, we propose using Power Point slides as references. (He et al., 2000; He et al., 1999) also investigate the correlation between Power Point slides and extractive summaries. Our learning method is based on classifiers, while (Marcu, 1997) uses rule-based method for parsing rhetorical structure. We train our rhetorical parser by using those reference rhetorical units of the transcriptions created by aligning Power Point slides. We propose a syntax-driven parsing model with reordering for summarization, and we propose a different classifier, HMSVM, for handling the dependency between neighboring sentences within each chunk, when we accomplish our summarization task.

9 Conclusion and Discussion

In this paper, we have shown a rhetorical syntax-driven summarization method for presentation speech. In view of the fact that rhetorical structure in speech is inherently hierarchical, our method chunks, parses, and reorders the presentation utterance sentences before selecting some of them as summary sentences.

We have proposed to use HMSVM classifiers for the parser and the summarizer, taking into account the dependency between neighboring chunks, rhetorical units and sentences with Markov property. Our rhetorical syntax-driven summarizer with reordering outperforms a binary summarizer without rhetorical information with 35.8% relative improvement and outperforms RSHMM (Fung et al., 2008) with 14.3% relative improvement. It gives a 4.3% relative improvement over the same model without reordering. For future work, we are interested in investigating how to apply our rhetorical syntax-driven method to other genres of speech, such as meetings and parliamentary speech.

10 ACKNOWLEDGEMENT

This work is partially supported by ITS/189/09 of the Hong Kong Innovation and Technology Fund(ITF). The authors would like to thank Chan Ho Yin for his valuable input to this work.

References

AKITA, Y., Nemoto Y. and T. Kawahara. 2007. PLSA-based topic detection in meetings for adaptation of lexicon and language model. *Proc. Interspeech 2007*, pages 602–605.

Altun, Y., I. Tsochantaridis, and T. Hofmann. 2003. Hidden Markov Support Vector Machines. In *Machine Learning-International Workshop Then Conference-*, volume 20.

Barzilay, R. and L. Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. *Proceedings of HLT-NAACL*, pages 113–120.

Branavan, SRK, P. Deshpande, and R. Barzilay. 2007. Generating a table-of-contents. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 544.

Eisenstein, J. and R. Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 334–343. Association for Computational Linguistics.

Fiscus, J.G. 1997. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER. In *In Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding*.

Fung, P. and G. Ngai. 2006. One story, one flow: Hidden Markov Story Models for multilingual multidocument summarization. *ACM Transactions on Speech and Language Processing (TSLP)*, 3(2):1–16.

Fung, P., G. Ngai, and P. Cheung. 2003. Combining optimal clustering and hidden Markov models for extractive summarization. *Proceedings of ACL Workshop on Multilingual Summarization*, pages 29–36.

Fung, P., H.Y. Chan, and J.J. Zhang. 2008. Rhetorical-State Hidden Markov Models For Extractive Speech Summarization. *ICASSP2008. Proceedings*, pages 4957–4960.

Furui, Y., K. Yamamoto, N. Kitaoka, and S. Nakagawa. 2008. Class Lecture Summarization Taking into Account Consecutiveness of Important Sentences. *Proceedings of Interspeech 2008*, pages 2438–2441.

Halliday, M.A.K. 1967. *Intonation and grammar in British English*. Mouton.

He, L., E. Sanocki, A. Gupta, and J. Grudin. 1999. Autosummarization of audio-video presentations. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, page 498. ACM.

He, L., E. Sanocki, A. Gupta, and J. Grudin. 2000. Comparing presentation summaries: slides vs. reading vs. listening. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 177–184. ACM New York, NY, USA.

Hearst, M.A. 1997. TextTiling: Segmenting Text into Multiparagraph Subtopic Passages. *Computational Linguistics*, 23(1):33–64.

Hirohata, M., Y. Shinnaka, K. Iwano, and S. Furui. 2005. Sentence extraction-based presentation summarization techniques and evaluation metrics. *Proc. ICASSP2005*, 1.

Hirschberg, J. and C.H. Nakatani. 1996. A prosodic analysis of discourse segments in direction-giving monologues. *Proceedings of the 34th conference on Association for Computational Linguistics*, pages 286–293.

Krippendorff, K. 1980. Content analysis: An introduction to its methodology. *Beverly Hills,: Sage Publications*.

Ladd, D.R. 1996. *Intonational Phonology*. Cambridge University Press.

Lin, C.Y. 2004. Rouge: A Package for Automatic Evaluation of Summaries. *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, pages 25–26.

Marcu, D. 1997. From discourse structures to text summaries. In *Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization*, pages 82–88.

Murray, G., M. Taboada, and S. Renals. 2006. Prosodic correlates of rhetorical relations. In *Proceedings of the Analyzing Conversations in Text and Speech (ACTS) Workshop at HLT-NAACL*, pages 1–7.

Nakatani, C.H., J. Hirschberg, and B.J. Grosz. 1995. Discourse structure in spoken language: Studies on speech corpora. *AAAI 1995 Spring Symposium Series: Empirical Methods in Discourse Interpretation and Generation*, pages 106–112.

Penn, G. and X. Zhu. 2008. A critical reassessment of evaluation baselines for speech summarization. *Proceedings of ACL-HLT. Columbus, OH*.

Sauper, C. and B. Regina. 2009. Automatically Generating Wikipedia Articles: A Structure-Aware Approach. In *Proceedings of the ACL 2009*, pages 208–216.

Tatar, D., E. Tamaianu-Morita, A. Mihis, and D. Lupsa. 2008. Summarization by Logic Segmentation and Text Entailment. *Advances in Natural Language Processing and Applications*, pages 15–26.

Teufel, S. and M. Moens. 2002. Summarizing scientific articles: experiments with relevance and rhetorical status. *Computational Linguistics*, 28(4):409–445.

Van Rijsbergen, CJ. 1979. Information Retrieval. *Buttersworth, London*.

Wilpon, JG and LR Rabiner. 1984. A modified K-means clustering algorithm for use in speaker-independent isolated word recognition. *The Journal of the Acoustical Society of America*, 75:S93.

Zhang, J.J., S. Huang, and P. Fung. 2008. RSHMM++ for extractive lecture speech summarization. In *IEEE Spoken Language Technology Workshop, 2008. SLT 2008*, pages 161–164.

Zhu, X. and G. Penn. 2005. Evaluation of sentence selection for speech summarization. In *ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization*. Citeseer.

Maximum Metric Score Training for Coreference Resolution

Shanheng Zhao and **Hwee Tou Ng**
Department of Computer Science
National University of Singapore
{zhaosh,nght}@comp.nus.edu.sg

Abstract

A large body of prior research on coreference resolution recasts the problem as a two-class classification problem. However, standard supervised machine learning algorithms that minimize classification errors on the training instances do not always lead to maximizing the F-measure of the chosen evaluation metric for coreference resolution. In this paper, we propose a novel approach comprising the use of instance weighting and beam search to maximize the evaluation metric score on the training corpus during training. Experimental results show that this approach achieves significant improvement over the state-of-the-art. We report results on standard benchmark corpora (two MUC corpora and three ACE corpora), when evaluated using the link-based MUC metric and the mention-based B-CUBED metric.

1 Introduction

Coreference resolution refers to the process of determining whether two or more noun phrases (NPs) in a text refer to the same entity. Successful coreference resolution benefits many natural language processing tasks. In the literature, most prior work on coreference resolution recasts the problem as a two-class classification problem. Machine learning-based classifiers are applied to determine whether a candidate anaphor and a potential antecedent are coreferential (Soon et al., 2001; Ng and Cardie, 2002b).

A large body of prior research on coreference resolution follows the same process: during training, they apply standard supervised machine learning algorithms to minimize the number of misclassified training instances; during testing, they maximize either the local or the global probability of the coreferential relation assignments according to the specific chosen resolution method.

However, minimizing the number of misclassified training instances during training does not guarantee maximizing the F-measure of the chosen evaluation metric for coreference resolution. First of all, coreference is a rare relation. There are far fewer positive training instances than negative ones. Simply minimizing the number of misclassified training instances is suboptimal and favors negative training instances. Secondly, evaluation metrics for coreference resolution are based on global assignments. Not all errors have the same impact on the metric score. Furthermore, the extracted training instances are not equally easy to be classified.

In this paper, we propose a novel approach comprising the use of instance weighting and beam search to address the above issues. Our proposed maximum metric score training (MMST) approach performs maximization of the chosen evaluation metric score on the training corpus during training. It iteratively assigns higher weights to the hard-to-classify training instances. The output of training is a standard classifier. Hence, during testing, MMST is faster than approaches which optimize the assignment of coreferential relations during testing. Experimental results show that MMST achieves significant improvements over the baselines. Unlike most of the previous work, we report improved results over the state-of-the-art on all five standard benchmark corpora

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1308–1316,
Beijing, August 2010

(two MUC corpora and three ACE corpora), with both the link-based MUC metric and the mention-based B-CUBED metric.

The rest of this paper is organized as follows. We first review the related work and the evaluation metrics for coreference resolution in Section 2 and 3, respectively. Section 4 describes the proposed MMST algorithm. Experimental results and related discussions are given in Section 5. Finally, we conclude in Section 6.

2 Related Work

Soon *et al.* (2001) proposed a training and testing framework for coreference resolution. During training, a positive training instance is formed by a pair of markables, i.e., the anaphor (a noun phrase) and its closest antecedent (another noun phrase). Each markable (noun phrase) between the two, together with the anaphor, form a negative training instance. A classifier is trained on all training instances, using a standard supervised learning algorithm. During testing, all preceding markables of a candidate anaphor are considered as potential antecedents, and are tested in a back-to-front manner. The process stops if either an antecedent is found or the beginning of the text is reached. This framework has been widely used in the community of coreference resolution.

Recent work boosted the performance of coreference resolution by exploiting fine-tuned feature sets under the above framework, or adopting alternative resolution methods during testing (Ng and Cardie, 2002b; Yang et al., 2003; Denis and Baldridge, 2007; Versley et al., 2008).

Ng (2005) proposed a ranking model to maximize F-measure during testing. In the approach, n different coreference outputs for each test text are generated, by varying four components in a coreference resolution system, i.e., the learning algorithm, the instance creation method, the feature set, and the clustering algorithm. An SVM-based ranker then picks the output that is likely to have the highest F-measure. However, this approach is time-consuming during testing, as F-measure maximization is performed during testing. This limits its usage on a very large corpus.

In the community of machine learning, researchers have proposed approaches for learning a model to optimize a chosen evaluation metric other than classification accuracy on all training instances. Joachims (2005) suggested the use of support vector machines to optimize nonlinear evaluation metrics. However, the approach does not differentiate between the errors in the same category in the contingency table. Furthermore, it does not take into account inter-instance relation (e.g., transitivity), which the evaluation metric for coreference resolution cares about.

Daume III (2006) proposed a structured learning framework for coreference resolution to approximately optimize the ACE metric. Our proposed approach differs in two aspects. First, we directly optimize the evaluation metric itself, and not by approximation. Second, unlike the incremental local loss in Daume III (2006), we evaluate the metric score globally.

In contrast to Ng (2005), Ng and Cardie (2002a) proposed a rule-induction system with rule pruning. However, their approach is specific to rule induction, and is not applicable to other supervised learning classifiers. Ng (2004) varied different components of coreference resolution, choosing the combination of components that results in a classifier with the highest F-measure on a held-out development set during training. In contrast, our proposed approach employs instance weighting and beam search to maximize the F-measure of the evaluation metric during training. Our approach is general and applicable to any supervised learning classifiers.

Recently, Wick and McCallum (2009) proposed a partition-wise model for coreference resolution to maximize a chosen evaluation metric using the Metropolis-Hastings algorithm (Metropolis et al., 1953; Hastings, 1970). However, they found that training on classification accuracy, in most cases, outperformed training on the coreference evaluation metrics. Furthermore, similar to Ng (2005), their approach requires the generation of multiple coreference assignments during testing.

Vemulapalli *et al.* (2009) proposed a document-level boosting technique for coreference resolution by re-weighting the documents that have the lowest F-measures. By combining multiple classifiers generated in multiple iterations, they achieved a CEAF score slightly better than the baseline. Different from them, our approach works at the instance level, and we output a single classifier.

3 Coreference Evaluation Metrics

In this section, we review two commonly used evaluation metrics for coreference resolution.

First, we introduce the terminology. The gold standard annotation and the output by a coreference resolution system are called key and response, respectively. In both the key and the response, a coreference chain is formed by a set of coreferential mentions. A *mention* (or markable) is a noun phrase which satisfies the markable definition in an individual corpus. A *link* refers to a pair of coreferential mentions. If a mention has no links to other mentions, it is called a *singleton*.

3.1 The MUC Evaluation Metric

Vilain *et al.* (1995) introduced the link-based MUC evaluation metric for the MUC-6 and MUC-7 coreference tasks. Let S_i be an equivalence class generated by the key (i.e., S_i is a coreference chain), and $p(S_i)$ be a partition of S_i relative to the response. Recall is the number of correctly identified links over the number of links in the key.

$$Recall = \frac{\sum(|S_i| - |p(S_i)|)}{\sum(|S_i| - 1)}$$

Precision, on the other hand, is defined in the opposite way by switching the role of key and response. F-measure is a trade-off between recall and precision.

$$F = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision}$$

3.2 The B-CUBED Evaluation Metric

Bagga and Baldwin (1998) introduced the mention-based B-CUBED metric. The B-CUBED metric measures the accuracy of coreference resolution based on individual mentions. Hence, it also gives credit to the identification of singletons, which the MUC metric does not. Recall is computed as

$$Recall = \frac{1}{N} \sum_{d \in D} \sum_{m \in d} \frac{|O_m|}{|S_m|}$$

where D, d, and m are the set of documents, a document, and a mention, respectively. S_m is the equivalence class generated by the key that contains m, while O_m is the overlap of S_m and the equivalence class generated by the response that contains m. N is the total number of mentions in D. The precision, again, is computed by switching the role of key and response. F-measure is computed in the same way as the MUC metric.

4 Maximum Metric Score Training

Before explaining the algorithm, we describe our coreference clustering method used during testing. It is the same as most prior work in the literature, including Soon *et al.* (2001) and Ng and Cardie (2002b). The individual classification decisions made by the coreference classifier do not guarantee that transitivity of coreferential NPs is obeyed. So it can happen that the pair A and B, and the pair B and C are both classified as coreferential, but the pair A and C is not classified as coreferential by the classifier. After all coreferential markable pairs are found (no matter by closest-first, best-first, or resolving-all strategies as in different prior work), all coreferential pairs are clustered together to form the coreference output. By doing so, transitivity is kept: a markable is in a coreference chain if and only if it is classified to be coreferential to at least one other markable in the chain.

4.1 Instance Weighting

Suppose there are m_k and m_r coreferential links in the key and the response, respectively, and a coreference resolution system successfully predicts n correct links. The recall and the precision are then $\frac{n}{m_k}$ and $\frac{n}{m_r}$, respectively. The learnt classifier predicts false positive and false negative instances during testing. For a false positive instance, if we could successfully predict it as negative, the recall is unchanged, but the precision will be $\frac{n}{m_r - 1}$, which is higher than the original precision $\frac{n}{m_r}$. For a false negative instance, it is more subtle. If the two markables in the instance are determined to be in the same coreference chain by the clustering algorithm, it does not matter whether we predict this instance as positive or negative, i.e., this false negative does not change the F-measure of the evaluation metric at all. If the two markables are not in the same coreference chain under the clustering, in case that we can predict it as positive, the recall will be $\frac{n+1}{m_k}$, which is higher than the original recall $\frac{n}{m_k}$, and the precision will be $\frac{n+1}{m_r+1}$, which is higher than the original precision $\frac{n}{m_r}$, as $n < m_r$. In both cases, the F-measure improves. If we can instruct the learning algorithm to pay more attention to these false positive and false negative instances and to predict them correctly by assigning them more weight, we should be able to improve the F-measure.

In the literature, besides the training instance extraction methods proposed by Soon *et al.* (2001) and Ng and Cardie (2002b) as discussed in Section 2, McCarthy and Lehnert (1995) used all possible pairs of training instances. We also use all pairs of training instances in our approach to keep as much information as possible. Initially all the pairs are equally weighted. We then iteratively assign more weights to the hard-to-classify pairs. The iterative process is conducted by a beam search algorithm.

4.2 Beam Search

Our proposed MMST algorithm searches for a set of weights to assign to training instances such that the classifier trained on the weighted training instances gives the maximum coreference metric score when evaluated on the training instances. Beam search is used to limit the search. Each search state corresponds to a set of weighted training instances, a classifier trained on the weighted training instances minimizing misclassifications, and the F-measure of the classifier when evaluated on the weighted training instances using the chosen coreference evaluation metric. The root of the search tree is the initial search state where all the training instances have identical weights of one. Each search state s can expand into two different children search states s_l and s_r. s_l (s_r) corresponds to assigning higher weights to the false positive (negative) training instances in s. The search space thus forms a binary search tree.

Figure 1 shows an example of a binary search tree. Initially, the tree has only one node: the root (node 1 in the figure). In each iteration, the algorithm expands all the leaf nodes in the beam. For example, in the first iteration, node 1 is expanded to generate node 2 and 3, which corresponds to adding weights to false positive and false negative training instances, respectively. An expanded node always has two children in the binary search tree. All the nodes are then sorted in descending order of F-measure. Only the top M nodes are kept, and the remaining nodes are discarded. The discarded nodes can either be leaf nodes or non-leaf nodes. For example, if node 5 is discarded because of low F-measure, it will not be expanded to generate children in the binary search tree. The iterative algorithm stops when all the nodes in the beam are non-leaf nodes, i.e., all the nodes in the beam have been expanded.

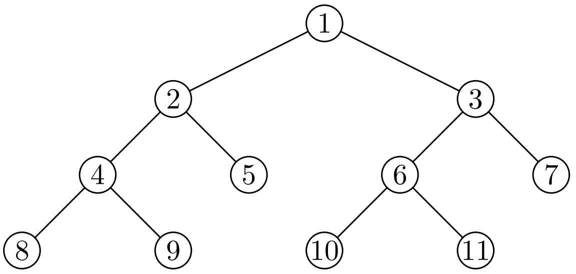


Figure 1: An example of a binary search tree

Figure 2 gives the formal description of the proposed maximum metric score training algorithm. In the algorithm, assume that we have N texts T_1, T_2, ..., T_N in the training data set. m_{ki} and m_{kj} are the ith and jth markable in the text T_k, respectively. Hence, for all $i < j$, $(m_{ki}, m_{kj}, w_{kij})$ is a training instance for the markable pair (m_{ki}, m_{kj}), in which w_{kij} is the weight of the instance. Let L_{kij} and L'_{kij} be the true and predicted label of the pair (m_{ki}, m_{kj}), respectively. Let W, C, F, and E be the set of weights $\{w_{kij} | 1 \leq k \leq N, i < j\}$, the classifier, the F-measure, and a boolean indicator of whether the search state has been expanded, respectively. Finally, M is the beam size, and δ controls how much we update the weights in each iteration.

Since we train the model on all possible pairs, during testing we also test if a potential anaphor is coreferential to each preceding antecedent.

```
INPUT: $T_1, T_2, \ldots, T_N$
OUTPUT: classifier $C$
$w_{kij} \leftarrow 1$, for all $1 \leq k \leq N$ and $i < j$
$C \leftarrow$ train($\{(m_{ki}, m_{kj}, w_{kij}) | 1 \leq k \leq N, i < j\}$)
$F \leftarrow$ resolve and evaluate $T_1, \ldots, T_N$ with $C$
$E \leftarrow false$
BEAM$\leftarrow \{(W, C, F, E)\}$
repeat
  BEAM$' \leftarrow \{\}$
  for all $(W, C, F, E)$ in BEAM do
    BEAM$' \leftarrow$ BEAM$' \bigcup \{(W, C, F, true)\}$
    if E=false then
      predict all $L'_{kij}$ with $C$ ($1 \leq k \leq N, i < j$)
      cluster into coreference chains based on $L'_{kij}$
      $W' \leftarrow W$
      for all $1 \leq k \leq N, i < j$ do
        if $L_{kij} = false$ and $L'_{kij} = true$ then
          $w'_{kij} \leftarrow w'_{kij} + \delta$
        end if
      end for
      $C' \leftarrow$ train($\{(m_{ki}, m_{kj}, w'_{kij}) | 1 \leq k \leq N, i < j\}$)
      $F' \leftarrow$ resolve and evaluate $T_1, \ldots, T_N$ with $C'$
      BEAM$' \leftarrow$ BEAM$' \bigcup \{(W', C', F', false)\}$
      $W'' \leftarrow W$
      for all $1 \leq k \leq N, i < j$ do
        if $L_{kij} = true$ and $L'_{kij} = false$ and
          $Chain(m_{ki}) \neq Chain(m_{kj})$ then
           $w''_{kij} \leftarrow w''_{kij} + \delta$
        end if
      end for
      $C'' \leftarrow$ train($\{(m_{ki}, m_{kj}, w''_{kij}) | 1 \leq k \leq N, i < j\}$)
      $F'' \leftarrow$ resolve and evaluate $T_1, \ldots, T_N$ with $C''$
      BEAM$' \leftarrow$ BEAM$' \bigcup \{(W'', C'', F'', false)\}$
    end if
  end for
  BEAM$\leftarrow$ BEAM$'$
  sort BEAM in descending order of $F$, keep top $M$ elements
until for all $E$ of all elements in BEAM, $E = true$
return $C$, from the top element $(W, C, F, E)$ of BEAM
```

Figure 2: The maximum metric score training (MMST) algorithm

5 Experiments

5.1 Experimental Setup

In the experiments, we used all the five commonly used evaluation corpora for coreference resolution, namely the two MUC corpora (MUC6 and MUC7) and the three ACE corpora (BNEWS, NPAPER, and NWIRE). The MUC6 and the MUC7 corpora were defined in the DARPA Message Understanding Conference (MUC-6, 1995; MUC-7, 1998). The dry-run texts were used as the training data sets. In both corpora, each training data set contains 30 texts. The test data sets for MUC6 and MUC7 consist of the 30 and 20 formal evaluation texts, respectively. The ACE corpora were defined in NIST Automatic Content Extraction phase 2 (ACE-2) (NIST, 2002). The three data sets are from different news sources: broadcast news (BNEWS), newspaper (NPAPER), and newswire (NWIRE). Each of the three data sets contains two portions: training and development test. They were used as our training set and test set, respectively. The BNEWS, NPAPER, and NWIRE data sets contain 216, 76, and 130 training texts, and 51, 17, and 29 test texts, respectively.

Unlike some previous work on coreference resolution that assumes that the gold standard markables are known, we work directly on raw text input. Versley *et al.* (2008) presented the BART package[1], an open source coreference resolution toolkit, that accepts raw text input and reported state-of-the-art MUC F-measures on the three ACE corpora. BART uses an extended feature set and tree kernel support vector machines (SVM) under the Soon *et al.* (2001) training and testing framework. We used the BART package in our experiments, and implemented the proposed MMST algorithm on top of it. In our experiments reported in this paper, the features we used are *identical* to the features output by the preprocessing code of BART reported in Versley *et al.* (2008), except that we did not use their tree-valued and string-valued features (see the next subsection for details).

Since we use automatically extracted markables, it is possible that some extracted markables and the gold standard markables are unmatched, or *twinless* as defined in Stoyanov *et al.* (2009). How to use the B-CUBED metric for evaluating twinless markables has been explored recently. In this paper, we adopt the $B^3 all$ variation proposed by Stoyanov *et al.* (2009), which retains all twinless markables. We also experimented with their $B^3 0$ variation, which gave similar results. Note that no matter which variant of the B-CUBED metric is used, it is a fair comparison as long as the baseline and our proposed MMST algorithm are compared against each other using the same B-CUBED variant.

5.2 The Baseline Systems

We include state-of-the-art coreference resolution systems in the literature for comparison. Since we use the BART package in our experiments,

[1] `http://www.sfs.uni-tuebingen.de/~versley/BART/`

we include the results of the original BART system (with its extended feature set and SVM-light-TK (Moschitti, 2006), as reported in Versley *et al.* (2008)) as the first system for comparison. Versley *et al.* (2008) reported only the results on the three ACE data sets with the MUC evaluation metric. Since we used all the five data sets in our experiments, for fair comparison, we also include the MUC results reported in Ng (2004). To the best of our knowledge, Ng (2004) was the only prior work which reported MUC metric scores on all the five data sets. The MUC metric scores of Versley *et al.* (2008) and Ng (2004) are listed in the row "Versley *et al.* 08" and "Ng 04", respectively, in Table 1. For the B-CUBED metric, we include Ng (2005) for comparison, although it is unclear how Ng (2005) interpreted the B-CUBED metric. The scores are listed in the row "Ng 05" in Table 2.

Tree kernel SVM learning is time-consuming. To reduce the training time needed, instead of using SVM-light-TK, we used a much faster learning algorithm, J48, which is the WEKA implementation of the C4.5 decision tree learning algorithm. (Quinlan, 1993; Witten and Frank, 2005). As tree-valued features and string-valued features cannot be used with J48, in our experiments we excluded them from the extended feature set that BART used to produce state-of-the-art MUC F-measures on the three ACE corpora. All our results in this paper were obtained using this reduced feature set and J48 decision tree learning. However, given sufficient computational resources, our proposed approach is able to apply to any supervised machine learning algorithms.

Our baselines that follow the Soon *et al.* (2001) framework, using the reduced feature set and J48 decision tree learning, are shown in the row "*SNL*-Style Baseline" in Table 1 and 2. The results suggest that our baseline system is comparable to the state of the art. Although in Table 1, the performance of the *SNL*-style baseline is slightly lower than Versley *et al.* (2008) on the three ACE corpora, the computational time needed has been greatly reduced.

Our MMST algorithm trains and tests on all pairs of markables. To show the effectiveness of weight updating of MMST, we built another baseline which trains and tests on all pairs. The performance of this system is shown in the row "*All*-Style Baseline" in Table 1 and 2.

5.3 Results Using Maximum Metric Score Training

Next, we show the results of using the proposed maximum metric score training algorithm. From the description of the algorithm, it can be seen that there are two parameters in the algorithm. One parameter is M, the size of the beam. The other parameter is δ, which controls how much we increase the weight of a training instance in each iteration.

Since the best M and δ for the MUC evaluation metric were not known, we used held-out development sets to tune the parameters. Specifically, we trained classifiers with different combinations of M and δ on a development training set, and evaluated their performances on a development test set. In our experiments, the development training set contained 2/3 of the texts in the training set of each individual corpus, while the development test set contained the remaining 1/3 of the texts. After having picked the best M and δ values, we trained a classifier on the entire training set with the chosen parameters. The learnt classifier was then applied to the test set.

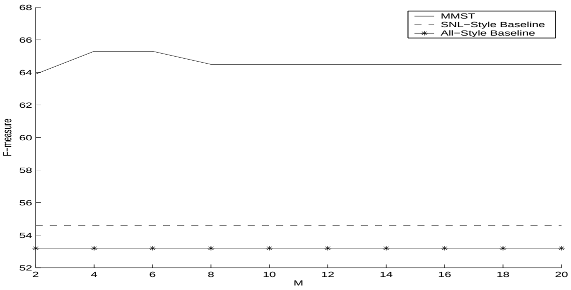


Figure 3: Tuning M on the held-out development set

To limit the search space, we tuned the two parameters sequentially. First, we fixed $\delta = 1$, which is equivalent to duplicating each training instance once in J48, and evaluated $M = 2, 4, 6, \ldots, 20$. After having chosen the best M that corresponded to the maximum F-measure, we fixed the value of M, and evaluated $\delta = 0.1, 0.2, 0.3, \ldots, 2.0$. Take MUC6 as an exam-

	MUC6			MUC7			BNEWS			NPAPER			NWIRE		
Model	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Versley *et al.* 08	–			–			60.7	65.4	63.0	64.1	67.7	65.8	60.4	65.2	62.7
Ng 04	75.8	61.4	**67.9**	64.2	60.2	62.1	63.1	67.8	65.4	73.5	63.3	**68.0**	53.1	60.6	56.6
SNL-Style Baseline	67.0	49.2	56.7	63.0	54.2	58.3	57.4	64.3	60.7	61.6	67.3	64.3	58.6	66.1	62.1
All-Style Baseline	56.9	69.2	62.5	51.5	73.4	60.6	53.0	76.7	62.7	56.3	75.4	64.4	53.0	74.5	61.9
MMST	73.3	59.9	$65.9^{**\dagger\dagger}$	66.8	59.8	$\mathbf{63.1}^{**\dagger}$	70.5	61.9	$\mathbf{65.9}^{**}\dagger$	69.9	64.0	$66.8^{\dagger}$	64.7	64.7	$\mathbf{64.7}^{**\dagger}$
	$M = 6,\ \delta = 1.0$			$M = 6,\ \delta = 0.7$			$M = 6,\ \delta = 1.8$			$M = 6,\ \delta = 0.9$			$M = 14,\ \delta = 0.7$		

Table 1: Results for the two MUC and three ACE corpora with MUC evaluation metric

	MUC6			MUC7			BNEWS			NPAPER			NWIRE		
Model	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Ng 05	–			–			57.0	77.1	65.6	62.8	71.2	66.7	59.3	75.4	66.4
SNL-Style Baseline	57.8	74.4	65.1	57.6	76.5	65.7	62.0	74.7	67.8	61.8	70.4	65.8	65.8	75.9	70.5
All-Style Baseline	51.6	86.3	64.6	49.1	90.1	63.6	61.6	83.7	**71.0**	63.9	74.0	68.6	64.8	80.1	**71.7**
MMST	62.7	81.5	$\mathbf{70.9}^{**\dagger\dagger}$	61.8	73.6	$\mathbf{67.2}^{\dagger\dagger}$	61.6	83.7	$\mathbf{71.0}^{**}$	63.1	76.2	$\mathbf{69.1}^{**}$	64.3	81.0	**71.7**
	$M = 6,\ \delta = 1.0$			$M = 8,\ \delta = 0.8$			$M = 6,\ \delta = 0.9$			$M = 14,\ \delta = 0.5$			$M = 6,\ \delta = 0.1$		

Table 2: Results for the two MUC and three ACE corpora with B^3 evaluation metric

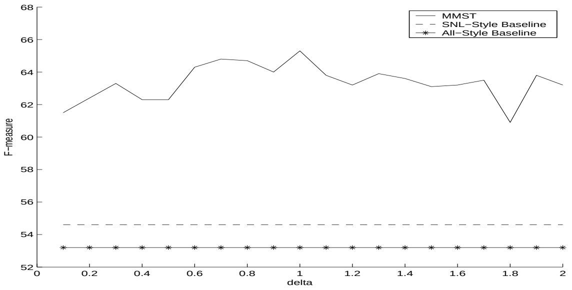


Figure 4: Tuning δ on the held-out development set

ple. The results of tuning M on MUC6 are shown in Figure 3. The maximum F-measure is obtained when $M = 4$ and $M = 6$. On all the different M values we have tried, MMST outperforms both the *SNL*-style baseline and the *All*-style baseline on the development test set. We then fixed $M = 6$, and evaluated different δ values. The results are shown in Figure 4. The best F-measure was obtained when $\delta = 1.0$. Again, on all the different δ values we have tried, MMST outperforms both baselines on the development test set.

The rows "MMST" in Table 1 and 2 show the performance of MMST on the test sets, with the tuned parameters indicated. In our experiments, the statistical significance test was conducted as in Chinchor (1995). * and ** stand for $p < 0.05$ and $p < 0.01$ over the *SNL*-style baseline, respectively. † and †† stand for $p < 0.05$ and $p < 0.01$ over the *All*-style baseline, respectively.

For the MUC metric, when compared to the *All*-style baseline, MMST gains 3.4, 2.5, 3.2, 2.4, and 2.8 improvement in F-measure on MUC6, MUC7, BNEWS, NPAPER, and NWIRE, respectively. The experimental results clearly show that MMST gains not only consistent, but also statistically significant improvement over both the *SNL*-style baseline and the *All*-style baseline in all combinations (five data sets and two baselines) on the MUC metric, except that it is not significant ($p = 0.06$) over the *SNL*-style baseline in NPAPER. As for the B-CUBED metric, MMST gains significant improvement in F-measure on MUC6 and MUC7 data sets, while its performance on the three ACE data sets are comparable to the *All*-style baseline.

5.4 Discussion

To see how MMST actually updates the weight, we use the MUC metric as an example. Under the experimental settings, it takes 6 – 9 iterations for MMST to stop on the five data sets. The number of explored states in the binary search tree, including the root, is 33, 39, 25, 29, and 75 on MUC6, MUC7, BNEWS, NPAPER, and NWIRE, respectively. It is instructive to find out the final weight of each instance. Take MUC6 as an example, the number of positive instances with weight 1, 2, 3, and 4 are 5,204, 1,568, 1,379, and 1,844, respectively, while the number of negative instances with weight 1 and 2 are 503,141 and 1,755, respectively. Counting the weighted number of instances (e.g., an instance with weight 2 is equivalent to 2 instances), we have 19,853 positive and 506,651 negative training instances. This changes the ratio of the positive instances from 1.9% to 3.8%. As a by-product, MMST reduces data skewness, while using all possible NP pairs for training to keep as much information as possible.

The change of weights of the training instances is equivalent to the change of distribution of the training instances. This effectively changes the classification hypothesis to the one that tends to yield higher evaluation metric score. Take the following sentence in the MUC6 data set as an example:

> In a news release, ***the company*** said the new name more accurately reflects ***its*** focus on high-technology communications, including business and entertainment software, interactive media and wireless data and voice transmission.

In the above example, the pronoun *its* is coreferential to the antecedent NP *the company*. The baseline classifier gives a probability of 0.02 that the two NPs are coreferential. The pair is classified wrongly and none of the other pairs in the article can link the two NPs together through clustering. However, with MMST, this probability increases to 0.54, which leads to the correct classification. This is because the baseline classifier is not good at predicting in the case when the second markable is a pronoun. In the above example, *its* can have another candidate antecedent *the new name*. There are far more negative training instances than positive ones for this case. In fact, in the induced decision tree by the baseline, the leaf node corresponding to the pair *the company* – *its* has 7,782 training instances, out of which only 175 are positive. With MMST, however, these numbers decrease to 83 and 45, respectively. MMST also promotes the Anaphor_Is_Pronoun feature to a higher level in the decision tree. Although we use decision tree to illustrate the working of the algorithm, MMST is not limited to tree learning, and can make use of any learning algorithms that are able to take advantage of instance weighting.

It can also be seen that with the B-CUBED metric, MMST gains improvement on MUC6 and MUC7, but not on the three ACE corpora. However, the results of MMST on the three ACE corpora with the B-CUBED evaluation metric are at least comparable with the *All*-style baseline. This is because we always pick the classifier which corresponds to the maximum evaluation metric score on the training set and the classifier corresponding to the *All*-style baseline is one of the candidates. In addition, our MMST approach improves upon state-of-the-art results (Ng, 2004; Ng, 2005; Versley et al., 2008) on most of the five standard benchmark corpora (two MUC corpora and three ACE corpora), with both the link-based MUC metric and the mention-based B-CUBED metric.

Finally, our approach performs all the F-measure maximization during training, and is very fast during testing, since the output of the MMST algorithm is a standard classifier. For example, on the MUC6 data set with the MUC evaluation metric, it took 1.6 hours and 31 seconds for training and testing, respectively, on an Intel Xeon 2.33GHz machine.

6 Conclusion

In this paper, we present a novel maximum metric score training approach comprising the use of instance weighting and beam search to maximize the chosen coreference metric score on the training corpus during training. Experimental results show that the approach achieves significant improvement over the baseline systems. The proposed approach improves upon state-of-the-art results on most of the five standard benchmark corpora (two MUC corpora and three ACE corpora), with both the link-based MUC metric and the mention-based B-CUBED metric.

Acknowledgments

We thank Yannick Versley for providing us the BART package and the preprocessed data. This research was done for CSIDM Project No. CSIDM-200804 partially funded by a grant from the National Research Foundation (NRF) administered by the Media Development Authority (MDA) of Singapore.

References

Bagga, Amit and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the LREC1998*, pages 563–566.

Chinchor, Nancy. 1995. Statistical significance of MUC-6 results. In *Proceedings of the MUC-6*, pages 39–43.

Daume III, Hal. 2006. *Practical Structured Learning for Natural Language Processing*. Ph.D. thesis, University of Southern California.

Denis, Pascal and Jason Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *Proceedings of the NAACL-HLT2007*, pages 236–243.

Hastings, W. K. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Joachims, Thorsten. 2005. A support vector method for multivariate performance measures. In *Proceedings of the ICML2005*, pages 377–384.

McCarthy, Joseph F. and Wendy G. Lehnert. 1995. Using decision trees for coreference resolution. In *Proceedings of the IJCAI1995*, pages 1050–1055.

Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087–1092.

Moschitti, Alessandro. 2006. Making tree kernels practical for natural language learning. In *Proceedings of the EACL2006*, pages 113–120.

MUC-6. 1995. Coreference task definition (v2.3, 8 Sep 95). In *Proceedings of the MUC-6*, pages 335–344.

MUC-7. 1998. Coreference task definition (v3.0, 13 Jul 97). In *Proceedings of the MUC-7*.

Ng, Vincent and Claire Cardie. 2002a. Combining sample selection and error-driven pruning for machine learning of coreference rules. In *Proceedings of the EMNLP2002*, pages 55–62.

Ng, Vincent and Claire Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *Proceedings of the ACL2002*, pages 104–111.

Ng, Vincent. 2004. *Improving Machine Learning Approaches to Noun Phrase Coreference Resolution*. Ph.D. thesis, Cornell University.

Ng, Vincent. 2005. Machine learning for coreference resolution: From local classification to global ranking. In *Proceedings of the ACL2005*, pages 157–164.

NIST. 2002. The ACE 2002 evaluation plan. `ftp://jaguar.ncsl.nist.gov/ace/doc/ACE-EvalPlan-2002-v06.pdf`.

Quinlan, J. Ross. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

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

Stoyanov, Veselin, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of the ACL-IJCNLP2009*, pages 656–664.

Vemulapalli, Smita, Xiaoqiang Luo, John F. Pitrelli, and Imed Zitouni. 2009. Classifier combination techniques applied to coreference resolution. In *Proceedings of the NAACL-HLT2009 Student Research Workshop and Doctoral Consortium*, pages 1–6.

Versley, Yannick, Simone Paolo Ponzetto, Massimo Poesio, Vladimir Eidelman, Alan Jern, Jason Smith, Xiaofeng Yang, and Alessandro Moschitti. 2008. BART: A modular toolkit for coreference resolution. In *Proceedings of the ACL2008:HLT Demo Session*, pages 9–12.

Vilain, Marc, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the MUC-6*, pages 45–52.

Wick, Michael and Andrew McCallum. 2009. Advances in learning and inference for partition-wise models of coreference resolution. Technical Report UM-CS-2009-028, University of Massachusets, Amherst, USA.

Witten, Ian H. and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, second edition.

Yang, Xiaofeng, Guodong Zhou, Jian Su, and Chew Lim Tan. 2003. Coreference resolution using competition learning approach. In *Proceedings of the ACL2003*, pages 176–183.

Paraphrasing with Search Engine Query Logs

Shiqi Zhao[†‡], **Haifeng Wang**[†], **and Ting Liu**[‡]
[†]Baidu Inc.
[‡]HIT Center for Information Retrieval, Harbin Institute of Technology
{zhaoshiqi, wanghaifeng}@baidu.com, tliu@ir.hit.edu.cn

Abstract

This paper proposes a method that extracts paraphrases from search engine query logs. The method first extracts paraphrase query-title pairs based on an assumption that a search query and its corresponding clicked document titles may mean the same thing. It then extracts paraphrase query-query and title-title pairs from the query-title paraphrases with a pivot approach. Paraphrases extracted in each step are validated with a binary classifier. We evaluate the method using a query log from Baidu[1], a Chinese search engine. Experimental results show that the proposed method is effective, which extracts more than 3.5 million pairs of paraphrases with a precision of over 70%. The results also show that the extracted paraphrases can be used to generate high-quality paraphrase patterns.

1 Introduction

The use of paraphrases is ubiquitous in human languages, which also presents a challenge for natural language processing (NLP). Previous studies have shown that paraphrasing can play important roles in plenty of areas, such as machine translation (MT) (Callison-Burch et al., 2006; Kauchak and Barzilay, 2006), question answering (QA) (Duboue and Chu-Carroll, 2006; Riezler et al., 2007), natural language generation (NLG) (Iordanskaja et al., 1991), and so on. As a result, the research on paraphrasing and its applications have attracted significant interest.

This paper proposes a method that uses search engine query logs for extracting paraphrases, which is illustrated in Figure 1. Specifically, three kinds of paraphrases can be extracted with our method, which include (1) query-title (Q-T): a query and a document title that users clicked on; (2) query-query (Q-Q): two queries, for which users clicked on the same document title; (3) title-title (T-T): two titles that users clicked on for the same query. We train a classifier for each kind to filter incorrect pairs and refine the paraphrases.

Extracting paraphrases using query logs has many advantages. First, query logs keep growing, which have no scale limitation. Second, query logs reflect web users' real needs, hence the extracted paraphrases may be more useful than that from other kinds of corpora. Third, paraphrases extracted from query logs can be directed applied in search engines for query suggestion and document reranking. In addition, we find that both queries and titles contain a good many question sentences, which can be useful in developing QA systems.

We conduct experiments using a query log of a commercial Chinese search engine Baidu, from which we extracted about 2.7 million pairs of paraphrase Q-T, 0.4 million pairs of paraphrase Q-Q, and 0.4 million pairs of paraphrase T-T. The precision of the paraphrases is above 70%. In addition, we generate paraphrase patterns using the extracted paraphrases. The results show that 73,484 pairs of paraphrase patterns have been generated, with a precision of over 78%.

In the rest of the paper, we first review related work in Section 2. Section 3 describes our method in detail. Section 4 presents the evaluation and re-

[1] www.baidu.com

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1317–1325,
Beijing, August 2010

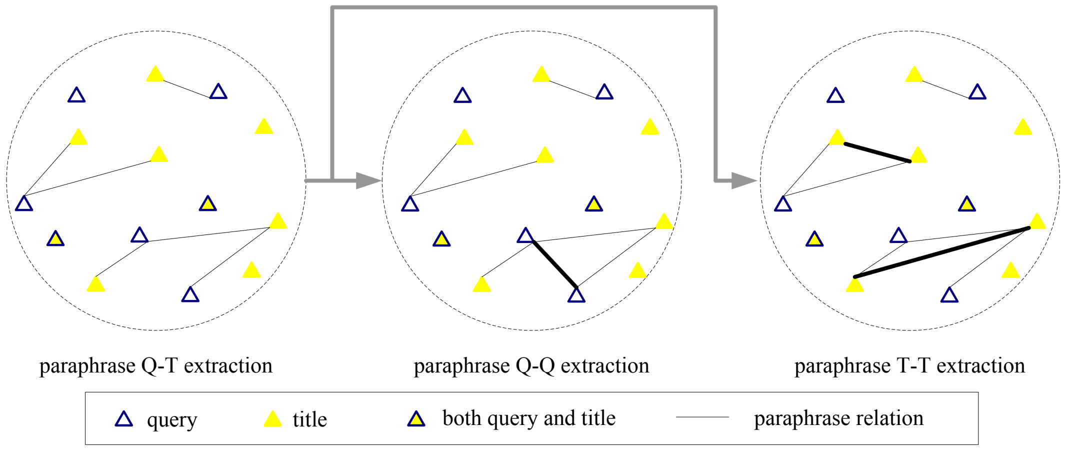

paraphrase Q-T extraction
paraphrase Q-Q extraction
paraphrase T-T extraction
query
title
both query and title
paraphrase relation

Figure 1: Illustration of the proposed method.

sults. Section 5 concludes the paper and discusses future directions.

2 Related Work

In this section, we briefly review previous studies on paraphrase extraction and query log mining in information retrieval (IR).

2.1 Paraphrase Extraction

A variety of data resources have been exploited for paraphrase extraction. For example, some researchers extract paraphrases from multiple translations of the same foreign novel (Barzilay and McKeown, 2001; Ibrahim et al., 2003), while some others make use of comparable news articles that report on the same event within a small time interval (Shinyama et al., 2002; Barzilay and Lee, 2003; Dolan et al., 2004). Besides the monolingual corpora, bilingual parallel corpora have also been used for extracting paraphrases (Bannard and Callison-Burch, 2005; Callison-Burch, 2008; Zhao et al., 2008). Their basic assumption is that phrases that align with the same foreign phrase may have the same meaning.

The above methods have achieved promising results. However, their performances are usually constrained due to the scale and domain limitation. As an alternative, researchers have tried to acquire paraphrases from large-scale web corpora (Lin and Pantel, 2001; Paşca and Dienes, 2005; Bhagat and Ravichandran, 2008) or directly based on web mining (Ravichandran and Hovy, 2002). These methods are guided by an extended version of distributional hypothesis, namely, if two phrases often occur in similar contexts, their meanings tend to be similar. The disadvantage of these methods is that the underlying assumption does not always hold. Phrases with opposite meanings can also occur in similar contexts, such as “X solves Y” and “X worsens Y” (Lin and Pantel, 2001). In addition, the extracted paraphrases are generally short fragments with two slots (variables) at both ends.

2.2 Query Log Mining in IR

Query logs are widely used in the IR community, especially for mining similar queries. For example, Wen et al. (2002) clustered queries based on user click information. Their basic idea is that if some queries result in similar user clicks, the meanings of these queries should be similar. Such methods have also been investigated in (Gao et al., 2007) for cross-lingual query suggestion and (Zhao et al., 2007) for synonymous questions identification. This paper is partly inspired by their studies. However, we do not simply use click information as clues for mining similar queries. Instead, we mine paraphrases across queries and clicked document titles.

In addition, query logs can be used for query expansion. For instance, Cui et al. (2002) extract probabilistic correlations between query terms and document terms by analyzing query logs, which are then used to select high-quality

H1: If a query q hits a title t, then q and t are likely to be paraphrases.

H2: If queries q_1 and q_2 hit the same title t, q_1 and q_2 are likely to be paraphrases.

H3: If a query q hits titles t_1 and t_2, then t_1 and t_2 are likely to be paraphrases.

Table 1: Hypotheses for extracting paraphrases.

expansion terms for new queries. Note that the expansion terms are merely related terms of the queries, not necessarily paraphrases.

There are other studies that use query logs for constructing ontologies (Sekine and Suzuki, 2007), learning named entities (Paşca, 2007), building user profiles (Richardson, 2008), correcting spelling errors (Ahmad and Kondrak, 2005), and so forth.

3 The Proposed Method

3.1 Basic Idea

Nowadays, more and more users tend to search long queries with search engines. Many users even directly search questions to get exact answers. By analyzing our query log that records rich information including user queries, clicked urls, titles, etc., we find that most titles of clicked documents are highly related with search queries. Especially, paraphrases can be easily found from long queries and the corresponding clicked titles. This motivates us to extract paraphrases from query-title pairs. Here we introduce a concept *hit* that will be frequently used: given a query q, a web document d, and d's title t, if there exist some users that click on d when searching q, then we say *q hits t*.

The hypothesis for extracting paraphrase Q-T is shown in Table 1 (H1). In addition, we find that when several queries hit the same title, the queries are likely to be paraphrases of each other. The other way round, when a query hits several titles, paraphrases can also be found among the titles. We therefore further extract paraphrase Q-Q and T-T from the paraphrase Q-T. The underlying hypotheses can be found in Table 1 (H2 and H3). Note that, based on H2 and H3, paraphrase Q-Q and T-T can be directly extracted from raw Q-T pairs. However, in consideration of precision, we extract them from paraphrase Q-T. We call our paraphrase Q-Q and T-T extraction approach as a pivot approach, since we use titles as *pivots* (queries as *targets*) when extracting paraphrase Q-Q and use queries as *pivots* (titles as *targets*) when extracting paraphrase T-T.

```
INPUT: $\mathcal{Q}$: query space, $\mathcal{T}$: title space
OUTPUT: $P_{qt}$: the set of paraphrase Q-T,
        $P_{qq}$: the set of paraphrase Q-Q,
        $P_{tt}$: the set of paraphrase T-T,
        $ParaSet$: the set of paraphrases

 1. FOR any $q \in \mathcal{Q}$ and $t \in \mathcal{T}$
 2.     IF $q$ hits $t$
 3.         IF $IsParaphrase(q, t)$
 4.             Add $\langle q, t \rangle$ to $P_{qt}$
 5.         END IF
 6.     END IF
 7. END FOR

 8. FOR any $q_1, q_2 \in \mathcal{Q}$ and $t \in \mathcal{T}$
 9.     IF $\langle q_1, t \rangle \in P_{qt}$ and $\langle q_2, t \rangle \in P_{qt}$
10.         IF $IsParaphrase(q_1, q_2)$
11.             Add $\langle q_1, q_2 \rangle$ to $P_{qq}$
12.         END IF
13.     END IF
14. END FOR

15. FOR any $t_1, t_2 \in \mathcal{T}$ and $q \in \mathcal{Q}$
16.     IF $\langle q, t_1 \rangle \in P_{qt}$ and $\langle q, t_2 \rangle \in P_{qt}$
17.         IF $IsParaphrase(t_1, t_2)$
18.             Add $\langle t_1, t_2 \rangle$ to $P_{tt}$
19.         END IF
20.     END IF
21. END FOR

22. RETURN $ParaSet = P_{qt} \cup P_{qq} \cup P_{tt}$
```

Table 2: Algorithm for extracting paraphrases.

3.2 Algorithm

Our paraphrase extraction algorithm is shown in Table 2. In particular, lines 1$\sim$7 extract paraphrase Q-T from the query log. Lines 8∼14 and 15∼21 extract paraphrase Q-Q and T-T, respectively. Line 22 combines the paraphrase Q-T, Q-Q, and T-T together. To filter noise, the extracted Q-T, Q-Q, and T-T pairs are all validated using a function $IsParaphrase(s_1, s_2)$. In this work, we recast paraphrase validation as a binary classification problem. Any pair of $\langle s_1, s_2\rangle$ is classified as 1 (paraphrase) or 0 (non-paraphrase) with a support vector machine (SVM) classifier. The features used for classification will be detailed in Section 3.3.

In practice, we exploit a query log that contains 287 million Q-T pairs, which are then filtered using the following constraints: (1) exclude Q-T pairs that are too short, i.e., either query q or tittle t contains less than three terms; (2) exclude Q-T pairs where q subsumes t or vice versa, e.g., "牛肉 (beef)" and "牛肉 的 做法 (cooking method of beef)"; (3) exclude Q-T pairs in which the similarity between q and t is below a predefined threshold T[2]; (4) exclude Q-T pairs whose t contains frequent internet terms, such as "主页 (home page)", "网站 (web site)", "在线 (online)", since such titles are mostly organization home pages, online videos, downloadable resources, etc., which are useless for our purpose of paraphrase extraction.

3.3 Features for Paraphrase Validation

Given a pair of candidate paraphrases $\langle s_1, s_2\rangle$, in which s_1 and s_2 can be either a query or a title, we exploit the following features in the classification-based paraphrase validation.

• **Frequency Feature** F_F. F_F is defined based on each $\langle s_1, s_2\rangle$'s frequency. We expect that more frequent $\langle s_1, s_2\rangle$ should be more reliable.

$$F_F(s_1, s_2) = \{ \begin{array}{ll} \frac{c(s_1,s_2)}{C} & if \;\; c(s_1, s_2) < C \\ 1 & if \;\; c(s_1, s_2) \geq C \end{array} \quad (1)$$

where $c(s_1, s_2)$ denotes the number of times that the $\langle s_1, s_2\rangle$ pair occurs in the corpus. C is a normalizing factor ($C = 10$ in our experiments).

• **Length Rate Feature** F_{LR}:

$$F_{LR}(s_1, s_2) = \frac{min\{c_w(s_1), c_w(s_2)\}}{max\{c_w(s_1), c_w(s_2)\}} \quad (2)$$

where $c_w(s)$ denotes the number of words in s.

• **Word Overlap Rate Feature** F_{WOR}:

$$F_{WOR}(s_1, s_2) = \frac{c_w(s_1 \cap s_2)}{max\{c_w(s_1), c_w(s_2)\}} \quad (3)$$

where "$s_1 \cap s_2$" is the intersection of s_1 and s_2.

• **Character Overlap Rate Feature** F_{COR}. Chinese words are composed of characters. It is quite often that words with similar characters share similar meanings, such as "爽快 (comfortable)" and "痛快 (comfortable)", "出售 (sell)" and "销售 (sell)". Here we use F_{COR} to measure the similarity between s_1 and s_2 at the character level. Detailedly, we segment s_1 and s_2 into sets of characters and compute the overlap rate based on Equation (3)[3].

• **Cosine Similarity Feature** F_{CS}. In F_{CS}, both s_1 and s_2 are represented as vectors and their cosine similarity is computed as:

$$F_{CS}(s_1, s_2) = \frac{vec_w(s_1) \cdot vec_w(s_2)}{\|vec_w(s_1)\| \times \|vec_w(s_2)\|} \quad (4)$$

where $vec_w(s)$ is the vector of words in s, "$\cdot$" denotes the dot product of two vectors, $\|vec_w(s)\|$ is the norm of a vector. Here, the weight of each word w in a vector is computed using a heuristic similar to tf-idf:

$$W(w) = tf(w) \times log(\frac{N}{c(w)} + 0.1) \quad (5)$$

where $tf(w)$ is the frequency of w in the given s, $c(w)$ is the number of times that w occurs in the corpus, $N = \max_w c(w)$.

• **Edit Distance Feature** F_{ED}. Let $ED(s_1, s_2)$ be the edit distance at the word level between s_1 and s_2, we compute F_{ED} as follows:

$$F_{ED}(s_1, s_2) = 1 - \frac{ED(s_1, s_2)}{max\{c_w(s_1), c_w(s_2)\}} \quad (6)$$

[2] The similarity is computed based on word overlap rate, which will be described in detail in section 3.3. We set $T = 0.6$ in the experiments.

[3] In F_{COR}, $c_w(s)$ of Equation (3) denotes the number of characters in s.

• **Named Entity (NE) Similarity Feature** F_{NE}. NE information is critical in paraphrase identification (Shinyama et al., 2002). We therefore compute the NE similarity between s_1 and s_2 and take it as a feature. We employ a Chinese NE recognition tool that can recognize *person names*, *locations*, *organizations*, and *numerals*. The NE similarity is computed as:

$$F_{NE}(s_1, s_2) = \frac{c_{ne}(s_1 \cap s_2) + 1}{max\{c_{ne}(s_1), c_{ne}(s_2)\} + 1} \quad (7)$$

where $c_{ne}(s)$ denotes the number of NEs in s. Equation (7) guarantees $F_{NE} = 1$ if there are no NEs in either s_1 or s_2.

• **Pivot Fertility Feature** F_{PF}**:** F_{PF} is a feature specially designed for paraphrase Q-Q and T-T extraction, which are based on the pivot approach[4]. Specifically, we define *fertility* of a pivot as the number of targets it corresponds to. Our observation indicates that the larger the fertility of a pivot is, the more noisy the targets are. Hence we define F_{PF} as:

$$F_{PF}(s_1, s_2) = \max_p \frac{1}{f(p)} \quad (8)$$

where $s_1 = q_1$, $s_2 = q_2$, $p = t$ when classifying Q-Q, while $s_1 = t_1$, $s_2 = t_2$, $p = q$ when classifying T-T. $f(p)$ denotes the fertility of the pivot p. The value is maximized over p if s_1 and s_2 can be extracted with multiple pivots.

3.4 Generating Paraphrase Patterns

A key feature of our method is that the extracted paraphrases are particularly suitable for generating paraphrase patterns, especially for the hot domains that are frequently searched. For example, there are quite a few paraphrases concerning the therapy of various diseases, from which we can easily induce patterns expressing the meaning of "How to treat [X] disease", such as "[X] 病 如何 治疗", "怎么 治疗 [X] 病", and "[X] 病 的 治疗 方法". Therefore, in this work, we try to generate paraphrase patterns using the extracted paraphrases.

In our preliminary experiments, we only induce paraphrase patterns from paraphrases that contain no more than 6 words. In addition, only one slot is allowed in each pair of paraphrase patterns. Let s_1 and s_2 be a pair of paraphrases extracted above. If there exist words $w \in s_1$ and $v \in s_2$ that satisfy (1) $w = v$, (2) w and v are not stop words, then we can induce a pair of paraphrase patterns by replacing w in s_1 and v in s_2 with a slot "[X]". It is obvious that several pairs of paraphrase patterns may be induced from one pair of paraphrases.

	SAME	**RELA**	**DIFF**
percent (%)	55.92	44.08	-

Table 3: Human labeling of candidate Q-T.

4 Experiments

We experiment with a query log that contains a total of 284,316,659 queries. Statistics reveal that 170,315,807 queries (59.90%) lead to at least one user click, each having 1.69 clicks on average. We extract 287,129,850 raw Q-T pairs using the query log, from which 4,448,347 pairs of candidate Q-T are left after filtering as described in Section 3.2. Almost all queries and titles are written in Chinese, though some of them contain English or Japanese words. The preprocessing of candidate Q-T includes Chinese word segmentation (WSeg) and NE recognition (NER). Our WSeg tool is implemented based on forward maximum matching, while the NER tool is based on a NE dictionary mined from the web.

4.1 Evaluation of Candidate Q-T

We first evaluate candidate Q-T without validation. To this end, we randomly sampled 5000 pairs of candidate Q-T and labeled them manually. Each pair is labeled into one of the 3 classes: SAME - q and t have the same meaning; RELA - q and t have related meanings; DIFF - q and t have clearly different meanings. The labeling results are listed in Table 3. We can see that no candidate Q-T is in the DIFF class. This is not surprising, since users are unlikely to click on web pages unrelated to their queries.

To gain a better insight into the data, we analyzed the subtle types of candidate Q-T in both SAME and RELA classes. In detail, we sampled

[4] F_{PF} is not used in paraphrase Q-T validation.

1000 pairs of candidate Q-T from the 5000 pairs labeled above, in which 563 are in the SAME class, while the other 437 are in the RELA class. Our analysis suggests that candidate Q-T in the SAME class can be divided into 4 subtle types:

- Trivial change (12.61%): changes of punctuation or stop words, such as “考研 失败 怎么 办” and “考研 失败 怎么 办？”.
- Word or phrase replacement (68.38%): replacements of synonymous words or phrases, such as “咖啡 斑 的 治疗 多少 钱 (how mach is ...)” and “咖啡 斑 的 治疗 费用 是 多少 (what is the price of ...)”.
- Structure change (7.10%): changes of both words and word orders, such as “减肥 中 水果 可以 吃 什么 (what fruit can I eat on a diet)” and “吃 什么 水果 可以 瘦身 (what fruit can help loss weight)”.
- Others (11.90%): candidate Q-T that cannot be classified into the 3 types above.

The above analysis reveals that more than two thirds of candidate Q-T in the SAME class are in the “word or phrase replacement” type, while the ones with structure changes are slightly more than 7%. We believe this is mainly because queries and titles are relatively short and their structures are simple. Thus structure rewriting can hardly be conducted. This distribution is in line with that reported in (Zhao et al., 2008).

As for the RELA class, we find that 42.33% of such candidate Q-T share a problem of named entity mismatch, such as “美国 (US) 大型 水利 工程” and “中国 (China) 急需 大型 水利 工程”. This indicates that the NE similarity feature is necessary in paraphrase validation.

4.2 Evaluation of Paraphrase Q-T

The candidate Q-T extracted above are classified with a SVM classifier[5] under its default setting. To evaluate the classifier, we run 5-fold cross validation with the 5000 human annotated data, in which we use 4000 for training and the rest 1000 for testing in each run. The evaluation criteria are precision (P), recall (R), and f-measure (F), which are defined as follows:

$$P = \frac{\|S_a \cap S_m\|}{\|S_a\|} \tag{9}$$

$$R = \frac{\|S_a \cap S_m\|}{\|S_m\|} \tag{10}$$

$$F = \frac{2 \times P \times R}{P + R} \tag{11}$$

where S_a is the set of paraphrases automatically recognized with the classifier, S_m is the set of paraphrases manually annotated. Precision, recall, and f-measure are averaged over 5 runs in the 5-fold cross validation.

Figure 2 (a) shows the classification results (dark bars). For comparison, we also show the precision, recall[6], and f-measure of the candidate Q-T (light bars). As can be seen, the precision is improved from 0.5592 to 0.7444 after classification. F-measure is also evidently enhanced. This result indicates that the classification-based paraphrase validation is effective. We then use all of the 5000 annotated data to train a classifier and classify all the candidate Q-T. Results show that 2,762,291 out of 4,448,347 pairs of candidate Q-T are classified as paraphrases.

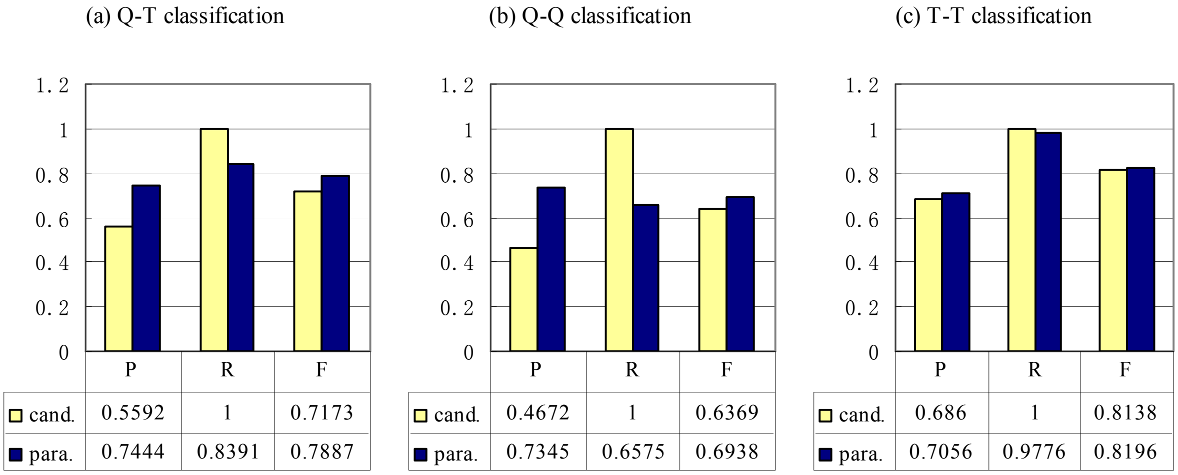
4.3 Evaluation of Paraphrase Q-Q and T-T

From the paraphrase Q-T, we further extracted 934,758 pairs of candidate Q-Q and 438,954 pairs of candidate T-T (without validation). We randomly sampled 5000 from each for human annotation. The results show that the precisions of candidate Q-Q and T-T are 0.4672 and 0.6860, respectively. As can be seen, the precision of candidate Q-Q is much lower than that of candidate T-T. Our analysis reveals that it is mainly because candidate Q-Q are more noisy, since user queries contain quite a lot of spelling mistakes and informal expressions.

The candidate Q-Q and T-T are also refined based on classification. We first evaluate the classification performance using the 5000 human labeled data. The experimental setups for Q-Q and

[5] We use libsvm-2.82 toolkit, which can be downloaded from http://www.csie.ntu.edu.tw/ cjlin/libsvm/

[6] We assume all possible paraphrases are included in the candidates, thus its recall is 100%.



(a)	P	R	F
□ cand.	0.5592	1	0.7173
■ para.	0.7444	0.8391	0.7887

(b)	P	R	F
□ cand.	0.4672	1	0.6369
■ para.	0.7345	0.6575	0.6938

(c)	P	R	F
□ cand.	0.686	1	0.8138
■ para.	0.7056	0.9776	0.8196

Figure 2: Classification precision (P), recall (R), and f-measure (F).

T-T classification are the same as that of Q-T classification, in which we run 5-fold cross validation with a SVM classifier using its default parameters. Figure 2 (b) and (c) give the classification results (dark bars) as well as the precision, recall, and f-measure of the candidates (light bars).

We can see that the precision of Q-Q is significantly enhanced from 0.4672 to 0.7345 after classification, which suggests that a substantial part of errors and noise are removed. The increase of f-measure demonstrates the effectiveness of classification despite the decrease of recall. Meanwhile, the quality of candidate T-T is not clearly improved after classification. The reason should be that the precision of candidate T-T is already pretty high. We then use all 5000 human labeled data to train a classifier for Q-Q and T-T respectively and classify all candidate Q-Q and T-T. Results show that 390,920 pairs of paraphrase Q-Q and 415,539 pairs of paraphrase T-T are extracted after classification.

4.4 Evaluation of Paraphrase Patterns

Using the method introduced in Section 3.4, we have generated 73,484 pairs of paraphrase patterns that appear at least two times in the corpus. We randomly selected 500 pairs and labeled them manually. The results show that the precision is 78.4%. Two examples are shown in Table 4, in which p_1 and p_2 are paraphrase patterns. Some slot fillers are also listed below. We realize that the method currently used for inducing paraphrase patterns is simple. Hence we will improve the method in our following experiments. Specifically, multiple slots will be allowed in a pair of patterns. In addition, we will try to apply the alignment techniques in the generation of paraphrase patterns, as Zhao et al. (2008) did.

p_1	[X] 文件 怎么 打开
p_2	如何 打开 [X] 文件
	(how to open [X] file)
slot	7z; ashx; aspx; bib; cda; cdfs; cmp; cpi; csf; csv; cur; dat; dek...
p_1	关于 [X] 的 诗词
p_2	有关 [X] 的 诗歌
	(poems about [X])
slot	草原 (prairies); 长江 (Yangtze River); 泰山 (Mount Tai); 乡愁 (nostalgia)...

Table 4: Examples of paraphrase patterns.

4.5 Analysis

Feature Contribution. To investigate the contributions of different features used in classification, we tried different feature combinations for each of our three classifiers. The results are shown in Table 5, in which "+" means the feature has contribution to the corresponding classifier. As can be seen, the character overlap rate feature (F_{COR}), cosine similarity feature (F_{CS}), and NE similarity

Feature	Q-T	Q-Q	T-T
F_F	+		
F_{LR}		+	
F_{WOR}			
F_{COR}	+	+	+
F_{CS}	+	+	+
F_{ED}		+	
F_{NE}	+	+	+
F_{PF}		+	

Table 5: Feature contribution.

feature (F_{NE}) are the most useful, which play important roles in all the three classifiers. The other features are useful in some of the classifiers except the word overlap rate feature (F_{WOR}). The classification results reported in prior sections are all achieved with the optimal feature combination.

Analysis of the Paraphrases. We combine the extracted paraphrase Q-T, Q-Q and T-T and get a total of 3,560,257 pairs of unique paraphrases. Statistics show that only 8380 pairs (0.24%) are from more than one source, which indicates that the intersection among the three sets is very small. Further statistics show that the average length of the queries and titles in the paraphrases is 6.69 (words).

To have a detailed analysis of the extracted paraphrases, we randomly selected 1000 pairs and manually labeled the precision, types, and domains. It is found that more than 43% of the paraphrases are paraphrase questions, in which *how* (36%), *what* (19%), and *yes/no* (14%) questions are the most common. In addition, we find that the precision of paraphrase questions (84.26%) is evidently higher than non-question paraphrases (65.14%). Those paraphrase questions are useful in question analysis and expansion in QA, which can hardly be extracted from other kinds of corpora.

As expected, the paraphrases we extract cover a variety of domains. However, around 50% of them are in the 7 most popular domains[7], including: (1) health and medicine, (2) documentary download, (3) entertainment, (4) software, (5) education and study, (6) computer game, (7) economy and finance. This analysis reflects what web users are most concerned about. These domains, especially (4) and (6), are not well covered by the parallel and comparable corpora previously used for paraphrase extraction.

5 Conclusions and Future Directions

In this paper, we put forward a novel method that extracts paraphrases from search engine query logs. Our contribution is that we, for the first time, propose to extract paraphrases from user queries and the corresponding clicked document titles. Specifically, three kinds of paraphrases are extracted, which can be (1) a query and a hit title, (2) two queries that hit the same title, and (3) two titles hit by the same query. The extracted paraphrases are refined based on classification. Using the proposed method, we extracted over 3.5 million pairs of paraphrases from a query log of Baidu. Human evaluation results show that the precision of the paraphrases is above 70%. The results also show that we can generate high-quality paraphrase patterns from the extracted paraphrases.

Our future research will be conducted along the following directions. Firstly, we will use a much larger query log for paraphrase extraction, so as to enhance the coverage of paraphrases. Secondly, we plan to have a deeper study of the transitivity of paraphrasing. Simply speaking, we want to find out whether we can extract $\langle s_1, s_3 \rangle$ as paraphrases given that $\langle s_1, s_2 \rangle$ and $\langle s_2, s_3 \rangle$ are paraphrases.

6 Acknowledgments

We would like to thank Wanxiang Che, Hua Wu, and the anonymous reviewers for their useful comments on this paper.

References

Farooq Ahmad and Grzegorz Kondrak. 2005. Learning a Spelling Error Model from Search Query Logs. In *Proceedings of HLT/EMNLP*, pages 955-962.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of ACL*, pages 597-604.

[7] Note that pornographic queries have been filtered from the query log beforehand.

Regina Barzilay and Lillian Lee. 2003. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In *Proceedings of HLT-NAACL*, pages 16-23.

Regina Barzilay and Kathleen R. McKeown. 2001. Extracting Paraphrases from a Parallel Corpus. In *Proceedings of ACL/EACL*, pages 50-57.

Rahul Bhagat and Deepak Ravichandran. 2008. Large Scale Acquisition of Paraphrases for Learning Surface Patterns. In *Proceedings of ACL-08: HLT*, pages 674-682.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved Statistical Machine Translation Using Paraphrases. In *Proceedings of HLT-NAACL*, pages 17-24.

Chris Callison-Burch. 2008. Syntactic Constraints on Paraphrases Extracted from Parallel Corpora. In *Proceedings of EMNLP*, pages 196-205.

Hang Cui, Ji-Rong Wen, Jian-Yun Nie, Wei-Ying Ma. 2002. Probabilistic Query Expansion Using Query Logs In *Proceedings of WWW*, pages 325-332.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of COLING*, pages 350-356.

Pablo Ariel Duboue and Jennifer Chu-Carroll. 2006. Answering the Question You Wish They Had Asked: The Impact of Paraphrasing for Question Answering. In *Proceedings of HLT-NAACL*, pages 33-36.

Wei Gao, Cheng Niu, Jian-Yun Nie, Ming Zhou, Jian Hu, Kam-Fai Wong, and Hsiao-Wuen Hon. 2007. Cross-Lingual Query Suggestion Using Query Logs of Different Languages. In *Proceedings of SIGIR*, pages 463-470.

Ali Ibrahim, Boris Katz, Jimmy Lin. 2003. Extracting Structural Paraphrases from Aligned Monolingual Corpora. In *Proceedings of IWP*, pages 57-64.

Lidija Iordanskaja, Richard Kittredge, and Alain Polguère. 1991. Lexical Selection and Paraphrase in a Meaning-Text Generation Model. In Cécile L. Paris, William R. Swartout, and William C. Mann (Eds.): Natural Language Generation in Artificial Intelligence and Computational Linguistics, pages 293-312.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for Automatic Evaluation. In *Proceedings of HLT-NAACL*, pages 455-462.

De-Kang Lin and Patrick Pantel. 2001. Discovery of Inference Rules for Question Answering. In *Natural Language Engineering* 7(4): 343-360.

Marius Paşca and Péter Dienes. 2005. Aligning Needles in a Haystack: Paraphrase Acquisition Across the Web. In *Proceedings of IJCNLP*, pages 119-130.

Marius Paşca. 2007. Weakly-supervised Discovery of Named Entities using Web Search Queries. In *Proceedings of CIKM*, pages 683-690.

Deepak Ravichandran and Eduard Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proceedings of ACL*, pages 41-47.

Matthew Richardson. 2008. Learning about the World through Long-Term Query Logs. In *ACM Transactions on the Web* 2(4): 1-27.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal and Yi Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In *Proceedings of ACL*, pages 464-471.

Satoshi Sekine and Hisami Suzuki. 2007. Acquiring Ontological Knowledge from Query Logs. In *Proceedings of WWW*, pages 1223-1224.

Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. 2002. Automatic Paraphrase Acquisition from News Articles. In *Proceedings of HLT*, pages 40-46.

Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. 2002. Query Clustering Using User Logs. In *ACM Transactions on Information Systems* 20(1): 59-81, 2002.

Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2008. Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora. In *Proceedings of ACL-08:HLT*, pages 780-788.

Shiqi Zhao, Ming Zhou, and Ting Liu. 2007. Learning Question Paraphrases for QA from Encarta Logs. In *Proceedings of IJCAI*, pages 1795-1800.

Leveraging Multiple MT Engines for Paraphrase Generation

Shiqi Zhao[†‡]**, Haifeng Wang**[†]**, Xiang Lan**[‡]**, and Ting Liu**[‡]
[†]Baidu Inc.
[‡]HIT Center for Information Retrieval, Harbin Institute of Technology
{zhaoshiqi, wanghaifeng}@baidu.com,
{xlan, tliu}@ir.hit.edu.cn

Abstract

This paper proposes a method that leverages multiple machine translation (MT) engines for paraphrase generation (PG). The method includes two stages. Firstly, we use a multi-pivot approach to acquire a set of candidate paraphrases for a source sentence S. Then, we employ two kinds of techniques, namely the selection-based technique and the decoding-based technique, to produce a best paraphrase T for S using the candidates acquired in the first stage. Experimental results show that: (1) The multi-pivot approach is effective for obtaining plenty of valuable candidate paraphrases. (2) Both the selection-based and decoding-based techniques can make good use of the candidates and produce high-quality paraphrases. Moreover, these two techniques are complementary. (3) The proposed method outperforms a state-of-the-art paraphrase generation approach.

1 Introduction

This paper addresses the problem of paraphrase generation (PG), which seeks to generate paraphrases for sentences. PG is important in many natural language processing (NLP) applications. For example, in machine translation (MT), a sentence can be paraphrased so as to make it more translatable (Zhang and Yamamoto, 2002; Callison-Burch et al., 2006). In question answering (QA), a question can be paraphrased to improve the coverage of answer extraction (Duboue and Chu-Carroll, 2006; Riezler et al., 2007). In natural language generation (NLG), paraphrasing can help to increase the expressive power of the NLG systems (Iordanskaja et al., 1991).

In this paper, we propose a novel PG method. For an English sentence S, the method first acquires a set of candidate paraphrases with a multi-pivot approach, which uses MT engines to automatically translate S into multiple pivot languages and then translate them back into English. Furthermore, the method employs two kinds of techniques to produce a best paraphrase T for S using the candidates, i.e., the selection-based and decoding-based techniques. The former selects a best paraphrase from the candidates based on Minimum Bayes Risk (MBR), while the latter trains a MT model using the candidates and generates paraphrases with a MT decoder.

We evaluate our method on a set of 1182 English sentences. The results show that: (1) although the candidate paraphrases acquired by MT engines are noisy, they provide good raw materials for further paraphrase generation; (2) the selection-based technique is effective, which results in the best performance; (3) the decoding-based technique is promising, which can generate paraphrases that are different from the candidates; (4) both the selection-based and decoding-based techniques outperform a state-of-the-art approach SPG (Zhao et al., 2009).

2 Related Work

2.1 Methods for Paraphrase Generation

MT-based method is the mainstream method on PG. It regards PG as a monolingual machine translation problem, i.e., "translating" a sentence S into another sentence T in the same language.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1326–1334,
Beijing, August 2010

Quirk et al. (2004) first presented the MT-based method. They trained a statistical MT (SMT) model on a monolingual parallel corpus extracted from comparable news articles and applied the model to generate paraphrases. Their work shows that SMT techniques can be extended to PG. However, its usefulness is limited by the scarcity of monolingual parallel data.

To overcome the data sparseness problem, Zhao et al. (2008a) improved the MT-based PG method by training the paraphrase model using multiple resources, including monolingual parallel corpora, monolingual comparable corpora, bilingual parallel corpora, etc. Their results show that bilingual parallel corpora are the most useful among the exploited resources. Zhao et al. (2009) further improved the method by introducing a *usability sub-model* into the paraphrase model so as to generate varied paraphrases for different applications.

The main disadvantage of the MT-based method is that its performance heavily depends on the fine-grained paraphrases, such as paraphrase phrases and patterns, which provide paraphrase options in decoding. Hence one has to first extract fine-grained paraphrases from various corpora with different methods (Zhao et al., 2008a; Zhao et al., 2009), which is difficult and time-consuming.

In addition to the MT-based method, researchers have also investigated other methods for paraphrase generation, such as the pattern-based methods (Barzilay and Lee, 2003; Pang et al., 2003), thesaurus-based methods (Bolshakov and Gelbukh, 2004; Kauchak and Barzilay, 2006), and NLG-based methods (Kozlowski et al., 2003; Power and Scott, 2005).

2.2 Pivot Approach for Paraphrasing

Bannard and Callison-Burch (2005) introduced the pivot approach to extracting paraphrase phrases from bilingual parallel corpora. Their basic assumption is that two English phrases aligned with the same phrase in a foreign language (also called a pivot language) are potential paraphrases. Zhao et al. (2008b) extended the approach and used it to extract paraphrase patterns. Both of the above works have proved the effectiveness of the pivot approach in paraphrase extraction.

Pivot approach can also be used in paraphrase generation. It generates paraphrases by translating sentences from a source language to one (single-pivot) or more (multi-pivot) pivot languages and then translating them back to the source language. Duboue et al. (2006) first proposed the multi-pivot approach for paraphrase generation, which was specially designed for question expansion in QA. In addition, Max (2009) presented a single-pivot approach for generating sub-sentential paraphrases. A clear difference between our method and the above works is that we propose selection-based and decoding-based techniques to generate high-quality paraphrases using the candidates yielded from the pivot approach.

3 Multi-pivot Approach for Acquiring Candidate Paraphrases

A single-pivot PG approach paraphrases a sentence S by translating it into a pivot language PL with a MT engine MT_1 and then translating it back into the source language with MT_2. In this paper, a single-pivot PG system is represented as a triple (MT_1, PL, MT_2). A multi-pivot PG system is made up of a set of single-pivot systems with various pivot languages and MT engines. Given m pivot languages and n MT engines, we can build a multi-pivot PG system consisting of N ($N \leq n * m * n$) single-pivot ones, where $N = n * m * n$ iff all the n MT engines can perform bidirectional translation between the source and each pivot language.

In this work, we experiment with 6 pivot languages (Table 1) and 3 MT engines (Table 2) in the multi-pivot approach. All the 3 MT engines are off-the-shelf systems, in which Google and Microsoft translators are SMT engines, while Systran translator is a rule-based MT engine. Each MT engine can translate English to all the 6 pivot languages and back to English. We thereby construct a multi-pivot PG system consisting of 54 (3*6*3) single-pivot systems.

The advantages of the multi-pivot PG approach lie in two aspects. First, it effectively makes use of the vast bilingual data and translation rules underlying the MT engines. Second, the approach is simple, which just sends sentences to the online MT engines and gets the translations back.

Source Sentence	he said there will be major cuts in the salaries of high-level civil servants .
(GG, G, MS)	he said there **are significant** cuts in the salaries of high-level **officials** .
(GG, F, GG)	he said there will be **significant** cuts in the salaries of *top civil level* .
(MS, C, MS)	he said **that** there will be *a major senior civil service pay cut* .
(MS, F, ST)	he said there will be **great** cuts in the **wages** of the *high level civils servant* .
(ST, G, GG)	he said **that** there **are** major cuts in the salaries of **senior government officials** .

Table 3: Examples of candidate paraphrases obtained using the multi-pivot approach.

1	French (**F**)	4	Italian (**I**)
2	German (**G**)	5	Portuguese (**P**)
3	Spanish (**S**)	6	Chinese (**C**)

Table 1: Pivot languages used in the approach.

1	Google Translate (**GG**) (translate.google.com)
2	Microsoft Translator (**MS**) (www.microsofttranslator.com)
3	Systran Online Translation (**ST**) (www.systransoft.com)

Table 2: MT engines utilized in the approach.

4 Producing High-quality Paraphrases using the Candidates

Table 3 shows some examples of candidate paraphrases for a sentence. As can be seen, the candidates do provide some correct and useful paraphrase substitutes (in bold) for the source sentence. However, they also contain quite a few errors (in italic) due to the limited translation quality of the MT engines. The problem is even worse when the source sentences get longer and more complicated. Therefore, we need to combine the outputs of the multiple single-pivot PG systems and produce high-quality paraphrases out of them. To this end, we investigate two techniques, namely, the selection-based and decoding-based techniques.

4.1 Selection-based Technique

Given a source sentence S along with a set D of candidate paraphrases $\{T_1, T_2, ..., T_i, ...T_N\}$, the goal of the selection-based technique is to select the best paraphrase $\hat{T_i}$ for S from D. The paraphrase selection technique we propose is based on Minimum Bayes Risk (MBR). In detail, the MBR based technique first measures the quality of each candidate paraphrase $T_i \in D$ in terms of Bayes risk (BR), and then selects the one with the minimum BR as the best paraphrase. In detail, given S, a candidate $T_i \in D$, a reference paraphrase T[1], and a loss function $L(T, T_i)$ that measures the quality of T_i relative to T, we define the Bayes risk as follows:

$$BR(T_i) = E_{P(T,S)}[L(T, T_i)], \qquad (1)$$

where the expectation is taken under the true distribution $P(T, S)$ of the paraphrases. According to (Bickel and Doksum, 1977), the candidate paraphrase that minimizes the Bayes risk can be found as follows:

$$\hat{T_i} = \arg\min_{T_i \in D} \sum_{T \in \mathcal{T}} L(T, T_i) P(T|S), \qquad (2)$$

where $\mathcal{T}$ represents the space of reference paraphrases. In practice, however, the collection of reference paraphrases is not available. We thus construct a set $D' = D \bigcup \{S\}$ to approximate $\mathcal{T}$[2]. In addition, we cannot estimate $P(T|S)$ in Equation (2), either. Therefore, we make a simplification by assigning a constant c to $P(T|S)$ for each $T \in D'$, which can then be removed:

$$\hat{T_i} = \arg\min_{T_i \in D} \sum_{T \in D'} L(T, T_i). \qquad (3)$$

Equation (3) can be further rewritten using a gain function $G(T, T_i)$ instead of the loss function:

[1] Here we assume that we have the collection of all possible paraphrases of S, which are used as references.

[2] The source sentence S is included in D' based on the consideration that a sentence is allowed to keep unchanged during paraphrasing.

$$\hat{T}_i = \arg\max_{T_i \in D} \sum_{T \in D'} G(T, T_i). \qquad (4)$$

We define the gain function based on BLEU: $G(T, T_i) = BLEU(T, T_i)$. BLEU is a widely used metric in the automatic evaluation of MT (Papineni et al., 2002). It measures the similarity of two sentences by counting the overlapping n-grams (n=1,2,3,4 in our experiments):

$$BLEU(T, T_i) = BP \cdot \exp(\sum_{n=1}^{4} w_n \log p_n(T, T_i)),$$

where $p_n(T, T_i)$ is the n-gram precision of T_i and $w_n = 1/4$. BP (≤ 1) is a brevity penalty that penalizes T_i if it is shorter than T.

In summary, for each sentence S, the MBR based technique selects a paraphrase that is the most similar to all candidates and the source sentence. The underlying assumption is that correct paraphrase substitutes should be common among the candidates, while errors committed by the single-pivot PG systems should be all different. We denote this approach as **S-1** hereafter.

Approaches for comparison. In the experiments, we also design another two paraphrase selection approaches S-2 and S-3 for comparison with S-1.

S-2: S-2 selects the best single-pivot PG system from all the 54 ones. The selection is also based on MBR and BLEU. For each single-pivot PG system, we sum up its gain function values over a set of source sentences (i.e., $\sum_S \sum_{T_S \in D'_S} G(T_S, T_{Si})$). Then we select the one with the maximum gain value as the best single-pivot system. In our experiments, the selected best single-pivot PG system is (ST, P, GG), the candidate paraphrases acquired by which are then returned as the best paraphrases in S-2.

S-3: S-3 is a simple baseline, which just randomly selects a paraphrase from the 54 candidates for each source sentence S.

4.2 Decoding-based Technique

The selection-based technique introduced above has an inherent limitation that it can only select a paraphrase from the candidates. That is to say, it can never produce a perfect paraphrase if all the candidates have some tiny flaws. To solve this problem, we propose the decoding-based technique, which trains a MT model using the candidate paraphrases of each source sentence S and generates a new paraphrase T for S with a MT decoder.

major cuts	high-level civil servants
significant cuts	senior officials
major cuts*	high-level officials
important cuts	senior civil servants
big cuts	
great cuts	

Table 4: Extracted phrase pairs. (*This is called a *self-paraphrase* of the source phrase, which is generated when a phrase keeps unchanged in some of the candidate paraphrases.)

In this work, we implement the decoding-based technique using Giza++ (Och and Ney, 2000) and Moses (Hoang and Koehn, 2008), both of which are commonly used SMT tools. For a sentence S, we first construct a set of parallel sentences by pairing S with each of its candidate paraphrases: $\{(S,T_1),(S,T_2),...,(S,T_N)\}$ ($N = 54$). We then run word alignment on the set using Giza++ and extract aligned phrase pairs as described in (Koehn, 2004). Here we only keep the phrase pairs that are aligned $\geq$3 times on the set, so as to filter errors brought by the noisy sentence pairs. The extracted phrase pairs are stored in a phrase table. Table 4 shows some extracted phrase pairs.

Note that Giza++ is sensitive to the data size. Hence it is interesting to examine if the alignment can be improved by augmenting the parallel sentence pairs. To this end, we have tried augmenting the parallel set for each sentence S by pairing any two candidate paraphrases. In this manner, C_N^2 sentence pairs are augmented for each S. We conduct word alignment using the $(N + C_N^2)$ sentence pairs and extract aligned phrases from the original N pairs. However, we have not found clear improvement after observing the results. Therefore, we do not adopt the augmentation strategy in our experiments.

Using the extracted phrasal paraphrases, we conduct decoding for the sentence S with Moses, which is based on a log-linear model. The default setting of Moses is used, except that the distortion model for phrase reordering is turned off[3]. The language model in Moses is trained using a 9 GB English corpus. We denote the above approach as **D-1** in what follows.

Approach for comparison. The main advantage of the decoding-based technique is that it allows us to customize the paraphrases for different requirements through tailoring the phrase table or tuning the model parameters. As a case study, this paper shows how to generate paraphrases with varied *paraphrase rates*[4].

D-2: The extracted phrasal paraphrases (including self-paraphrases) are stored in a phrase table, in which each phrase pair has 4 scores measuring their alignment confidence (Koehn et al., 2003). Our basic idea is to control the paraphrase rate by tuning the scores of the self-paraphrases. We thus extend D-1 to D-2, which assigns a weight λ ($\lambda > 0$) to the scores of the self-paraphrase pairs. Obviously, if we set $\lambda < 1$, the self-paraphrases will be penalized and the decoder will prefer to generate a paraphrase with more changes. If we set $\lambda > 1$, the decoder will tend to generate a paraphrase that is more similar to the source sentence. In our experiments, we set $\lambda = 0.1$ in D-2.

5 Experimental Setup

Our test sentences are extracted from the parallel reference translations of a Chinese-to-English MT evaluation[5], in which each Chinese sentence c has 4 English reference translations, namely e_1, e_2, e_3, and e_4. We use e_1 as a test sentence to paraphrase and e_2, e_3, e_4 as human paraphrases of e_1 for comparison with the automatically generated paraphrases. We process the test set by manually filtering ill-formed sentences, such as the ungrammatical or incomplete ones. 1182 out of 1357 test sentences are retained after filtering. Statistics show that about half of the test sentences are from news and the other half are from essays. The average length of the test sentences is 34.12 (words).

Score	Adequacy	Fluency
5	All	Flawless English
4	Most	Good English
3	Much	Non-native English
2	Little	Disfluent English
1	None	Incomprehensible

Table 5: Five point scale for human evaluation.

Manual evaluation is used in this work. A paraphrase T of a sentence S is manually scored based on a five point scale, which measures both the "adequacy" (i.e., how much of the meaning of S is preserved in T) and "fluency" of T (See Table 5). The five point scale used here is similar to that in the human evaluation of MT (Callison-Burch et al., 2007). In MT, adequacy and fluency are evaluated separately. However, we find that there is a high correlation between the two aspects, which makes it difficult to separate them. Thus we combine them in this paper.

We compare our method with a state-of-the-art approach SPG[6] (Zhao et al., 2009), which is a statistical approach specially designed for PG. The approach first collects a large volume of fine-grained paraphrase resources, including paraphrase phrases, patterns, and collocations, from various corpora using different methods. Then it generates paraphrases using these resources with a statistical model[7].

6 Experimental Results

We evaluate six approaches, i.e., S-1, S-2, S-3, D-1, D-2 and SPG, in the experiments. Each approach generates a 1-best paraphrase for a test sentence S. We randomize the order of the 6 paraphrases of each S to avoid bias of the raters.

[3] We conduct monotone decoding as previous work (Quirk et al., 2004; Zhao et al., 2008a, Zhao et al., 2009).

[4] The paraphrase rate reflects how different a paraphrase is from the source sentence.

[5] 2008 NIST Open Machine Translation Evaluation: Chinese to English Task.

[6] SPG: Statistical Paraphrase Generation.

[7] We ran SPG under the setting of baseline-2 as described in (Zhao et al., 2009).

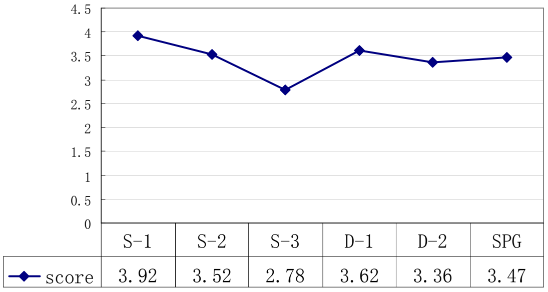


Figure 1: Evaluation results of the approaches.

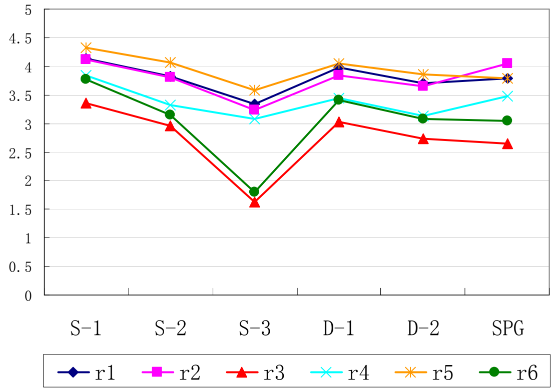


Figure 2: Evaluation results from each rater.

6.1 Human Evaluation Results

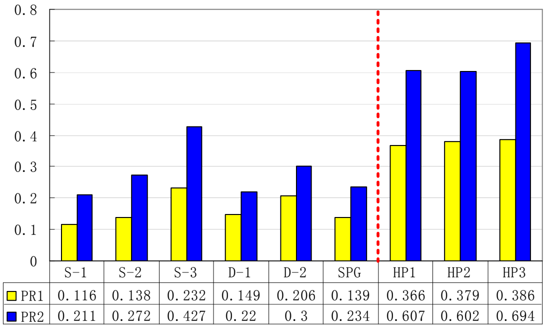
We have 6 raters in the evaluation, all of whom are postgraduate students. In particular, 3 raters major in English, while the other 3 major in computer science. Each rater scores the paraphrases of 1/6 test sentences, whose results are then combined to form the final scoring result. The average scores of the six approaches are shown in Figure 1. We can find that among the selection-based approaches, the performance of S-3 is the worst, which indicates that randomly selecting a paraphrase from the candidates works badly. S-2 performs much better than S-3, suggesting that the quality of the paraphrases acquired with the best single-pivot PG system are much higher than the randomly selected ones. S-1 performs the best in all the six approaches, which demonstrates the effectiveness of the MBR-based selection technique. Additionally, the fact that S-1 evidently outperforms S-2 suggests that it is necessary to extend a single-pivot approach to a multi-pivot one.

To get a deeper insight of S-1, we randomly sample 100 test sentences and manually score all of their candidates. We find that S-1 successfully picks out a paraphrase with the highest score for 72 test sentences. We further analyze the remaining 28 sentences for which S-1 fails and find that the failures are mainly due to the BLEU-based gain function. For example, S-1 sometimes selects paraphrases that have correct phrases but incorrect phrase orders, since BLEU is weak in evaluating phrase orders and sentence structures. In the next step we shall improve the gain function by investigating other features besides BLEU.

In the decoding-based approaches, D-1 ranks the second in the six approaches only behind S-1. We will further improve D-1 in the future rather than simply use Moses in decoding with the default setting. However, the value of D-1 lies in that it enables us to break down the candidates and generate new paraphrases flexibly. The performance decreases when we extend D-1 to D-2 to achieve a larger paraphrase rate. This is mainly because more errors are brought in when more parts of a sentence are paraphrased.

We can also find from Figure 1 that S-1, S-2, and D-1 all get higher scores than SPG, which shows that our method outperforms this state-of-the-art approach. This is more important if we consider that our method is lightweight, which makes no effort to collect fine-grained paraphrase resources beforehand. After observing the results, we believe that the outperformance of our method can be mainly ascribed to the selection-based and decoding-based techniques, since we avoid many errors by voting among the candidates. For instance, an ambiguous phrase may be incorrectly paraphrased by some of the single-pivot PG systems or the SPG approach. However, our method may obtain the correct paraphrase through statistics over all candidates and selecting the most credible one.

The human evaluation of paraphrases is subjective. Hence it is necessary to examine the coherence among the raters. The scoring results from the six raters are depicted in Figure 2. As it can be seen, they show similar trends though the raters have different degrees of strictness.



	S-1	S-2	S-3	D-1	D-2	SPG	HP1	HP2	HP3
🟨 PR1	0.116	0.138	0.232	0.149	0.206	0.139	0.366	0.379	0.386
🟦 PR2	0.211	0.272	0.427	0.22	0.3	0.234	0.607	0.602	0.694

Figure 3: Paraphrase rates of the approaches.

6.2 Paraphrase Rate

Human evaluation assesses the quality of paraphrases. However, the paraphrase rates cannot be reflected. A paraphrase that is totally transformed from the source sentence and another that is almost unchanged may get the same score. Therefore, we propose two strategies, i.e., PR1 and PR2, to compute the paraphrase rate:

$$PR1(T) = 1 - \frac{OL(S,T)}{L(S)}; \;\; PR2(T) = \frac{ED(S,T)}{L(S)}.$$

Here, PR1 is defined based on word overlapping rate, in which $OL(S,T)$ denotes the number of overlapping words between a paraphrase T and its source sentence S, $L(S)$ denotes the number of words in S. PR2 is defined based on edit distance, in which $ED(S,T)$ denotes the edit distance between T and S. Obviously, PR1 only measures the percentage of words that are changed from S to T, whereas PR2 further takes word order changes into consideration. It should be noted that PR1 and PR2 not only count the correct changes between S and T, but also count the incorrect ones. We compute the paraphrase rate for each of the six approaches by averaging the paraphrase rates over the whole test set. The results are shown in the left part of Figure 3.

On the whole, the paraphrase rates of the approaches are not high. In particular, we can see that the paraphrase rate of D-2 is clearly higher than D-1, which is in line with our intention of designing D-2. We can also see that the paraphrase rate of S-3 is the highest among the approaches. We find it is mainly because the paraphrases generated with S-3 contain quite a lot of errors, which contribute most of the changes.

7 Analysis

7.1 Effectiveness of the Proposed Method

Our analysis starts from the candidate paraphrases acquired with the multi-pivot approach. Actually, the results of S-3 reflect the average quality of the candidate paraphrases. A score of 2.78 (See Figure 1) indicates that the candidates are unacceptable according to the human evaluation metrics. This is in line with our expectation that the automatically acquired paraphrases through a two-way translation are noisy. However, the results of S-1 and D-1 demonstrate that, using the selection-based and decoding-based techniques, we can produce paraphrases of good quality. Especially, S-1 gets a score of nearly 4, which suggests that the paraphrases are pretty good according to our metrics. Moreover, our method outperforms SPG built on pre-extracted fine-grained paraphrases. It shows that our method makes good use of the paraphrase knowledge from the large volume of bilingual data underlying the multiple MT engines.

7.2 How to Choose Pivot Languages and MT Engines in the Multi-pivot Approach

In our experiments, besides the six pivot languages used in the multi-pivot system, we have also tried another five pivot languages, including Arabic, Japanese, Korean, Russian, and Dutch. They are finally abandoned since we find that they perform badly. Our experience on choosing pivot languages is that: (1) a pivot language should be a language whose translation quality can be well guaranteed by the MT engines; (2) it is better to choose a pivot language similar to the source language (e.g., French - English), which is easier to translate; (3) the translation quality of a pivot language should not vary a lot among the MT engines. On the other hand, it is better to choose MT engines built on diverse models and corpora, which can provide different paraphrase options. We plan to employ a syntax-based MT engine in our further experiments besides the currently used phrase-based SMT and rule-based MT engines.

S	he said there will be major cuts in the salaries of high-level civil servants .
S-1	he said **that** there will be **significant** cuts in the salaries of **senior officials** .
S-2	he said there will be major cuts in salaries of *civil servants high level* .
S-3	he said **that** there will be **significant** cuts in the salaries of **senior officials** .
D-1	he said **,** there will be **significant** cuts in salaries of **senior** civil servants .
D-2	he said **,** there will be **significant** cuts in salaries of **senior officials** .
SPG	he said **that** there will be *the main* cuts in the **wages** of high-level civil servants .
HP1	he said there will be a **big salary cut for high-level government employees** .
HP2	he said **salaries of senior public servants would be slashed** .
HP3	he **claimed to implement huge salary cut to senior** civil servants .

Table 6: Comparing the automatically generated paraphrases with the human paraphrases.

7.3 Comparing the Selection-based and Decoding-based Techniques

It is necessary to compare the paraphrases generated via the selection-based and decoding-based techniques. As stated above, the selection-based technique can only select a paraphrase from the candidates, while the decoding-based technique can generate a paraphrase different from all candidates. In our experiments, we find that for about 90% test sentences, the paraphrases generated by the decoding-based approach D-1 are outside the candidates. In particular, we compare the paraphrases generated by S-1 and D-1 and find that, for about 40% test sentences, S-1 gets higher scores than D-1, while for another 21% test sentences, D-1 gets higher scores than S-1[8]. This indicates that the selection-based and decoding-based techniques are complementary. In addition, we find examples in which the decoding-based technique can generate a perfect paraphrase for the source sentence, even if all the candidate paraphrases have obvious errors. This also shows that the decoding-based technique is promising.

7.4 Comparing Automatically Generated Paraphrases with Human Paraphrases

We also analyze the characteristics of the generated paraphrases and compare them with the human paraphrases (i.e., the other 3 reference translations in the MT evaluation, see Section 5, which are denoted as HP1, HP2, and HP3). We find that, compared with the automatically generated paraphrases, the human paraphrases are more complicated, which involve not only phrase replacements, but also structure reformulations and even inferences. Their paraphrase rates are also much higher, which can be seen in the right part of Figure 3. We show the automatic and human paraphrases for the example sentence of this paper in Table 6. To narrow the gap between the automatic and human paraphrases, it is necessary to learn structural paraphrase knowledge from the candidates in the future work.

8 Conclusions and Future Work

We put forward an effective method for paraphrase generation, which has the following contributions. First, it acquires a rich fund of paraphrase knowledge through the use of multiple MT engines and pivot languages. Second, it presents a MBR-based technique that effectively selects high-quality paraphrases from the noisy candidates. Third, it proposes a decoding-based technique, which can generate paraphrases that are different from the candidates. Experimental results show that the proposed method outperforms a state-of-the-art approach SPG.

In the future work, we plan to improve the selection-based and decoding-based techniques. We will try some standard system combination strategies, like confusion networks and consensus decoding. In addition, we will refine our evaluation metrics. In the current experiments, paraphrase correctness (adequacy and fluency) and paraphrase rate are evaluated separately, which seem to be incompatible. We plan to combine them together and propose a uniform metric.

[8] For the rest 39%, S-1 and D-1 get identical scores.

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of ACL*, pages 597-604.

Regina Barzilay and Lillian Lee. 2003. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In *Proceedings of HLT-NAACL*, pages 16-23.

Peter J. Bickel and Kjell A. Doksum. 1977. *Mathematical Statistics: Basic Ideas and Selected Topics*. Holden-Day Inc., Oakland, CA, USA.

Igor A. Bolshakov and Alexander Gelbukh. 2004. Synonymous Paraphrasing Using WordNet and Internet. In *Proceedings of NLDB*, pages 312-323.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz and Josh Schroeder. 2007. (Meta-) Evaluation of Machine Translation. In *Proceedings of ACL-2007 Workshop on Statistical Machine Translation*, pages 136-158.

Chris Callison-Burch, Philipp Koehn, and Miles Osborne. 2006. Improved Statistical Machine Translation Using Paraphrases. In *Proceedings of HLT-NAACL*, pages 17-24.

Pablo Ariel Duboue and Jennifer Chu-Carroll. 2006. Answering the Question You Wish They Had Asked: The Impact of Paraphrasing for Question Answering. In *Proceedings of HLT-NAACL*, pages 33-36.

Hieu Hoang and Philipp Koehn. 2008. Design of the Moses Decoder for Statistical Machine Translation. In *Proceedings of ACL Workshop on Software engineering, testing, and quality assurance for NLP*, pages 58-65.

Lidija Iordanskaja, Richard Kittredge, and Alain Polguère. 1991. Lexical Selection and Paraphrase in a Meaning-Text Generation Model. In Cécile L. Paris, William R. Swartout, and William C. Mann (Eds.): Natural Language Generation in Artificial Intelligence and Computational Linguistics, pages 293-312.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for Automatic Evaluation. In *Proceedings of HLT-NAACL*, pages 455-462.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models: User Manual and Description for Version 1.2.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL*, pages 127-133.

Raymond Kozlowski, Kathleen F. McCoy, and K. Vijay-Shanker. 2003. Generation of single-sentence paraphrases from predicate/argument structure using lexico-grammatical resources. In *Proceedings of IWP*, pages 1-8.

Aurélien Max. 2009. Sub-sentential Paraphrasing by Contextual Pivot Translation. In *Proceedings of the 2009 Workshop on Applied Textual Inference, ACL-IJCNLP 2009*, pages 18-26.

Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of ACL*, pages 440-447.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based Alignment of Multiple Translations: Extracting Paraphrases and Generating New Sentences. In *Proceedings of HLT-NAACL*, pages 102-109.

Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*, pages 311-318.

Richard Power and Donia Scott. 2005. Automatic generation of large-scale paraphrases. In *Proceedings of IWP*, pages 73-79.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual Machine Translation for Paraphrase Generation. In *Proceedings of EMNLP*, pages 142-149.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal and Yi Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In *Proceedings of ACL*, pages 464-471.

Yujie Zhang and Kazuhide Yamamoto. 2002. Paraphrasing of Chinese Utterances. In *Proceedings of COLING*, pages 1163-1169.

Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven Statistical Paraphrase Generation. In *Proceedings of ACL-IJCNLP 2009*, pages 834-842.

Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. 2008a. Combining Multiple Resources to Improve SMT-based Paraphrasing Model. In *Proceedings of ACL-08:HLT*, pages 1021-1029.

Shiqi Zhao, Haifeng Wang, Ting Liu, and Sheng Li. 2008b. Pivot Approach for Extracting Paraphrase Patterns from Bilingual Corpora. In *Proceedings of ACL-08:HLT*, pages 780-788.

Resolving Surface Forms to Wikipedia Topics

Yiping Zhou Lan Nie Omid Rouhani-Kalleh Flavian Vasile Scott Gaffney
Yahoo! Labs at Sunnyvale
{zhouy,lannie,omid,flavian,gaffney}@yahoo-inc.com

Abstract

Ambiguity of entity mentions and concept references is a challenge to mining text beyond surface-level keywords. We describe an effective method of disambiguating surface forms and resolving them to Wikipedia entities and concepts. Our method employs an extensive set of features mined from Wikipedia and other large data sources, and combines the features using a machine learning approach with automatically generated training data. Based on a manually labeled evaluation set containing over 1000 news articles, our resolution model has 85% precision and 87.8% recall. The performance is significantly better than three baselines based on traditional context similarities or sense commonness measurements. Our method can be applied to other languages and scales well to new entities and concepts.

1 Introduction

Ambiguity in natural language is prevalent and, as such, it can be a difficult challenge for information retrieval systems and other text mining applications. For example, a search for "*Ford*" in Yahoo! News retrieves about 40 thousand articles containing *Ford* referring to a company (Ford Motors), an athlete (Tommy Ford), a place (Ford City), etc. Due to reference ambiguity, even if we knew the user was only interested in the company, they would still have to contend with articles referring to the other concepts as well.

In this paper we focus on the problem of resolving references of named-entities and concepts in natural language through their textual *surface forms*. Specifically, we present a method of resolving surface forms in general text documents to Wikipedia entries. The tasks of resolution and disambiguation are nearly identical; we make the distinction that resolution specifically applies when a known set of referent concepts are given a priori. Our approach differs from others in multiple aspects including the following.

1) We employ a rich set of disambiguation features leveraging mining results from large-scale data sources. We calculate context-sensitive features by extensively mining the categories, links and contents of the entire Wikipedia corpus. Additionally we make use of context-independent data mined from various data sources including Web user-behavioral data and Wikipedia. Our features also capture the one-to-one relationship between a surface form and its referent.

2) We use machine learning methods to train resolution models with a large automatically labeled training set. Both ranking-based and classification-based resolution approaches are explored.

3) Our method disambiguates both entities and word senses. It scales well to new entities and concepts, and it can be easily applied to other languages.

We propose an extensive set of metrics to evaluate not only overall resolution performance but also out-of-Wikipedia prediction. Our systems for English language are evaluated using real-world test sets and compared with a number of baselines. Evaluation results show that our systems consistently and significantly outperform others across all test sets.

The paper is organized as follows. We first describe related research in Section 2, followed by an introduction of Wikipedia in Section 3. We then introduce our learning method in Section 4 and our features in Section 5. We show our experimental results in Section 6, and finally close with a discussion of future work.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1335–1343,
Beijing, August 2010

2 Related Work

Named entity disambiguation research can be divided into two categories: some works (Bagga and Baldwin, 1998; Mann and Yarowsky, 2003; Pedersen et al., 2005; Fleischman and Hovy, 2004; Ravin and Kazi, 1999) aim to cluster ambiguous surface forms to different groups, with each representing a unique entity; others (Cucerzan, 2007; Bunescu and Paşca, 2006; Han and Zhao, 2009; Milne and Witten, 2008a; Milne and Witten, 2008b) resolve a surface form to an entity or concept extracted from existing knowledge bases. Our work falls into the second category.

Looking specifically at resolution, Bunescu and Pasca (2006) built a taxonomy SVM kernel to enrich a surface form's representation with words from Wikipedia articles in the same category. Cucerzan (2007) employed context vectors consisting of phrases and categories extracted from Wikipedia. The system also attempted to disambiguate all surface forms in a context simultaneously, with the constraint that their resolved entities should be globally consistent on the category level as much as possible. Milne and Witten (2008a, 2008b) proposed to use Wikipedia's link structure to capture the *relatedness* between Wikipedia entities so that a surface form is resolved to an entity based on its relatedness to the surface form's surrounding entities. Besides relatedness, they also define a *commonness* feature that captures how common it is that a surface form links to a particular entity in general. Han and Zhao (2009) defined a novel alignment strategy to calculate similarity between surface forms based on semantic relatedness in the context.

Milne and Witten's work is most related to what we propose here in that we also employ features similar to their relatedness and commonness features. However, we add to this a much richer set of features which are extracted from Web-scale data sources beyond Wikipedia, and we develop a machine learning approach to automatically blend our features using completely automatically generated training data.

3 Wikipedia

Wikipedia has more than 200 language editions, and the English edition has more than 3 million articles as of March 2009. Newsworthy events are often added to Wikipedia within days of occurrence; Wikipedia has bi-weekly snapshots available for download.

Each article in Wikipedia is uniquely identified by its title which is usually the most common surface form of an entity or concept. Each article includes body text, outgoing links and categories. Here is a sample sentence in the article titled "Aristotle" in wikitext format. "*Together with Plato and [[Socrates]] (Plato's teacher), Aristotle is one of the most important founding figures in [[Western philosophy]].*" Near the end of the article, there are category links such as "*[[Category:Ancient Greek mathematicians]]*". The double brackets annotate outgoing links to other Wikipedia articles with the specified titles. The category names are created by authors. Articles and category names have many-to-many relationships.

In addition to normal articles, Wikipedia also has special types of articles such as redirect articles and disambiguation articles. A redirect article's title is an alternative surface form for a Wikipedia entry. A disambiguation article lists links to similarly named articles, and usually its title is a commonly used surface form for multiple entities and concepts.

4 Method of Learning

Our goal is to resolve surface forms to entities or concepts described in Wikipedia. To this end, we first need a recognizer to detect surface forms to be resolved. Then we need a resolver to map a surface form to the most probable entry in Wikipedia (or to *out-of-wiki*) based on the context.

Recognizer: We first create a set of Wikipedia (article) entries $E = \{e_1, e_2, ...\}$ to which we want to resolve surface forms. Each entry's surface forms are mined from multiple data sources. Then we use simple string match to recognize surface forms from text documents.

Among all Wikipedia entries, we exclude those with low importance. In our experiments, we removed the entries that would not interest general Web users, such as stop words and punctuations. Second, we collect surface forms for entries in E using Wikipedia and Web search query click logs based on the following assumptions:

- Each Wikipedia article title is a surface form for the entry. Redirect titles are taken as alternative surface forms for the target entry.
- The anchor text of a link from one article to another is taken as an alternative surface form for the linked-to entry.
- Web search engine queries resulting in user clicks on a Wikipedia article are taken as alternative surface forms for the entry.

As a result, we get a number of surface forms for each entry e_i. If we let s_{ij} denote the j-th surface form for entry i, then we can represent our entry dictionary as $EntSfDict = \{<e_1, (s_{11}, s_{12}, ...)>, <e_2, (s_{21}, s_{22}, ...)>, ...\}$.

Resolver: We first build a labeled training set automatically, and then use supervised learning methods to learn models to resolve among Wikipedia entries. In the rest of this section we describe the resolver in details.

4.1 Automatically Labeled Data

To learn accurate models, supervised learning methods require training data with both large quantity and high quality, which often takes lots of human labeling effort. However, in Wikipedia, links provide a *supervised* mapping from surface forms to article entries. We use these links to automatically generate training data. If a link's anchor text is a surface form in *EntSfDict*, we extract the anchor text as surface form *s* and the link's destination article as Wikipedia entry *e*, then add the pair *(s, e)* with a positive judgment to our labeled example set. Continuing, we use *EntSfDict* to find other Wikipedia entries for which *s* is a surface form and create negative examples for these and add them to our labeled example set. If *e* does not exist in *EntSfDict* (for example, if the link points to a Wikipedia article about a stop word), then a negative training example is created for every Wikipedia entry to which *s* may resolve. We use *oow* (out-of-wiki) to denote this case.

Instead of article level coreference resolution, we only match partial names with full names based on the observation that surface forms for named entities are usually capitalized word sequences in English language and a named entity is often mentioned by a long surface form followed by mentions of short forms in the same article. For each pair *(s, e)* in the labeled example set, if *s* is a partial name of a full name *s'* occurring earlier in the same document, we replace *(s, e)* with *(s', e)* in the labeled example set.

Using this methodology we created 2.4 million labeled examples from only 1% of English Wikipedia articles. The abundance of data made it possible for us to experiment on the impact of training set size on model accuracy.

4.2 Learning Algorithms

In our experiments we explored both Gradient Boosted Decision Trees (GBDT) and Gradient Boosted Ranking (GBRank) to learn resolution models. They both can easily combine features of different scale and with missing values. Other supervised learning methods are to be explored in the future.

GBDT: We use the stochastic variant of GBDTs (Friedman, 2001) to learn a binary logistic regression model with the judgments as the target. GBDTs compute a function approximation by performing a numerical optimization in the function space. It is done in multiple stages, with each stage modeling residuals from the model of the last stage using a small decision tree. A brief summary is given in Algorithm 1. In the stochastic version of GBDT, one sub-samples the training data instead of using the entire training set to compute the loss function.

Algorithm 1 GBDTs

```
Input: training data $\{(x_i, y_i)\}_{i=1}^{N}$, loss function
$L[y, f(x)]$, the number of nodes for each tree $J$
, the number of trees $M$.
1:    Initialize $f(x)=f^0$
2:    For $m = 1$ to $M$
2.1:     For $i = 1$ to $N$, compute the negative
         gradient by taking the derivative of the
         loss with respect to $f(x)$ and substitute
         with $y_i$ and $f_i^{m-1}(x_i)$.
2.2:     Fit a $J$-node regression tree to the
         components of the negative gradient.
2.3:     Find the within-node updates $a_j^m$ for $j$
         $= 1$ to $J$ by performing $J$ univariate op-
         timizations of the node contributions to
         the estimated loss.
2.4:     Do the update $f_i^m(x_i) = f_i^{m-1}(x_i) + r \times a_j^m$,
         where $j$ is the node that $x_i$ belongs to, $r$
         is learning rate.
3:    End for
4:    Return $f^M$
```

In our setting, the loss function is a negative binomial log-likelihood, x_i is the feature vector for a surface-form and Wikipedia-entry pair *(s_i, e_i)*, and y_i is +1 for positive judgments and -1 is for negative judgments.

GBRank: From a given surface form's judgments we can infer that the correct Wikipedia entry is preferred over other entries. This allows us to derive pair-wise preference judgments from absolute judgments and train a model to rank all the Wikipedia candidate entries for each surface form. Let $S=\{(x_i,x_i^{'})\,|\,l(x_i)\geq l(x_i^{'}),i=1,...,N\}$ be the set of preference judgments, where x_i and x_i' are the feature vectors for two pairs of surface-forms and Wikipedia-entry, $l(x_i)$ and $l(x_i')$ are their absolute judgments respectively. GBRank (Zheng et al., 2007) tries to learn a function h such that $h(x_i)\geq h(x_i^{'})$ for $(x_i,x_i^{'})\in S$. A sketch of the algorithm is given in Algorithm 2.

Algorithm 2 GBRank

```
1:    Initialize $h=h_0$
2:    For $k=1$ to $K$
2.1:    Use $h_{k-1}$ as an approximation of $h$ and
        compute
        $S^{+}=\{(x_i,x_i^{'})\in S\,|\,h_{k-1}(x_i)\geq h_{k-1}(x_i^{'})+\tau\}$
        $S^{-}=\{(x_i,x_i^{'})\in S\,|\,h_{k-1}(x_i)< h_{k-1}(x_i^{'})+\tau\}$
        where $\tau=\alpha(l(x_i)-l(x_i^{'}))$
2.2:    Fit a regression function $g_k$ using
        GBDT and the incorrectly predicted
        examples
        $\{(x_i,h_{k-1}(x_i^{'})+\tau),(x_i^{'},h_{k-1}(x_i)-\tau)\,|\,(x_i,x_i^{'})\in S^{-}\}$
2.3:    Do the update
        $h_k(x)=(kh_{k-1}(x)+\eta g_k(x))/(k+1)$, where $\eta$ is
        learning rate.
3:    End for
4:    Return $h_K$
```

We use a tuning set independent from the training set to select the optimal parameters for GBDT and GBRank. This includes the number of trees M, the number of nodes J, the learning rate r, and the sampling rate for GBDT; and for GBRank we select K, α and η.

The feature importance measurement given by GBDT and GBRank is computed by keeping track of the reduction in the loss function at each feature variable split and then computing the total reduction of loss along each explanatory feature variable. We use it to analyze feature effectiveness.

4.3 Prediction

After applying a resolution model on the given test data, we obtain a score for each surface-form and Wikipedia-entry pair *(s, e)*. Among all the pairs containing s, we find the pair with the highest score, denoted by *(s, $\tilde{e}$)*.

It's very common that a surface form refers to an entity or concept not defined in Wikipedia. So it's important to correctly predict whether the given surface form cannot be mapped to any Wikipedia entry in *EntSfDict*.

We apply a threshold to the scores from resolution models. If the score for *(s, $\tilde{e}$)* is lower than the threshold, then the prediction is *oow* (see Section 4.1), otherwise $\tilde{e}$ is predicted to be the entry referred by s. We select thresholds based on F1 (see Section 6.2) on a tuning set that is independent from our training set and test set.

5 Features

For each surface-form and Wikipedia-entry pair *(s, e)*, we create a feature vector including features capturing the context surrounding s and features independent of the context. They are context-dependent and context-independent features respectively. Various data sources are mined to extract these features, including Wikipedia articles, Web search query-click logs, and Web-user browsing logs. In addition, *(s, e)* is compared to all pairs containing s based on above features and the derived features are called differentiation features.

5.1 Context-dependent Features

These features measure whether the given surface form s resolving to the given Wikipedia entry e would make the given document more coherent. They are based on 1) the vector representation of e, and 2) the vector representation of the context of s in a document d.

Representation of *e*: By thoroughly mining Wikipedia and other large data sources we extract contextual clues for each Wikipedia entry e and formulate its representation in the following ways.

1) *Background representation*. The overall background description of e is given in the corresponding Wikipedia article, denoted as A_e. Naturally, a bag of terms and surface forms in A_e can represent e. So we represent e by a back-

ground word vector E_{bw} and a background surface form vector E_{bs}, in which each element is the occurrence count of a word or a surface form in A_e's first paragraph.

2) *Co-occurrence representation.* The terms and surface forms frequently co-occurring with *e* capture its contextual characteristics. We first identify all the Wikipedia articles linking to A_e. Then, for each link pointing to A_e we extract the surrounding words and surface forms within a window centered on the anchor text. The window size is set to 10 words in our experiment. Finally, we select the words and surface forms with the top co-occurrence frequency, and represent *e* by a co-occurring word vector E_{cw} and a co-occurring surface form vector E_{cs}, in which each element is the co-occurrence frequency of a selected word or surface form.

3) *Relatedness representation.* We analyzed the relatedness between Wikipedia entries from different data sources using various measurements, and we computed over 20 types of relatedness scores in our experiments. In the following we discuss three types as examples. The first type is computed based on the overlap between two Wikipedia entries' categories. The second type is mined from Wikipedia inter-article links. (In our experiments, two Wikipedia entries are considered to be related if the two articles are mutually linked to each other or co-cited by many Wikipedia articles.) The third type is mined from Web-user browsing data based on the assumption that two Wikipedia articles co-occurring in the same browsing session are related. We used approximately one year of Yahoo! user data in our experiments. A number of different metrics are used to measure the relatedness. For example, we apply the algorithm of Google distance (Milne and Witten, 2008b) on Wikipedia links to calculate the Wikipedia link-based relatedness, and use mutual information for the browsing-session-based relatedness. In summary, we represent *e* by a related entry vector E_r for each type of relatedness, in which each element is the relatedness score between *e* and a related entry.

Representation of *s*: We represent a surface form's context as a vector, then calculate a context-dependent feature for a pair <*s,e*> by a similarity function *Sim* from two vectors. Here are examples of context representation.

1) *s* is represented by a word vector S_w and a surface form vector S_s, in which each element is the occurrence count of a word or a surface form surrounding *s.* We calculate each vector's similarity with the background and co-occurrence representation of *e*, and it results in $Sim(S_w, E_{bw})$, $Sim(S_w, E_{cw})$, $Sim(S_s, E_{bs})$ and $Sim(S_s, E_{cs})$.

2) *s* is represented by a Wikipedia entry vector S_e, in which each element is a Wikipedia entry to which a surrounding surface form *s* could resolve. We calculate its similarity with the relatedness representation of *e*, and it results in $Sim(S_e, E_r)$.

In the above description, similarity is calculated by dot product or in a summation-of-maximum fashion. In our experiments we extracted surrounding words and surface forms for *s* from the whole document or from the text window of 55 tokens centered on *s*, which resulted in 2 sets of features. We created around 50 context-dependent features in total.

5.2 Context-independent Features

These features are extracted from data beyond the document containing s. Here are examples.

- During the process of building the dictionary *EntSfDict* as described in Section 4, we count how often *s* maps to *e* and estimate the probability of *s* mapping to *e* for each data source. These are the commonness features.
- The number of Wikipedia entries that *s* could map to is a feature about the ambiguity of *s*.
- The string similarity between *s* and the title of A_e is used as a feature. In our experiments string similarity was based on word overlap.

5.3 Differentiation Features

Among all surface-form and Wikipedia-entry pairs that contain *s*, at most one pair gets the positive judgment. Based on this observation we created differentiation features to represent how *(s, e)* is compared to other pairs for *s*. They are derived from the context-dependent and context-independent features described above. For example, we compute the difference between the string similarity for *(s, e)* and the maximum string similarity for all pairs containing *s*. The derived feature value would be zero if *(s, e)* has larger string similarity than other pairs containing *s*.

6 Experimental Results

In our experiments we used the Wikipedia snapshot for March 6th, 2009. Our dictionary *EntSfDict* contains 3.5 million Wikipedia entries and 6.5 million surface forms.

A training set was created from randomly selected Wikipedia articles using the process described in Section 4.1. We varied the number of Wikipedia articles from 500 to 40,000, but the performance did not increase much after 5000. The experimental results reported in this paper are based on the training set generated from 5000 articles. It contains around 1.4 million training examples. There are approximately 300,000 surface forms, out of which 28,000 are the *oow* case.

Around 400 features were created in total, and 200 of them were selected by GBDT and GBRank to be used in our resolution models.

6.1 Evaluation Datasets

Three datasets from different data sources are used in evaluation.

1) *Wikipedia hold-out set*. Using the same process for generating training data and excluding the surface forms appearing in the training data, we built the hold-out set from approximately 15,000 Wikipedia articles, containing around 600,000 labeled instances. There are 400,000 surface forms, out of which 46,000 do not resolve to any Wikipedia entry.

2) *MSNBC News test set*. This entity disambiguation data set was introduced by Cucerzan (2007). It contains 200 news articles collected from ten MSNBC news categories as of January 2, 2007. Surface forms were manually identified and mapped to Wikipedia entities. The data set contains 756 surface forms. Only 589 of them are contained in our dictionary *EntSfDict*, mainly because *EntSfDict* excludes surface forms of out-of-Wikipedia entities and concepts. Since the evaluation task is focused on resolution performance rather than recognition, we exclude the missing surface forms from the labeled example set. The final dataset contains 4,151 labeled instances. There are 589 surface forms and 40 of them do not resolve to any Wikipedia entry.

3) *Yahoo! News set*. One limitation of the MSNBC test set is the small size. We built a much larger data set by randomly sampling around 1,000 news articles from Yahoo! News over 2008 and had them manually annotated. The experts first identified *person*, *location* and *organization* names, then mapped each name to a Wikipedia article if the article is about the entity referred to by the name. We didn't include more general concepts in this data set to make the manual effort easier. This data set contains around 100,000 labeled instances. The data set includes 15,387 surface forms and 3,532 of them cannot be resolved to any Wikipedia entity. We randomly split the data set to 2 parts of equal size. One part is used to tune parameters of GBDT and GBRank and select thresholds based on F1 value. The evaluation results presented in this paper is based on the remaining part of the Yahoo! News set.

6.2 Metrics

The possible outcomes from comparing a resolution system's prediction with ground truth can be categorized into the following types.

- True Positive (TP), the predicted *e* was correctly referred to by *s*.
- True Negative (TN), *s* was correctly predicted as resolving to *oow*.
- Mismatch (MM), the predicted *e* was not correctly referred to by *s* and should have been *e'* from *EntSfDict*.
- False Positive (FP), the predicted *e* was not correctly referred to by *s* and should have been *oow*.
- False Negative (FN), the predicted *oow* is not correct and should have been *e'* from *EntSfDict*.

Similar to the widely used metrics for classification systems, we use following metrics to evaluate disambiguation performance.

$$precision = \frac{TP}{TP+FP+MM} \qquad recall = \frac{TP}{TP+FN+MM}$$

$$F1 = \frac{2\times precision \times recall}{precision + recall} \qquad accuracy = \frac{TP+TN}{TP+FP+TN+FN+MM}$$

In the Yahoo! News test set, 23.5% of the surface forms do not resolve to any Wikipedia entries, and in the other two test sets the percentages of *oow* are between 10% and 20%. This demonstrates it is necessary in real-world applications to explicitly measure *oow* prediction. We propose following metrics.

$$precision\_oow = \frac{TN}{TN+FN} \qquad recall\_oow = \frac{TN}{TN+FP}$$

$$F1\_oow = \frac{2\times precision\_oow \times recall\_oow}{precision\_oow + recall\_oow}$$

6.3 Evaluation Results

With our training set we trained one resolution model using GBDT (named as *WikiRes-c*) and another resolution model using GBRank (named as *WikiRes-r*). The models were evaluated along with the following systems.

1) *Baseline-r*: each surface form *s* is randomly mapped to *oow* or a candidate entry for *s* in *EntSfDict*.

2) *Baseline-p*: each surface form *s* is mapped to the candidate entry *e* for *s* with the highest commonness score. The commonness score is linear combination of the probability of *s* being mapped to *e* estimated from different data sources. The commonness score is among the features used in *WikiRes-c* and *WikiRes-r*.

3) *Baseline-m*: we implemented the approach brought by Cucerzan (2007) based on our best understanding. Since we use a different version of Wikipedia and a different entity recognition approach, the evaluation result differs from the result presented in their paper. But we believe our implementation follows the algorithm described in their paper.

In Table 1 we present the performance for each system on the Yahoo! News test set and the MSNBC test set. The performance of *WikiRes-c* and *WikiRes-r* are computed after we apply the thresholds selected on the tuning set described in Section 6.1. In the upper half of Table 1, the three baselines use the thresholds that lead to the best F1 on the Yahoo! News test set. In the lower half of Table 1, the three baselines use the thresholds that lead to the best F1 on the MSNBC test set.

Among the three baselines, *Baseline-r* has the lowest performance. *Baseline-m* uses a few context-sensitive features and *Baseline-p* uses a context-independent feature. These two types of features are both useful, but *Baseline-p* shows better performance, probably because the surface forms in our test sets are dominated by common senses. In our resolution models, these features are combined together with many other features calculated from different large-scale data sources and on different granularity levels. As shown in Table 1, both of our resolution solutions substantially outperform other systems. Furthermore, *WikiRes-c* and *WikiRes-r* have similar performance.

	Precision	**Recall**	**F1**	**Accuracy**	**p-value**
Yahoo! News Test Set					
Baseline-r	47.023	60.831	53.043	47.023	0
Baseline-p	73.869	88.157	80.383	73.175	5.2e-78
Baseline-m	62.240	80.517	70.208	62.240	1.3e-160
WikiRes-r	83.406	**88.858**	86.046	80.717	0.012
WikiRes-c	**85.038**	87.831	**86.412**	**81.463**	---
MSNBC Test Set					
Baseline-r	60.272	64.545	62.335	60.272	8.9e-19
Baseline-p	82.292	86.182	84.192	82.003	0.306
Baseline-m	78.947	84.545	81.651	78.947	0.05
WikiRes-r	**88.785**	**86.364**	**87.558**	**84.550**	0.102
WikiRes-c	88.658	85.273	86.932	83.192	---

Table 1. Performance on the Yahoo! News Test Set and the MSNBC Test set

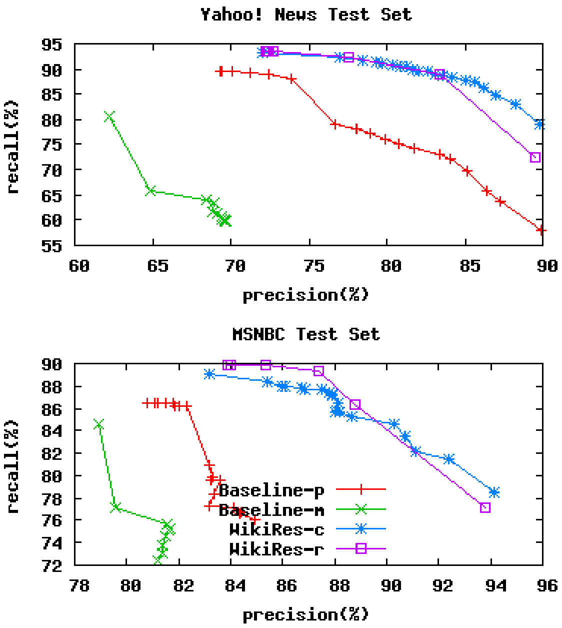


Figure 1. Precision-recall on the Yahoo! News Test Set and the MSNBC Test Set

We compared *WikiRes-c* with each competitor and from the statistical significance test results in the last column of Table 1 we see that on the Yahoo! News test set *WikiRes-c* significantly outperforms others. The p-values for the MSNBC test set are much higher than for the Yahoo! News test set because the MSNBC test set is much smaller.

Attempting to address this point, we see that the F1 values of *WikiRes* on the MSNBC test set and on the Yahoo! News test set only differs by a couple percentage points, although, these test sets were created independently. This suggests the objectivity of our method for creating the Yahoo! News test set and provides a way to measure resolution model performance on what would occur in a general news corpus in a statistically significant manner.

In Figure 1 we present the precision-recall curves on the Yahoo! News and the MSNBC test sets. We see that our resolution models are substantially better than the other two baselines at any particular precision or recall value on both test sets. *Baseline-r* is not included in the comparison since it does not have the tradeoff between precision and recall. We find the precision-recall curve of *WikiRes-r* is very similar to *WikiRes-c* at the lower precision area, but its recall is much lower than other systems after precision reaches around 90%. So, in Figure 1 the curves of *WikiRes-r* are truncated at the high precision area.

In Table 2 we compare the performance of out-of-Wikipedia prediction. The comparison is done on the Yahoo! News test set only, since there are only 40 surface forms of *oow* case in the MSNBC test set. Each system's threshold is the same as that used for the upper half of Table 1. The results show our models have substantially higher precision and recall than *Baseline-p* and *Baseline-m*. From the statistical significance test results in the last column, we can see that *WikiRes-c* significantly outperforms *Baseline-p* and *Baseline-m*. Also, our current approaches still have room to improve in the area of out-of-Wikipedia prediction.

We also evaluated our models on a Wikipedia hold-out set. The model performance is greater than that obtained from the previous two test sets because the hold-out set is more similar to the training data source itself. Again, our models perform better than others.

From the feature importance lists of our GBDT model and GBRank model, we find that the commonness features, the features based on Wikipedia entries' co-occurrence representation and the corresponding differentiation features are the most important.

	Precision	Recall	F1	p-value
Baseline-p	64.907	22.152	33.03	1.6e-20
Baseline-m	47.207	44.78	45.961	1.3e-34
WikiRes-r	**68.166**	52.994	59.630	0.084
WikiRes-c	67.303	**59.777**	**63.317**	---

Table 2. Performance of Out-of-Wikipedia Prediction on the Yahoo! News Test Set

7 Conclusions

We have described a method of learning to resolve surface forms to Wikipedia entries. Using this method we can enrich the unstructured documents with structured knowledge from Wikipedia, the largest knowledge base in existence. The enrichment makes it possible to represent a document as a machine-readable network of senses instead of just a bag of words. This can supply critical semantic information useful for next-generation information retrieval systems and other text mining applications.

Our resolution models use an extensive set of novel features and are leveraged by a machine learned approach that depends only on a purely automated training data generation facility. Our methodology can be applied to any other language that has Wikipedia and Web data available (after modifying the simple capitalization rules in Section 4.1). Our resolution models can be easily and quickly retrained with updated data when Wikipedia and the relevant Web data are changed.

For future work, it will be important to investigate other approaches to better predict *oow*. Adding global constraints on resolutions of the same term at multiple locations in the same document may also be important. Of course, developing new features (such as part-of-speech, named entity type, etc) and improving training data quality is always critical, especially for social content sources such as those from Twitter. Finally, directly demonstrating the degree of applicability to other languages is interesting when accounting for the fact that the quality of Wikipedia is variable across languages.

References

Bagga, Amit and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the Vector Space Model. *Proceedings of the 17th international conference on Computational linguistics*.

Bunescu, Razvan and Marius Paşca. 2006. Using Encyclopedic Knowledge for Named Entity Disambiguation. *Proceedings of the 11th Conference of the European Chapter of the Association of Computational Linguistics (EACL-2006)*.

Cucerzan, Silviu. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. *Pro-*

ceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.

Fleischman, Ben Michael and Eduard Hovy. 2004. Multi-Document Person Name Resolution. *Proceesing of the Association for Computational Linguistics*.

Friedman, J. H. 2001. Stochastic gradient boosting. *Computational Statistics and Data Analysis,* 38:367–378.

Han, Xianpei and Jun Zhao 2009. Named Entity Disambiguation by Leveraging Wikipedia Semantic Knowledge. *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval.*

Mann, S. Gidon and David Yarowsky. 2003. Unsupervised Personal Name Disambiguation. *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003.*

Milne, David and Ian H. Witten. 2008a. Learning to Link with Wikipedia. *In Proceedings of the ACM Conference on Information and Knowledge Management (CIKM'2008).*

Milne, David and Ian H. Witten. 2008b. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. *Proceedings of the first AAAI Workshop on Wikipedia and Artificial Intelligence.*

Pedersen, Ted, Amruta Purandare and Anagha Kulkarni. 2005. Name Discrimination by Clustering Similar Contexts. *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics (2005).*

Ravin, Y. and Z. Kazi. 1999. Is Hillary Rodham Clinton the President? In *Association for Computational Linguistics Workshop on Coreference and its Applications.*

Yarowsky, David. 1995. Unsupervised word sense disambiguation rivaling supervised methods. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics,* pages 189-196.

Zheng, Zhaohui, K. Chen, G. Sun, and H. Zha. 2007. A regression framework for learning ranking functions using relative relevance judgments. *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval,* pages 287-294.

Heterogeneous Parsing via Collaborative Decoding

Muhua Zhu **Jingbo Zhu** **Tong Xiao**
Natural Language Processing Lab.
Northeastern University
zhumuhua@gmail.com
{zhujingbo, xiaotong}@mail.neu.edu.cn

Abstract

There often exist multiple corpora for the same natural language processing (NLP) tasks. However, such corpora are generally used independently due to distinctions in annotation standards. For the purpose of full use of readily available human annotations, it is significant to simultaneously utilize multiple corpora of different annotation standards. In this paper, we focus on the challenge of constituent syntactic parsing with treebanks of different annotations and propose a collaborative decoding (or co-decoding) approach to improve parsing accuracy by leveraging bracket structure consensus between multiple parsing decoders trained on individual treebanks. Experimental results show the effectiveness of the proposed approach, which outperforms state-of-the-art baselines, especially on long sentences.

1 Introduction

Recent years have seen extensive applications of machine learning methods to natural language processing problems. Typically, increase in the scale of training data boosts the performance of machine learning methods, which in turn enhances the quality of learning-based NLP systems (Banko and Brill, 2001). However, annotating data by human is expensive in time and labor. For this reason, human-annotated corpora are considered as the most valuable resource for NLP.

In practice, there often exist more than one corpus for the same NLP tasks. For example, for constituent syntactic parsing (Collins, 1999; Charniak, 2000; Petrov et al., 2006) in Chinese, in addition to the most popular treebank Chinese Treebank (CTB) (Xue et al., 2002), there are also other treebanks such as Tsinghua Chinese Treebank (TCT) (Zhou, 1996). For the purpose of full use of readily available human annotations for the same tasks, it is significant if such corpora can be used jointly. At first sight, a direct combination of multiple corpora is a way to this end. However, corpora created for the same NLP tasks are generally built by different organizations. Thus such corpora often follow different annotation standards and/or even different linguistic theories. We take CTB and TCT as a case study. Although both CTB and TCT are Chomskian-style treebanks, they have annotation divergences in at least two dimensions: a) CTB and TCT have dramatically different tag sets, including parts-of-speech and grammar labels, and the tags cannot be mapped one to one; b) CTB and TCT have distinct hierarchical structures. For example, the words "中国 (Chinese) 传统 (traditional) 文化 (culture)" are grouped as a flat noun phrase according to the CTB standard (right side in Fig. 1), but in TCT, the last two words are instead grouped together beforehand (left side in Fig. 1). The differences cause such treebanks of different annotations to be generally used independently. This paper is dedicated to solving the problem of how to use jointly multiple disparate treebanks for constituent syntactic parsing. Hereafter, treebanks of different annotations are

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1344–1352,
Beijing, August 2010

called *heterogeneous treebanks*, and correspondingly, the problem of syntactic parsing with heterogeneous treebanks is referred to as *heterogeneous parsing*.

Previous work on heterogeneous parsing is often based on treebank transformation (or treebank conversion) (Wang et al., 1994; Niu et al., 2009). The basic idea is to transform annotations of one treebank (source treebank) to fit the standard of another treebank (target treebank). Due to divergences of treebank annotations, such transformation is generally achieved in an indirect way by selecting transformation results from the output of a parser trained on the target treebank. A common property of all the work mentioned above is that transformation accuracy is heavily dependent on the performance of parsers trained on the target treebank. Sometimes transformation accuracy is not so satisfactory that techniques like instance pruning are needed in order to refine transformation results (Niu et al., 2009).

We claim there exists another way, interesting but less studied for heterogeneous parsing. The basic idea is that, although there are annotation divergences between heterogenous treebanks, actually we can also find consensus in annotations of bracket structures. Thus we would like to train parsers on individual heterogeneous treebanks and guide the parsers to gain output with consensus in bracket structures as much as possible when they are parsing the same sentences.

To realize this idea, we propose a generic collaborative decoding (or co-decoding) framework where decoders trained on heterogeneous treebanks can exchange consensus information between each other during the decoding phase. Theoretically the framework is able to incorporate a large number of treebanks and various functions that formalize consensus statistics.

Our contributions can be summarized: 1) we propose a co-decoding approach to directly utilizing heterogeneous treebanks; 2) we propose a novel function to measure parsing consensus between multiple decoders. We also conduct experiments on two Chinese treebanks: CTB and TCT. The results show that our approach achieves promising improvements over baseline systems which make no use of consensus information.

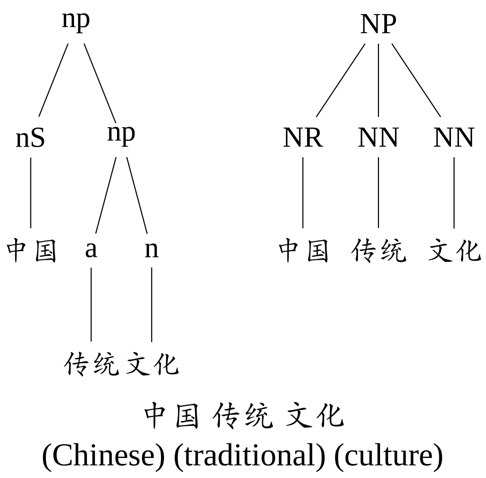


Figure 1: Example tree fragments with TCT (left) and CTB (right) annotations

2 Collaborative Decoding-based Heterogeneous Parsing

2.1 Motivation

This section describes the motivation to use co-decoding for heterogeneous parsing. We first use the example in Fig. 1 to illustrate what consensus information exists between heterogenous treebanks and why such information might help to improve parsing accuracy. This figure contains two partial parse trees corresponding to the words “中国 (Chinese) 传统 (traditional) 文化 (culture)”, annotated according to the TCT (left side) and CTB (right side) standards respectively. Despite the distinctions in tag sets and bracket structures, these parse trees actually have partial agreements in bracket structures. That is, not all bracket structures in the parse trees are different. Specifically put, although the internal structures of the parse trees are different, both CTB and TCT agree to take “中国 传统 文化” as a noun phrase. Motivated by this observation, we would like to guide parsers that are trained on CTB and TCT respectively to verify their output interactively by using consensus information implicitly contained in these treebanks. Better performance is expected when such information is considered.

A feasible framework to make use of consensus information is n-best combination (Henderson and Brill, 1999; Sagae and Lavie, 2006; Zhang et al., 2009; Fossum and Knight, 2009). In contrast to previous work on n-best combination where multiple parsers, say, Collins parser (Collins, 1999) and Berkeley parser (Petrov et al., 2006) are trained on the same training data, n-best combination for heterogeneous parsing is instead allowed to use either a single parser or multiple parsers which are trained on heterogeneous treebanks. Consensus information can be incorporated during the combination of the output (n-best list of full parse trees following distinct annotation standards) of individual parsers. However, despite the success of n-best combination methods, they suffer from the limited scope of n-best list. Taking this into account, we prefer to apply the co-decoding approach such that consensus information is expected to affect the entire procedure of searching hypothesis space.

2.2 System Overview

The idea of co-decoding is recently extensively studied in the literature of SMT (Li et al., 2009; Liu et al., 2009). As the name shows, co-decoding requires multiple decoders be combined and proceed collaboratively. As with n-best combination, there are at least two ways to build multiple decoders: we can either use multiple parsers trained on the same training data (use of diversity of models), or use a single parser on different training data (use of diversity of datasets) [1]. Both ways can build multiple decoders which are to be integrated into co-decoding. For the latter case, one method to get diverse training data is to use different portions of the same training set. In this study we extend the case to an extreme situation where heterogeneous treebanks are used to build multiple decoders.

Fig. 2 represents a basic flow chart of heterogeneous parsing via co-decoding. Note that here we discuss the case of co-decoding with only two decoders, but the framework is generic enough to integrate more than two decoders. For convenience of reference, we call a decoder without incorporating consensus information as *baseline decoder* and correspondingly refer to a decoder augmented with consensus information as *member decoder*. So the basic steps of co-decoding for heterogeneous parsing is to first build baseline decoders on heterogeneous treebanks and then use the baseline decoders to parse sentences with consensus information exchanged between each other.

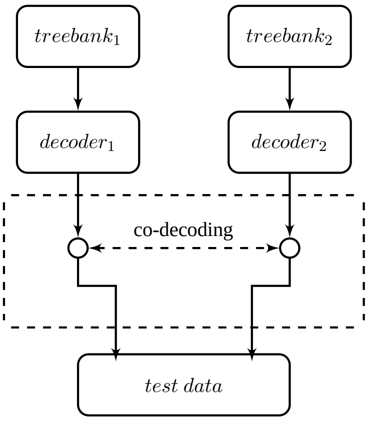


Figure 2: Basic flow chart of co-decoding

To complete co-decoding for heterogeneous parsing, three key components should be considered in the system:

- Co-decoding model. A co-decoder consists of multiple member decoders which are baseline decoders augmented with consensus information. Co-decoding model defines how baseline decoders and consensus information are correlated to get member decoders.

- Decoder coordination. Decoders in the co-decoding model cannot proceed independently but should have interactions between each other in order to exchange consensus information. A decoder coordination strategy decides on when, where, and how the interactions happen.

- Consensus-based score function. Consensus-based score functions formalize consensus information between member decoders. Taking time complexity into consideration, consensus statistics should be able to be computed efficiently.

[1] To make terminologies clear, we use *parser* as its regular sense, including training models (ex. Collins model 2) and parsing algorithms (ex. the CKY algorithm used in Collins parser), and we use *decoder* to represent parsing algorithms with specified parameter values

In the following subsections, we first present the generic co-decoding model and then describe in detail how member decoders collaborate. Finally we introduce a novel consensus-based score function which is used to quantify consensus information exchanged between member decoders.

2.3 Generic Co-decoding Model

The generic co-decoding model described here is also used in (Li et al., 2009) for co-decoding of machine translators. For a given sentence S, a parsing algorithm (decoder) seeks a parse tree T^* which is optimal in the sense that it maximizes some score function $F(T)$, as shown in Eq. 1.

$$T^* = \underset{Ts.t.S=yield(T)}{\arg\max} \quad F(T) \qquad (1)$$

where $Ts.t.S = yield(T)$ represents the set of parse trees that yield the input sentence S. For baseline decoders, the score function $F(T)$ is generally just the inside probability $P(T)$ [2] of a tree T, defined as the product of probabilities of grammar rules appearing in parse tree T: $\prod_{r \in R(T)} P(r)$. In the co-decoding framework, $F(T)$ is extended so as to integrate consensus-based score functions which measure consensus information between member decoders, as shown in Eq. 2.

$$F_m(T) = P_m(T) + \sum_{k,k \neq m}^{n} \Psi_k(H_k(S), T) \qquad (2)$$

We use d_k to denote the k_{th} decoder and use $H_k(S)$ to denote corresponding parsing hypothesis space of decoder d_k. Moreover, $P_m(T)$ is referred to as *baseline score* given by baseline decoders and $\Psi_k(H_k(S), T)$ is *consensus score* between decoders d_m and d_k, which is defined as a linear combination of consensus-based score functions, as shown in Eq. 3.

$$\Psi_k(H_k(S), T) = \sum_l \lambda_{k,l} f_{k,l}(H_k(S), T) \qquad (3)$$

where $f_{k,l}(H_k(S), T)$ represents a consensus-based score function between T and $H_k(S)$, and $\lambda_{k,l}$ is the corresponding weight. Index l ranges over all consensus-based score functions in Eq. 3. Theoretically we can define a variety of consensus-based score functions.

For the simplest case where there are only two member decoders and one consensus-based score function, Eq. 2 and Eq. 3 can be combined and simplified into the equation

$$F_i(T) = P_i(T) + \lambda_{1-i} f(H_{1-i}(S), T) \qquad (4)$$

where index i is set to the value of either 1 or 0. This simplified version is used in the experiments of this study.

2.4 Decoder Coordination

This subsection discusses the problem of decoder coordination. Note that although Eq. 2 is defined at sentence level, the co-decoding model actually should be applied to the parsing procedure of any subsequence (word span) of sentence S. So it is natural to render member decoders collaborate when they are processing the same word spans. To this end, we would like to adopt best-first CKY-style parsing algorithms as baseline decoders, since CKY-style decoders have the property that they process word spans in the ascending order of span sizes. Moreover, the hypotheses [3] spanning the same range of words are readily stacked together in a chart cell before CKY-style decoders move on to process other spans. Thus, member decoders can process the same word spans collaboratively from small ones to big ones until they finally complete parsing the entire sentence.

A second issue in Eq. 2 is that consensus-based score functions are dependent on hypothesis space $H_k(S)$. Unfortunately, the whole hypothesis space is not available most of the time. To address this issue, one practical method is to approximate $H_k(S)$ with a n-best hypothesis list. For best-first CKY parsing, we actually retain all unpruned partial hypotheses over the same span as the approximation. Hereafter, the approximation is denoted as $\hat{H}_k(S)$

Finally, we notice in Eq. 2 that consensus score

[2] Actually, the joint probability P(S,T) of sentence S and parse tree T is used, but we can prove that $P(S,T) = P(T)$.

[3] In the literature of syntactic parsing, especially in chart parsing, hypotheses is often called *edges*. This paper will continue to use the terminology *hypothesis* when no ambiguity exists.

$\Psi_k(H_k(S), T)$ and $H_k(S)$ form a circular dependency: searching for $H_k(S)$ requires both baseline score and consensus score; on the other hand, calculating consensus score needs $H_k(S)$ (its approximation in practice) to be known beforehand. Li et al. (2009) solves this dilemma with a bootstrapping method. It starts with seedy n-best lists generated by baseline decoders and then alternates between calculating consensus scores and updating n-best hypothesis lists. Such bootstrapping method is a natural choice to break down the circular dependency, but multi-pass re-decoding might dramatically reduce decoding efficiency. Actually, Li et al. (2009) restricts the iteration number to two in their experiments. In this paper, we instead use an alternative to the bootstrapping method. The process is described as follows.

1. In traditional best-first CKY-style parsing algorithms, hypotheses over the same word spans are grouped according to some criterion of hypothesis equivalence [4]. Among equivalent hypotheses, only a single optimal hypothesis is retained. In this paper, we instead keep top k of equivalent hypotheses in a data structure called *best-first cache*.

2. Use hypotheses in best-first caches to approximate $H_k(S)$, and calculate consensus score $\Psi_k(H_k(S), T)$ between decoders.

3. Use baseline score and consensus score to locally rerank hypotheses in best-first caches. Then remove hypotheses in caches except the top one hypothesis.

In this study, we choose the best-first CKY-style parsing algorithm used in Collins parser (Collins, 1999). Algorithm 1 extends this algorithm for co-decoding. The first two steps initialize baseline decoders and assign appropriate POS tags to sentence S^t. Since baseline decoders are built on heterogeneous treebanks, POS taggers corresponding to each baseline decoder are demanded, unless gold POS tags are provided. The third step is the core of the co-decoding algorithm. Here the *complete* procedure invokes baseline decoders to complete parsing on the span $[start, end]$ and generates $\hat{H}_k(s)$. The *co-decoding* procedure calculates consensus score and locally reranks hypotheses in best-first caches.

Algorithm 1 CKY-style Co-decoding

```
Argument: $d_k${the set of baseline decoders}
          $S^t${a sentence to be parsed}

Begin
Steps:
1. assign POS tags to sentence $S^t$
2. initialize baseline decoders $d_k$
3. for span from 2 to sentence_length do
       for start from 1 to (sentence_length-span+1) do
           end := (start + span - 1)
           for each base decoder $d_k$ do
               complete($d_k$, start, end)
           do co-decoding(start, end)
End

Subroutine:
    complete($d_k$, start, end): base decoder $d_k$ generates hypotheses over the span (begin.end), and fills in best-first caches.
    co-decoding(start, end): calculate consensus score and rerank hypotheses in best-first caches. The top 1 is chosen to be the best-first hypothesis.
```

2.5 Consensus-based Score Function

There are at least two feasible ways to measure consensus between constituency parse trees. By viewing parse trees from diverse perspectives, we can either use functions on bracket structures of parse trees, as in (Wang et al., 1994), or use functions on head-dependent relations by first transforming constituency trees into dependency trees, as in (Niu et al., 2009). Although the co-decoding model is generic enough to integrate various consensus-based score functions in a uniform way, this paper only uses a bracket structure-based function.

As mentioned above, the function proposed in (Wang et al., 1994) is based on bracket structures. Unfortunately, that function is not applicable in the situation of this paper. The reason is that, the function in (Wang et al., 1994) is defined to work on two parse trees, but this paper instead needs a function on a tree T and a set of trees (the approximation $\hat{H}_k(S)$). To this end, we first introduce the concept of *constituent set (CS)* of a parse tree. Conceptually, CS of a parse tree is a set of word spans corresponding to all the sub-

[4] the simplest criterion of equivalence is whether hypotheses have the same grammar labels.

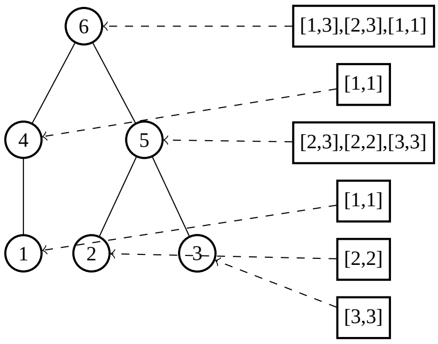


Figure 3: Constituent set of a synthetic parse tree

trees of the tree, as illustrated in Fig. 3. For example, the constituent set of the tree rooted at node 6 has three elements: $[1, 1]$, $[1, 3]$, and $[1, 2]$. For $\hat{H}_k(S)$, the constituent set is defined as the union of constituent sets of all elements it contains.

$$CS(\hat{H}_k(S)) = \bigcup_{T\in\hat{H}_k(S)} CS(T)$$

In practice, we need to cut off elements in $CS(\hat{H}_k(S))$ in order to retain most confident word spans.

With the concept of constituent set, a consensus-based score function on T and $\hat{H}_k(S)$ can be defined as follows.

$$f(\hat{H}_k(S), T) = \frac{\sum_{c\in CS(T)} I(c, CS(\hat{H}_k(S)))}{|CS(T)|} \qquad (5)$$

where $I(c, CS(\hat{H}_k(S)))$ is an indicator function which returns one if $c \in CS(T)$ is compatible with all the elements in $CS(\hat{H}_k(S))$, zero otherwise. Two spans, $[a, b]$ and $[i, j]$ are said to be compatible if they satisfy one of the following conditions: 1) $i > b$; 2) $a > j$; 3) $a \leq i \leq b$ and $j \leq b$; 4) $i \leq a \leq j$ and $b \leq j$. Fig 4 uses two example to illustrate the concept of compatibility.

3 Experiments

3.1 Data and Performance Metric

The most recent version of the CTB corpus, CTB 6.0 and the CIPS ParsEval data are used as heterogeneous treebanks in the experiments. Following the split utilized in (Huang et al., 2007), we divided the dataset into blocks of 10 files. For each block, the first file was added to the CTB development data, the second file was added to the CTB testing data, and the remaining 8 files were added to the CTB training data. For the sake of parsing efficiency, we randomly sampled 1,000 sentences of no more than 40 words from the CTB test set.

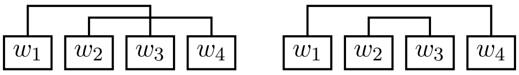


Figure 4: left) two spans conflict; right) two spans are compatible

CTB-Partitions	Train	Dev	Test
#Sentences	22,724	2,855	1,000
#Words	627,833	78,653	25,100
Ave-Length	30.1	30.0	20.3
TCT-Partitions	Train	Dev	Test
#Sentences	32,771	N/A	1,000
#Words	354,767	N/A	10,400
Ave-Length	10.6	N/A	10.4

Table 1: Basic statistics on the CTB and TCT data

CIPS-ParsEval data is publicly available for the first Chinese syntactic parsing competition, CIPS-ParsEval 2009. Compared to CTB, sentences in CIPS-ParsEval data are much shorter in length. We removed sentences which have words less than three. CIPS-ParsEval test set has 7,995 sentences after sentence pruning. As with the CTB test set, we randomly sampled 1,000 sentences for evaluating co-decoding performance. Since CIPS-ParsEval data is actually a portion of the TCT corpus, for convenience of reference, we will refer to CIPS-ParsEval data as TCT in the following sections. Table 1 contains statistics on CTB and TCT.

The two training sets are used individually to build baseline decoders. With regard to the test sets, each sentence in the test sets should have two kinds of POS tags, according to the CTB and TCT standards respectively. To this end, we applied a HMM-based method for POS annotation transformation (Zhu and Zhu, 2009). During the POS transformation, the divergences of word segmentation are omitted.

For all experiments, *bracketing F1* is used as the performance metric, provided by *EVALB* [5].

[5] http://nlp.cs.nyu.edu/evalb

3.2 Baseline Decoders

As already mentioned above, we apply Collins parser in this paper. Specifically speaking, two CKY-style baseline decoders to participate co-decoding are built on CTB and TCT respectively with Collins model two. For the CTB-based decoder, we use the CTB training data with slight modifications: we replaced POS tags of punctuations with specific punctuation symbols.

To get the TCT-based decoder, we made following modifications. Firstly, TCT is available with manually annotated head indices for all the constituents in parse trees. For example, a grammar label, say, np-1, means that the constituent is a noun phrase with the second child being its head child. In order to relax context independence assumptions made in PCFG, we appended head indices to grammar labels to get new labels, for example $np1$. Secondly, since Collins parser is a lexicalized parser, head rules specific to the TCT corpus were manually created, which are used together with readily available head indices. Such adaptation is also used in (Chen et al., 2009);

3.3 Parsing Results

We conduct experiments on both CTB and TCT test sets. Two parameters need to be set: the cutoff threshold for constructing constituent set of $\hat{H}_k(S)$ and the weight λ [6] of consensus score in Eq. 4. We tuned the parameters on the CTB development set and finally set them to 5 and 20 respectively in the experiments. Table 2 presents bracketing F1 scores of baseline systems and the co-decoding approach. Here, the row of *baseline* represents the performance of individual baseline decoders, and the comparison of baseline and co-decoding on a test set, say CTB, demonstrates how much boosting the other side, say TCT, can supply. For the co-decoding approach, the size of best-first cache is set to 5 which achieves the best result among the cache sizes we have experimented.

As the results show, co-decoding achieves promising improvements over baseline systems on both test sets. Interestingly, we see that the improvement on the TCT test set is larger than that on the CTB test set. In general, a relatively strong decoder can improve co-decoding performance more than a relatively weak decoder does. At the first sight, the TCT-based decoder seems to have better performance than the CTB-based decoder. But if taking sentence length into consideration, we can find that the TCT-based decoder is actually relatively weak. Table 3 shows the performance of the CTB-based decoder on short sentences.

Test Set	CTB	TCT
Baseline	79.82	81.02
Co-decoding	80.33	81.77

Table 2: Baseline and Co-decoding on the CTB and TCT test sets

3.4 Analysis

Fig. 5 shows the bracketing F1 on the CTB test set at different settings of the best-first cache size C. F1 scores reach the peak before C increases to 6. As a result, we set C to 5 in all our experiments.

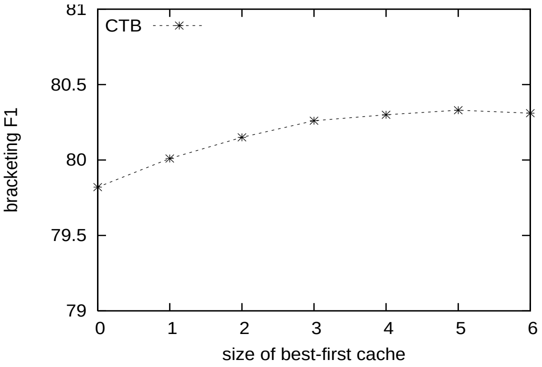


Figure 5: Bracketing F1 with varying best-first cache size

To evaluate the effect of sentence length on co-decoding, Table 3 presents F1 scores on portions of the CTB test set, partitioned according to sentence length. From the results we can see that co-decoding performs better on long sentences. One possible reason is that member decoders have more consensus on big spans. Taking this observation into consideration, one enhancement to the co-decoding approach is to enable co-decoding only on long sentences. This way, parsing ef-

[6] We use the same λ for both member decoders.

Partitions	[0,10]	(10,20]	(20,30]	(30,40]
# Sentence	276	254	266	204
Ave-Length	6.07	15.64	25.43	35.20
Baseline	92.83	84.34	78.98	76.69
Co-decoding	92.84	84.36	79.43	77.65

Table 3: Effect of sentence length on co-decoding performance

ficiency of co-decoding can be improved. It is worth emphasizing that co-decoding is still helpful for parsers whose performance on short sentences is not satisfactory, as shown in Table 2.

Another interesting analysis is to check how many parsing results are affected by co-decoding, compared to baseline decoders. Table 4 shows the statistics.

Test Set	# All	# Improved	# Decreased
CTB	1000	225	109
TCT	1000	263	92

Table 4: Statistics on sentences of test data

As the table shows, although overall accuracy is increased, we find that on some sentences, co-decoding instead worsens parsing accuracy. In order to get insights on error sources, we manually analyzed 20 sentences on which co-decoding achieves negative results. We find a large portion (14 of 20) of sentences are short sentences (of words less than 20). Actually, due to high accuracy of the CTB-based decoder on short sentences, co-decoding is indifferent when this decoder is processing short sentences. And we also find that some errors are derived from differences in annotation standards. Fortunately, the divergence of annotations mainly exists in relatively small spans. So one solution to the problem is to enable co-decoding on relatively big spans. These will be done in our future work.

4 Related Work

4.1 System Combination

In the literature of syntactic parsing, n-best combination methods include parse selection, constituent recombination, production recombination, and n-best reranking. Henderson and Brill (1999) performs parse selection by maximizing the expected precision of selected parse with respect to the set of parses to be combined. Sagae and Lavie (2006) proposes to recombine constituents from the output of individual parsers. More recently, Fossum and Knight (2009) studies a combination method at production level. Zhang et al. (2009) reranks n-best list of one parser with scores derived from another parser.

Compared to n-best combination, co-decoding (Li et al., 2009; Liu et al., 2009) combines systems during decoding phase. Theoretically, system combination during decoding phase helps decoders to select better approximation to hypothesis space, since pruning is practically unavoidable. To the best of our knowledge, co-decoding methods have not been applied to syntactic parsing.

4.2 Treebank Transformation

The focus of this study is heterogeneous parsing. Previous work on this challenge is generally based on treebank transformation. Wang et al. (1994) describes a method for transformation between constituency treebanks. The basic idea is to train a parser on a target treebank and generate a n-best list for each sentence in source treebank(s). Then, a matching metric which is a function on the number of the same word spans between two trees is defined to select a best parse from each n-best list. Niu et al. (2009) applies a closely similar framework as with (Wang et al., 1994) to transform a dependency treebank to a constituency one.

5 Conclusions

This paper proposed a co-decoding approach to the challenge of heterogeneous parsing. Compared to previous work on this challenge, co-decoding is able to directly utilize heterogeneous treebanks by incorporating consensus information between partial output of individual parsers during the decoding phase. Experiments demonstrate the effectiveness of the co-decoding approach, especially the effectiveness on long sentences.

Acknowledgments

This work was supported in part by the National Science Foundation of China (60873091). We would like to thank our anonymous reviewers for their comments.

References

Banko, Michele and Eric Brill. 2001. *Scaling to very very large corpora for natural language disambiguation*. In Proc. of ACL 2001, pages 26-33.

Chen, Xiao, Changning Huang, Mu Li, and Chunyu Kit. 2009. *Better Parser Combination*. Technique Report of CIPS-ParsEval 2009.

Collins, Michael. 1999. *Head-driven statistical models for natural language parsing*. Ph.D. thesis, University of Pennsylvania.

Charniak, Eugene. 2000. *A maximum-entropy-inspired parser*. In Proc. of NAACL 2000, pages 132-139.

Fossum, Victoria and Kevin Knight. 2009. *Combining constituent parsers*. In Proc. of NAACL 2009, pages 253-256.

Henderson, John and Eric Brill. 1999. *Exploiting diversity in natural language processing*. In Proc. of SIGDAT-EMNLP 1999, pages 187-194.

Huang, Zhongqiang, Mary P. Harper, and Wen Wang. 2007. *Mandarin part-of-speech tagging and discriminative reranking*. In Proc. of EMNLP-CoNLL 2007, pages 1093-1102.

Li, Mu, Nan Duan, Dongdong Zhang, Chi-Ho Li, and Ming Zhou. 2009. *Collaborative decoding: partial hypothesis re-ranking using trnaslationconsensus between decoders*. In Proc. of ACL 2009, pages 585-592.

Liu, Yang, Haitao Mi, Yang Feng, and Qun Liu. 2009. *Joint Decoding with Multiple Translation Models*. In Proc. of ACL 2009, pages 576-584.

Niu, Zheng-Yu, Haifeng Wang, Hua Wu. 2009. *Exploiting heterogeneous treebanks for parsing*. In Proc. of ACL 2009, pages 46-54.

Petrov, Slav, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. *Learning accurate, compact, and interpretable tree annotation*. In Proc. of COLING-ACL 2006, pages 433-440.

Sage, Kenji and Alon Lavie. 2006. *Parser combination by reparsing*. In Proc. of NAACL 2006, pages 129-132.

Xue, Nianwen, Fu dong Chiou, and Martha Palmer. 2002. *Building a large-scale Annotated Chinese corpus*. In Proc. of COLING 2002, pages 1-8.

Wang, Jong-Nae, Jing-Shin Chang, and Keh-Yih Su. 1994. *An automatic treebank conversion algorithm for corpus sharing*. In Proc. of ACL 1994, pages 248-254.

Zhang, Hui, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. *K-best combination of syntactic parsers*. In Proc. of EMNLP 2009, pages 1552-1560.

Zhou, Qiang. 1996. *Phrase bracketing and annotating on Chinese language corpus. (in Chinese)* Ph.D. thesis, Beijing University.

Zhu, Muhua and Jingbo Zhu. 2009. *Label Correspondence Learning for Part-of-Speech Annotation Transformation*. In Proc. of CIKM 2009, pages 1461-1464.

A Monolingual Tree-based Translation Model for Sentence Simplification*

Zhemin Zhu[1]
Department of Computer Science
Technische Universität Darmstadt

Delphine Bernhard[2]
LIMSI-CNRS

Iryna Gurevych[1]
Department of Computer Science
Technische Universität Darmstadt

[1]`http://www.ukp.tu-darmstadt.de` [2]`delphine.bernhard@limsi.fr`

Abstract

In this paper, we consider sentence simplification as a special form of translation with the complex sentence as the source and the simple sentence as the target. We propose a Tree-based Simplification Model (TSM), which, to our knowledge, is the first statistical simplification model covering splitting, dropping, reordering and substitution integrally. We also describe an efficient method to train our model with a large-scale parallel dataset obtained from the Wikipedia and Simple Wikipedia. The evaluation shows that our model achieves better readability scores than a set of baseline systems.

1 Introduction

Sentence simplification transforms long and difficult sentences into shorter and more readable ones. This helps humans read texts more easily and faster. Reading assistance is thus an important application of sentence simplification, especially for people with reading disabilities (Carroll et al., 1999; Inui et al., 2003), low-literacy readers (Watanabe et al., 2009), or non-native speakers (Siddharthan, 2002).

Not only human readers but also NLP applications can benefit from sentence simplification. The original motivation for sentence simplification is using it as a preprocessor to facilitate parsing or translation tasks (Chandrasekar et al., 1996). Complex sentences are considered as stumbling blocks for such systems. More recently, sentence simplification has also been shown helpful for summarization (Knight and Marcu, 2000), sentence fusion (Filippova and Strube, 2008b), semantic role labeling (Vickrey and Koller, 2008), question generation (Heilman and Smith, 2009), paraphrase generation (Zhao et al., 2009) and biomedical information extraction (Jonnalagadda and Gonzalez, 2009).

At sentence level, reading difficulty stems either from lexical or syntactic complexity. Sentence simplification can therefore be classified into two types: *lexical simplification* and *syntactic simplification* (Carroll et al., 1999). These two types of simplification can be further implemented by a set of simplification operations. *Splitting*, *dropping*, *reordering*, and *substitution* are widely accepted as important simplification operations. The splitting operation splits a long sentence into several shorter sentences to decrease the complexity of the long sentence. The dropping operation further removes unimportant parts of a sentence to make it more concise. The reordering operation interchanges the order of the split sentences (Siddharthan, 2006) or parts in a sentence (Watanabe et al., 2009). Finally, the substitution operation replaces difficult phrases or words with their simpler synonyms.

In most cases, different simplification operations happen simultaneously. It is therefore necessary to consider the simplification process as a combination of different operations and treat them as a whole. However, most of the existing models only consider one of these operations. Siddharthan (2006) and Petersen and Ostendorf (2007) focus on sentence splitting, while sentence compression systems (Filippova and Strube, 2008a) mainly use the dropping operation. As far as lexical simplification is concerned, word substitution is usually done by selecting simpler synonyms from Wordnet based on word frequency (Carroll et al., 1999).

In this paper, we propose a sentence simplification model by tree transformation which is based

* This work has been supported by the Emmy Noether Program of the German Research Foundation (DFG) under the grant No. GU 798/3-1, and by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under the grant No. I/82806.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1353–1361,
Beijing, August 2010

on techniques from statistical machine translation (SMT) (Yamada and Knight, 2001; Yamada and Knight, 2002; Graehl et al., 2008). Our model integrally covers splitting, dropping, reordering and phrase/word substitution. The parameters of our model can be efficiently learned from complex-simple parallel datasets. The transformation from a complex sentence to a simple sentence is conducted by applying a sequence of simplification operations. An expectation maximization (EM) algorithm is used to iteratively train our model. We also propose a method based on monolingual word mapping which speeds up the training process significantly. Finally, a decoder is designed to generate the simplified sentences using a greedy strategy and integrates language models.

In order to train our model, we further compile a large-scale complex-simple parallel dataset (PWKP) from Simple English Wikipedia[1] and English Wikipedia[2], as such datasets are rare.

We organize the remainder of the paper as follows: Section 2 describes the PWKP dataset. Section 3 presents our TSM model. Sections 4 and 5 are devoted to training and decoding, respectively. Section 6 details the evaluation. The conclusions follow in the final section.

2 Wikipedia Dataset: PWKP

We collected a paired dataset from the English Wikipedia and Simple English Wikipedia. The targeted audience of Simple Wikipedia includes "children and adults who are learning English language". The authors are requested to "use easy words and short sentences" to compose articles. We processed the dataset as follows:

Article Pairing 65,133 articles from Simple Wikipedia[3] and Wikipedia[4] were paired by following the "language link" using the dump files in Wikimedia.[5] Administration articles were further removed.

Plain Text Extraction We use JWPL (Zesch et al., 2008) to extract plain texts from Wikipedia articles by removing specific Wiki tags.

Pre-processing including sentence boundary detection and tokenization with the Stanford Parser package (Klein and Manning, 2003), and lemmatization with the TreeTagger (Schmid, 1994).

Monolingual Sentence Alignment As we need a parallel dataset aligned at the sentence level, we further applied monolingual sentence alignment on the article pairs. In order to achieve the best sentence alignment on our dataset, we tested three similarity measures: (i) sentence-level TF*IDF (Nelken and Shieber, 2006), (ii) word overlap (Barzilay and Elhadad, 2003) and (iii) word-based maximum edit distance (MED) (Levenshtein, 1966) with costs of insertion, deletion and substitution set to 1. To evaluate their performance we manually annotated 120 sentence pairs from the article pairs. Tab. 1 reports the precision and recall of these three measures. We manually adjusted the similarity threshold to obtain a recall value as close as possible to 55.8% which was previously adopted by Nelken and Shieber (2006).

Similarity	Precision	Recall
TF*IDF	91.3%	55.4%
Word Overlap	50.5%	55.1%
MED	13.9%	54.7%

Table 1: Monolingual Sentence Alignment

The results in Tab. 1 show that sentence-level TF*IDF clearly outperforms the other two measures, which is consistent with the results reported by Nelken and Shieber (2006). We henceforth chose sentence-level TF*IDF to align our dataset.

As shown in Tab. 2, PWKP contains more than 108k sentence pairs. The sentences from Wikipedia and Simple Wikipedia are considered as "complex" and "simple" respectively. Both the average sentence length and average token length in Simple Wikipedia are shorter than those in Wikipedia, which is in compliance with the purpose of Simple Wikipedia.

Avg. Sen. Len		Avg. Tok. Len		#Sen.Pairs
complex	simple	complex	simple	-
25.01	20.87	5.06	4.89	108,016

Table 2: Statistics for the PWKP dataset

In order to account for sentence splitting, we allow 1 to n sentence alignment to map one complex sentence to several simple sentences. We first perform 1 to 1 mapping with sentence-level TF*IDF and then combine the pairs with the same complex sentence and adjacent simple sentences.

3 The Simplification Model: TSM

We apply the following simplification operations to the parse tree of a complex sentence: splitting,

[1] http://simple.wikipedia.org

[2] http://en.wikipedia.org

[3] As of Aug 17th, 2009

[4] As of Aug 22nd, 2009

[5] http://download.wikimedia.org

dropping, reordering and substitution. In this section, we use a running example to illustrate this process. c is the complex sentence to be simplified in our example. Fig. 1 shows the parse tree of c (we skip the POS level).

c: ***August was the sixth month in the ancient Roman calendar which started in 735BC.***

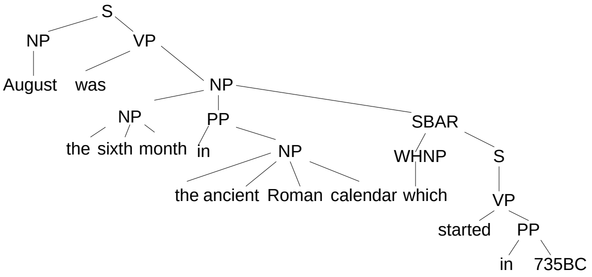


Figure 1: Parse Tree of c

3.1 Splitting

The first operation is sentence splitting, which we further decompose into two subtasks: (i) *segmentation*, which decides where and whether to split a sentence and (ii) *completion*, which makes the new split sentences complete.

First, we decide where we can split a sentence. In our model, the splitting point is judged by the syntactic constituent of the split boundary word in the complex sentence. The decision whether a sentence should be split is based on the length of the complex sentence. The features used in the *segmentation* step are shown in Tab. 3.

Word	Constituent	iLength	isSplit	Prob.
"which"	SBAR	1	true	0.0016
"which"	SBAR	1	false	0.9984
"which"	SBAR	2	true	0.0835
"which"	SBAR	2	false	0.9165

Table 3: Segmentation Feature Table (SFT)

Actually, we do not use the direct constituent of a word in the parse tree. In our example, the direct constituent of the word "which" is "WHNP". Instead, we use Alg. 1 to calculate the constituent of a word. Alg. 1 returns "SBAR" as the adjusted constituent for "which". Moreover, directly using the length of the complex sentence is affected by the data sparseness problem. Instead, we use $\mathrm{iLength}$ as the feature which is calculated as $\mathrm{iLength} = \mathrm{ceiling}(\frac{\mathrm{comLength}}{\mathrm{avgSimLength}})$, where $\mathrm{comLength}$ is the length of the complex sentence and $\mathrm{avgSimLength}$ is the average length of simple sentences in the training dataset. The "Prob." column shows the probabilities obtained after training on our dataset.

Algorithm 1 adjustConstituent($word$, $tree$)

```
$constituent \leftarrow word.father$;
$father \leftarrow constituent.father$;
**while** $father \neq NULL$ AND $constituent$ is the most left child of $father$ **do**
    $constituent \leftarrow father$;
    $father \leftarrow father.father$;
**end while**
**return** $constituent$;
```

In our model, one complex sentence can be split into two or more sentences. Since many splitting operations are possible, we need to select the most likely one. The probability of a segmentation operation is calculated as:

$$P(seg|c) = \prod_{w:c} SFT(w|c) \tag{1}$$

where w is a word in the complex sentence c and $SFT(w|c)$ is the probability of the word w in the Segmentation Feature Table (SFT); Fig. 2 shows a possible segmentation result of our example.

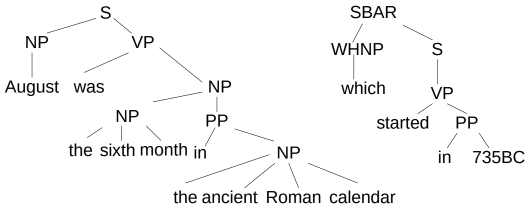


Figure 2: Segmentation

The second step is *completion*. In this step, we try to make the split sentences complete and grammatical. In our example, to make the second sentence "which started in 735BC" complete and grammatical we should first drop the border word "which" and then copy the dependent NP "the ancient Roman calendar" to the left of "started" to obtain the complete sentence "the ancient Roman calendar started in 735BC". In our model, whether the border word should be dropped or retained depends on two features of the border word: the direct constituent of the word and the word itself, as shown in Tab. 4.

Const.	Word	isDropped	Prob.
WHNP	which	True	1.0
WHNP	which	False	Prob.Min

Table 4: Border Drop Feature Table (BDFT)

In order to copy the necessary parts to complete the new sentences, we must decide which parts should be copied and where to put these parts in the new sentences. In our model, this is judged by two features: the dependency relation and the constituent. We use the Stanford Parser for parsing the dependencies. In our example, the dependency relation between “calendar” in the complex sentence and the verb “started” in the second split sentence is “gov_nsubj”.[6] The direct constituent of “started” is “VP” and the word “calendar” should be put on the “left” of “started”, see Tab. 5.

Dep.	Const.	isCopied	Pos.	Prob.
gov_nsubj	VP(VBD)	True	left	0.9000
gov_nsubj	VP(VBD)	True	right	0.0994
gov_nsubj	VP(VBD)	False	-	0.0006

Table 5: Copy Feature Table (CFT)

For dependent NPs, we copy the whole NP phrase rather than only the head noun.[7] In our example, we copy the whole NP phrase “the ancient Roman calendar” to the new position rather than only the word “calendar”. The probability of a completion operation can be calculated as

$$P(com|seg) = \prod_{bw:s} BDFT(bw|s) \prod_{w:s} \prod_{dep:w} CFT(dep).$$

where s are the split sentences, bw is a border word in s, w is a word in s, dep is a dependency of w which is out of the scope of s. Fig. 3 shows the most likely result of the completion operation for our example.

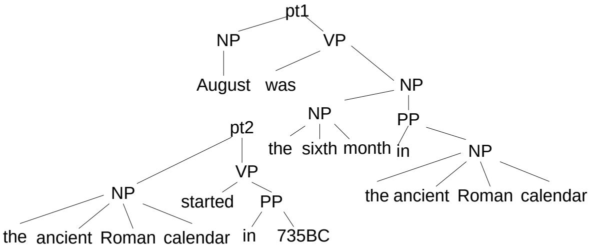


Figure 3: Completion

3.2 Dropping and Reordering

We first apply dropping and then reordering to each non-terminal node in the parse tree from top to bottom. We use the same features for both dropping and reordering: the node’s direct constituent and its children’s constituents pattern, see Tab. 6 and Tab. 7.

Constituent	Children	Drop	Prob.
NP	DT_JJ_NNP_NN	1101	7.66E-4
NP	DT_JJ_NNP_NN	0001	1.26E-7

Table 6: Dropping Feature Table (DFT)

Constituent	Children	Reorder	Prob.
NP	DT_JJ_NN	012	0.8303
NP	DT_JJ_NN	210	0.0039

Table 7: Reordering Feature Table (RFT)

The bits ‘1’ and ‘0’ in the “Drop” column indicate whether the corresponding constituent is retained or dropped. The number in the “Reorder” column represents the new order for the children. The probabilities of the dropping and reordering operations can be calculated as Equ. 2 and Equ. 3.

$$P(dp|node) = DFT(node) \qquad (2)$$

$$P(ro|node) = RFT(node) \qquad (3)$$

In our example, one of the possible results is dropping the NNP “Roman”, as shown in Fig. 4.

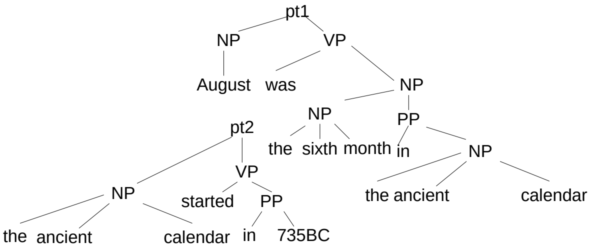


Figure 4: Dropping & Reordering

3.3 Substitution

3.3.1 Word Substitution

Word substitution only happens on the terminal nodes of the parse tree. In our model, the conditioning features include the original word and the substitution. The substitution for a word can be another word or a multi-word expression (see Tab. 8). The probability of a word substitution operation can be calculated as $P(sub|w) = SubFT(Substitution|Origin)$.

Origin	Substitution	Prob.
ancient	ancient	0.963
ancient	old	0.0183
ancient	than transport	1.83E-102
old	ancient	0.005

Table 8: Substitution Feature Table (SubFT)

3.3.2 Phrase Substitution

Phrase substitution happens on the non-terminal nodes and uses the same conditioning features as word substitution. The “Origin” consists of the leaves of the subtree rooted at the node. When we apply phrase substitution on a non-terminal node, then any simplification operation (including dropping, reordering and substitution) cannot happen on its descendants any more

[6] With Stanford Parser, “which” is a referent of “calender” and the nsubj of “started”. “calender” thus can be considered to be the nsubj of “started” with “started” as the governor.

[7] The copied NP phrase can be further simplified in the following steps.

because when a node has been replaced then its descendants are no longer existing. Therefore, for each non-terminal node we must decide whether a substitution should take place at this node or at its descendants. We perform substitution for a non-terminal $node$ if the following constraint is met:

$$Max(SubFT(*|node)) \geq \prod_{ch:node} Max(SubFT(*|ch)).$$

where ch is a child of the $node$. "$*$" can be any substitution in the SubFT. The probability of the phrase substitution is calculated as $P(sub|node) = SubFT(Substitution|Origin)$. Fig. 5 shows one of the possible substitution results for our example where "ancient" is replaced by "old".

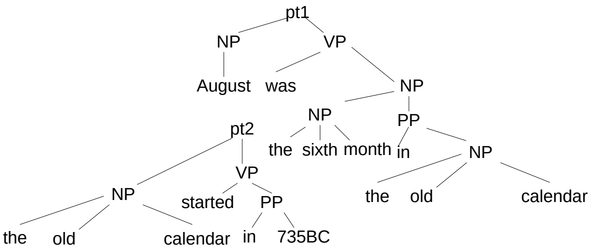


Figure 5: Substitution

As a result of all the simplification operations, we obtain the following two sentences: $s1 = Str(pt1)$=*"August was the sixth month in the old calendar."* and $s2 = Str(pt2)$=*"The old calendar started in 735BC."*

3.4 The Probabilistic Model

Our model can be formalized as a direct translation model from complex to simple $P(s|c)$ multiplied by a language model $P(s)$ as shown in Equ. 4.

$$s = \underset{s}{\operatorname{argmax}}\, P(s|c)P(s) \tag{4}$$

We combine the parts described in the previous sections to get the direct translation model:

$$P(s|c) = \sum_{\theta:Str(\theta(c))=s} (P(seg|c)P(com|seg) \tag{5}$$

$$\prod_{node} P(dp|node)P(ro|node)P(sub|node)$$

$$\prod_{w} (sub|w)).$$

where θ is a sequence of simplification operations and $Str(\theta(c))$ corresponds to the leaves of a simplified tree. There can be many sequences of operations that result in the same simplified sentence and we sum up all of their probabilities.

4 Training

In this section, we describe how we train the probabilities in the tables. Following the work of Yamada and Knight (2001), we train our model by maximizing $P(s|c)$ over the training corpus with the EM algorithm described in Alg. 2, using a constructed graph structure. We develop the Training Tree (Fig. 6) to calculate $P(s|c)$. $P(s|c)$ is equal to the inside probability of the root in the Training Tree. Alg. 3 and Alg. 4 are used to calculate the inside and outside probabilities. We refer readers to Yamada and Knight (2001) for more details.

Algorithm 2 EM Training ($dataset$)

```
Initialize all probability tables using the uniform distribution;
for several iterations do
    reset all $cnt$ = 0;
    for each sentence pair $< c, s >$ in $dataset$ do
        $tt$ = buildTrainingTree($< c, s >$);
        calcInsideProb($tt$);
        calcOutsideProb($tt$);
        update $cnt$ for each conditioning $feature$ in each $node$ of $tt$: $cnt = cnt + node.insideProb * node.outsideProb/root.insideProb$;
    end for
    updateProbability();
end for
```

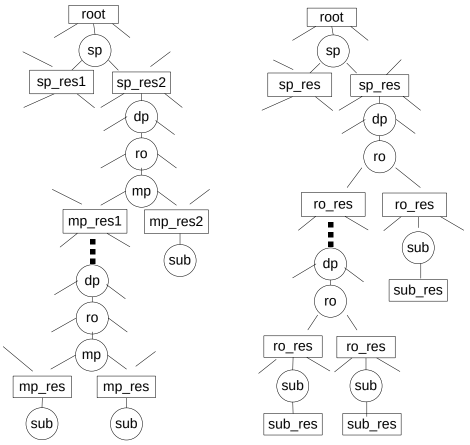


Figure 6: Training Tree (Left) and Decoding Tree (Right)

We illustrate the construction of the training tree with our running example. There are two kinds of nodes in the training tree: data nodes in rectangles and operation nodes in circles. Data nodes contain data and operation nodes execute operations. The training is a supervised learning process with the parse tree of c as input and the two strings $s1$ and $s2$ as the desired output. $root$ stores the parse tree of c and also $s1$ and $s2$. sp, ro, mp and sub are splitting, reordering, mapping and substitution operations. sp_res and mp_res store the results of sp and mp. In our example, sp splits the parse tree into two parse trees $pt1$ and $pt2$ (Fig. 3). sp_res1 contains $pt1$ and $s1$. sp_res2 contains $pt2$ and $s2$. Then dp, ro and mp are iteratively applied to each non-terminal node at each level of $pt1$ and $pt2$ from top to down. This process continues until the terminal nodes are reached or is stopped by a sub node. The function of mp operation is similar to the word mapping operation in the string-based machine translation. It maps substrings in the complex sentence which are dominated by the children of the current node to proper substrings in the simple sentences.

Speeding Up The example above is only one of the possible paths. We try all of the promising paths in training. Promising paths are the paths which are likely to succeed in transforming the parse tree of c into $s1$ and $s2$. We select the promising candidates using monolingual word mapping as shown in Fig. 7. In this example, only the word "which" can be a promising candidate for splitting. We can select the promising candidates for the dropping, reordering and mapping operations similarly. With this improvement, we can train on the PWKP dataset within 1 hour excluding the parsing time taken by the Stanford Parser.

We initialize the probabilities with the uniform distribution. The binary features, such as SFT and BDFT, are assigned the initial value of 0.5. For DFT and RFT, the initial probability is $\frac{1}{N!}$, where N is the number of the children. CFT is initialized as 0.25. SubFT is initialized as 1.0 for any substitution at the first iteration. After each iteration, the $updateProbability$ function recalculates these probabilities based on the cnt for each feature.

Algorithm 3 calcInsideProb (TrainingTree tt)

```
for each $node$ from $level = N$ to $root$ of $tt$ do
    if $node$ is a $sub$ node then
        $node.insideProb$ = $P(sub|node)$;
    else if $node$ is a $mp$ OR $sp$ node then
        $node.insideProb$ = $\prod_{child} child.insideProb$;
    else
        $node.insideProb$ = $\sum_{child} child.insideProb$;
    end if
end for
```

Algorithm 4 calcOutsideProb (TrainingTree tt)

```
for each $node$ from $root$ to $level = N$ of $tt$ do
    if $node$ is the $root$ then
        $node.outsideProb$ = 1.0;
    else if $node$ is a $sp\_res$ OR $mp\_res$ node then
        {COMMENT: $father$ are the fathers of the current $node$, $sibling$ are the children of $father$ excluding the current $node$}
        $node.outsideProb = \sum_{father} father.outsideProb * \prod_{sibling} sibling.insideProb$;
    else if $node$ is a $mp$ node then
        $node.outsideProb = father.outsideProb * 1.0$;
    else if $node$ is a $sp$, $ro$, $dp$ or $sub$ node then
        $node.outsideProb = father.outsideProb * P(sp$ or $ro$ or $dp$ or $sub|node)$;
    end if
end for
```

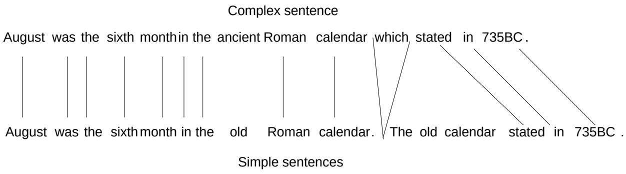

Complex sentence
August was the sixth month in the ancient Roman calendar which stated in 735BC .
August was the sixth month in the old Roman calendar. The old calendar stated in 735BC .
Simple sentences

Figure 7: Monolingual Word Mapping

5 Decoding

For decoding, we construct the decoding tree (Fig. 6) similarly to the construction of the training tree. The decoding tree does not have mp operations and there can be more than one sub nodes attached to a single ro_res. The $root$ contains the parse tree of the complex sentence. Due to space limitations, we cannot provide all the details of the decoder.

We calculate the inside probability and outside probability for each node in the decoding tree. When we simplify a complex sentence, we start from the root and greedily select the branch with the highest outside probability. For the substitution operation, we also integrate a trigram language model to make the generated sentences more fluent. We train the language model with SRILM (Stolcke, 2002). All the articles from the Simple Wikipedia are used as the training corpus, amounting to about 54 MB.

6 Evaluation

Our evaluation dataset consists of 100 complex sentences and 131 parallel simple sentences from PWKP. They have not been used for training. Four baseline systems are compared in our evaluation. The first is Moses which is a state of the art SMT system widely used as a baseline in MT community. Obviously, the purpose of Moses is cross-lingual translation rather than monolingual simplification. The goal of our comparison is therefore to assess how well a standard SMT system may perform simplification when fed with a proper training dataset. We train Moses with the same part of PWKP as our model. The second baseline system is a sentence compression system (Filippova and Strube, 2008a) whose demo system is available online.[8] As the compression system can only perform dropping, we further extend it to our third and fourth baseline systems, in order to make a reasonable comparison. In our third baseline system, we substitute the words in the output of the compression system with their simpler synonyms. This is done by looking up the synonyms in Wordnet and selecting the most frequent synonym for replacement. The word frequency is counted using the articles from Simple Wikipedia. The fourth system performs sentence splitting on the output of the third system. This is simply done by splitting the sentences at "and", "or", "but", "which", "who" and "that", and discarding the border words. In total, there are 5 systems in our evaluation: *Moses*, the MT system; *C*, the compression system; *CS*, the compression+substitution system; *CSS*, the compression+substitution+split system; *TSM*, our model. We also provide evaluation measures for the sentences in the evaluation dataset: *CW*: complex sentences from Normal Wikipedia and *SW*: parallel simple sentences from Simple Wikipedia.

6.1 Basic Statistics and Examples

The first three columns in Tab. 9 present the basic statistics for the evaluation sentences and the output of the five systems. $tokenLen$ is the average length of tokens which may roughly reflect the lexical difficulty. TSM achieves an average token length which is the same as the Simple Wikipedia (SW). $senLen$ is the average number of tokens in one sentence, which may roughly reflect the syntactic complexity. Both TSM and CSS produce shorter sentences than SW. Moses is very close to CW. $\#sen$ gives the number of sentences. Moses, C and CS cannot split sentences and thus produce about the same number of sentences as available in CW.

Here are two example results obtained with our TSM system.

Example 1. *CW*: "Genetic engineering has expanded the genes available to breeders to utilize in creating desired germlines for new crops." *SW*: "New plants were created with genetic engineering." *TSM*: "Engineering has expanded the genes available to breeders to use in making germlines for new crops."

Example 2. *CW*: "An umbrella term is a word that provides a superset or grouping of related concepts, also called a hypernym." *SW*: "An umbrella term is a word that provides a superset or grouping of related concepts." *TSM*: "An umbrella term is a word. A word provides a superset of related concepts, called a hypernym."

In the first example, both substitution and dropping happen. TSM replaces "utilize" and "creating" with "use" and "making". "Genetic" is dropped. In the second example, the complex sentence is split and "also" is dropped.

6.2 Translation Assessment

In this part of the evaluation, we use traditional measures used for evaluating MT systems. Tab. 9 shows the BLEU and NIST scores. We use "mteval-v11b.pl"[9] as the evaluation tool. CW and SW are used respectively as source and reference sentences. TSM obtains a very high BLEU score (0.38) but not as high as Moses (0.55). However, the original complex sentences (CW) from Normal Wikipedia get a rather high BLEU (0.50), when compared to the simple sentences. We also find that most of the sentences generated by Moses are exactly the same as those in CW: this shows that Moses only performs few modifications to the original complex sentences. This is confirmed by MT evaluation measures: if we set CW as both source and reference, the BLEU score obtained by Moses is 0.78. TSM gets 0.55 in the same setting which is significantly smaller than Moses and demonstrates that TSM is able to generate simplifications with a greater amount of variation from the original sentence. As shown in the "#Same" column of Tab. 9, 25 sentences generated by Moses are exactly identical to the complex sentences, while the number for TSM is 2 which is closer to SW. It is however not clear how well BLEU and NIST discriminate simplification systems. As discussed in Jurafsky and Martin (2008), "BLEU does poorly at comparing systems with radically different architectures and is most appropriate when evaluating incremental changes with similar architectures." In our case, TSM and CSS can be considered as having similar architectures as both of them can do splitting, dropping

[8] `http://212.126.215.106/compression/`

[9] `http://www.statmt.org/moses/`

	TokLen	SenLen	#Sen	BLEU	NIST	#Same	Flesch	Lix(Grade)	OOV%	PPL
CW	4.95	27.81	100	0.50	6.89	100	49.1	53.0 (10)	52.9	384
SW	4.76	17.86	131	1.00	10.98	3	60.4 (PE)	44.1 (8)	50.7	179
Moses	4.81	26.08	100	**0.55**	**7.47**	25	54.8	48.1 (9)	52.0	363
C	4.98	18.02	103	0.28	5.37	1	56.2	45.9 (8)	51.7	481
CS	4.90	18.11	103	0.19	4.51	0	59.1	45.1 (8)	**49.5**	616
CSS	4.98	**10.20**	**182**	0.18	4.42	0	65.5 (PE)	38.3 (6)	53.4	581
TSM	**4.76**	13.57	180	0.38	6.21	**2**	**67.4 (PE)**	**36.7 (5)**	50.8	**353**

Table 9: Evaluation

and substitution. But Moses mostly cannot split and drop. We may conclude that TSM and Moses have different architectures and BLEU or NIST is not suitable for comparing them. Here is an example to illustrate this: *(CW):* "Almost as soon as he leaves, Annius and the guard Publius arrive to **escort** Vitellia to Titus, who has now chosen her as his empress." *(SW):* "Almost as soon as he leaves, Annius and the guard Publius arrive to **take** Vitellia to Titus, who has now chosen her as his empress." *(Moses):* The same as (SW). *(TSM):* "Annius and the guard Publius arrive to **take** Vitellia to Titus. Titus has now chosen her as his empress." In this example, *Moses* generates an exactly identical sentence to *SW*, thus the BLUE and NIST scores of *Moses* is the highest. *TSM* simplifies the complex sentence by dropping, splitting and substitution, which results in two sentences that are quite different from the *SW* sentence and thus gets lower BLUE and NIST scores. Nevertheless, the sentences generated by *TSM* seem better than *Moses* in terms of simplification.

6.3 Readability Assessment

Intuitively, readability scores should be suitable metrics for simplification systems. We use the Linux "style" command to calculate the Flesch and Lix readability scores. The results are presented in Tab. 9. "PE" in the Flesch column stands for "Plain English" and the "Grade" in Lix represents the school year. TSM achieves significantly better scores than Moses which has the best BLEU score. This implies that good monolingual translation is not necessarily good simplification. OOV is the percentage of words that are not in the Basic English BE850 list.[10] TSM is ranked as the second best system for this criterion.

The perplexity (PPL) is a score of text probability measured by a language model and normalized by the number of words in the text (Equ. 6). PPL can be used to measure how tight the language model fits the text. Language models constitute an important feature for assessing readability (Schwarm and Ostendorf, 2005). We train a trigram LM using the simple sentences in PWKP and calculate the PPL with SRILM. TSM gets the best PPL score. From this table, we can conclude that TSM achieves better overall readability than the baseline systems.

$$PPL(text) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}} \tag{6}$$

There are still some important issues to be considered in future. Based on our observations, the current model performs well for word substitution and segmentation. But the completion of the new sentences is still problematic. For example, we copy the dependent NP to the new sentences. This may break the coherence between sentences. A better solution would be to use a pronoun to replace the NP. Sometimes, excessive droppings occur, e.g., "older" and "twin" are dropped in "She has an older brother and a twin brother...". This results in a problematic sentence: "She has an brother and a brother...". There are also some errors which stem from the dependency parser. In Example 2, "An umbrella term" should be a dependency of "called". But the parser returns "superset" as the dependency. In the future, we will investigate more sophisticated features and rules to enhance TSM.

7 Conclusions

In this paper, we presented a novel large-scale parallel dataset PWKP for sentence simplification. We proposed TSM, a tree-based translation model for sentence simplification which covers splitting, dropping, reordering and word/phrase substitution integrally for the first time. We also described an efficient training method with speeding up techniques for TSM. The evaluation shows that TSM can achieve better overall readability scores than a set of baseline systems.

[10] `http://simple.wikipedia.org/wiki/Wikipedia:Basic_English_alphabetical_wordlist`

References

Barzilay, Regina and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

Carroll, John, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL'99)*, pages 269–270.

Chandrasekar, R., Christine Doran, and B. Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING'96)*, pages 1041–1044.

Filippova, Katja and Michael Strube. 2008a. Dependency tree based sentence compression. In *International Natural Language Generation Conference (INLG'08)*, pages 25–32.

Filippova, Katja and Michael Strube. 2008b. Sentence fusion via dependency graph compression. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 177–185.

Graehl, Jonathan, Kevin Knight, and Jonathan May. 2008. Training tree transducers. In *Computational Linguistics*, volume 34, pages 391–427. MIT Press.

Heilman, M. and N. A. Smith. 2009. Question generation via overgenerating transformations and ranking. Technical Report CMU-LTI-09-013, Language Technologies Institute, Carnegie Mellon University.

Inui, Kentaro, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. 2003. Text simplification for reading assistance: A project note. In *Proceedings of the 2nd International Workshop on Paraphrasing: Paraphrase Acquisition and Applications (IWP)*, pages 9–16.

Jonnalagadda, Siddhartha and Graciela Gonzalez. 2009. Sentence simplification aids protein-protein interaction extraction. In *Proceedings of the 3rd International Symposium on Languages in Biology and Medicine*.

Jurafsky, Daniel and James H. Martin. 2008. *Speech and Language Processing (2nd Edition)*. Prentice Hall, 2 edition.

Klein, Dan and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NISP'02)*, pages 3–10.

Knight, Kevin and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *AAAI*, pages 703–710.

Levenshtein. 1966. Binary code capable of correcting deletions, insertions and reversals. In *Soviet Physics*, pages 707–710.

Nelken, Rani and Stuart M. Shieber. 2006. Towards robust context-sensitive sentence alignment for monolingual corpora. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 161–168.

Petersen, Sarah E. and Mari Ostendorf. 2007. Text simplification for language learners: a corpus analysis. In *Proc. of Workshop on Speech and Language Technology for Education*, pages 69–72.

Schmid, Helmut. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, pages 44–49.

Schwarm, Sarah E. and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *ACL'05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 523–530.

Siddharthan, Advaith. 2002. An architecture for a text simplification system. In *Proceedings of the Language Engineering Conference (LEC'02)*, pages 64–71.

Siddharthan, Advaith. 2006. Syntactic simplification and text cohesion. In *Research on Language & Computation*, volume 4, pages 77–109. Springer Netherlands, June.

Stolcke, Andreas. 2002. SRILM - An Extensible Language Modeling Toolkit. pages 901–904.

Vickrey, David and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*, pages 344–352, June.

Watanabe, Willian Massami, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. 2009. Facilita: reading assistance for low-literacy readers. In *SIGDOC '09: Proceedings of the 27th ACM international conference on Design of communication*, pages 29–36. ACM.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *ACL'01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530.

Yamada, Kenji and Kevin Knight. 2002. A decoder for syntax-based statistical mt. In *ACL'02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 303–310.

Zesch, Torsten, Christof Müller, and Iryna Gurevych. 2008. Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, pages 1646–1652.

Zhao, Shiqi, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of ACL-IJCNLP*, pages 834–842, Suntec, Singapore, August.

A Minimum Error Weighting Combination Strategy for Chinese Semantic Role Labeling

Tao Zhuang and **Chengqing Zong**
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
{tzhuang, cqzong}@nlpr.ia.ac.cn

Abstract

Many Semantic Role Labeling (SRL) combination strategies have been proposed and tested on English SRL task. But little is known about how much Chinese SRL can benefit from system combination. And existing combination strategies trust each individual system's output with the same confidence when merging them into a pool of candidates. In our approach, we assign different weights to different system outputs, and add a weighted merging stage to the conventional SRL combination architecture. We also propose a method to obtain an appropriate weight for each system's output by minimizing some error function on the development set. We have evaluated our strategy on Chinese Proposition Bank data set. With our minimum error weighting strategy, the F_1 score of the combined result achieves 80.45%, which is 1.12% higher than baseline combination method's result, and 4.90% higher than the best individual system's result.

1 Introduction

In recent years, Chinese Semantic Role Labeling has received much research effort (Sun and Jurafsky, 2004; Xue, 2008; Che et al., 2008; Ding and Chang, 2008; Sun et al., 2009; Li et al., 2009). And Chinese SRL is also included in CoNLL-2009 shared task (Hajič et al., 2009). On the data set used in (Xue, 2008), the F_1 score of the SRL results using automatic syntactic analysis is still in low 70s (Xue, 2008; Che et al., 2008; Sun et al., 2009). As pointed out by Xue (Xue, 2008), the SRL errors are mainly caused by the errors in automatic syntactic analysis. In fact, Chinese SRL suffers from parsing errors even more than English SRL, because the state-of-the-art parser for Chinese is still not as good as that for English. And previous research on English SRL shows that combination is a robust and effective method to alleviate SRL's dependency on parsing results (Màrquez et al., 2005; Koomen et al., 2005; Pradhan et al., 2005; Surdeanu et al., 2007; Toutanova et al., 2008). However, the effect of combination for Chinese SRL task is still unknown. This raises two questions at least: (1) How much can Chinese SRL benefit from combination? (2) Can existing combination strategies be improved? All existing combination strategies trust each individual system's output with the same confidence when putting them into a pool of candidates. But according to our intuition, different systems have different performance. And the system that have better performance should be trusted with more confidence. We can use our prior knowledge about the combined systems to do a better combination.

The observations above motivated the work in this paper. Instead of directly merging outputs with equal weights, different outputs are assigned different weights in our approach. An output's weight stands for the confidence we have in that output. We acquire these weights by minimizing an error function on the development set. And we use these weights to merge the outputs. In this paper, outputs are generated by a full parsing based Chinese SRL system and a shallow parsing based SRL system. The full parsing based system

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1362–1370,
Beijing, August 2010

use multiple parse trees to generate multiple SRL outputs. Whereas the shallow parsing based system only produce one SRL output. After merging all SRL outputs, we use greedy and integer linear programming combination methods to combine the merged outputs.

We have evaluated our combination strategy on Chinese Propbank data set used in (Xue, 2008) and get encouraging results. With our minimum error weighting (MEW) strategy, the F_1 score of the combined result achieves 80.45%. This is a significant improvement over the best reported SRL performance on this data set, which is 74.12% in the literature (Sun et al., 2009).

2 Related work

A lot of research has been done on SRL combination. Most of them focused on English SRL task. But the combination methods are general. And they are closely related to the work in this paper.

Punyakanok et al. (2004) formulated an Integer Linear Programming (ILP) model for SRL. Based on that work, Koomen et al. (2005) combined several SRL outputs using ILP method. Màrquez et al. (2005) proposed a combination strategy that does not require the individual system to give a score for each argument. They used a binary classifier to filter different systems' outputs. Then they used a greedy method to combine the candidates that pass the filtering process. Pradhan et al. (2005) combined systems that are based on phrase-structure parsing, dependency parsing, and shallow parsing. They also used greedy method when combining different outputs. Surdeanu et al. (2007) did a complete research on a variety of combination strategies. All these research shows that combination can improve English SRL performance by 2$\sim$5 points on F_1 score. However, little is known about how much Chinese SRL can benefit from combination. And, as we will show, existing combination strategies can still be improved.

3 Individual SRL Systems

3.1 Full Parsing Based System

The full parsing based system utilize full syntactic analysis to perform semantic role labeling. We implemented a Chinese semantic role labeling system similar to the one described in (Xue, 2008). Our system consists of an argument identification stage and an argument classification stage. In the argument identification stage, a number of argument locations are identified in a sentence. In the argument classification stage, each location identified in the first stage is assigned a semantic role label. The features used in this paper are the same with those used in (Xue, 2008).

Maximum entropy classifier is employed for both the argument identification and classification tasks. And Zhang Le's MaxEnt toolkit[1] is used for implementation.

3.2 Shallow Parsing Based System

The shallow parsing based system utilize shallow syntactic information at the level of phrase chunks to perform semantic role labeling. Sun et al. (2009) proposed such a system on Chinese SRL and reported encouraging results. The system used in this paper is based on their approach. For Chinese chunking, we adopted the method used in (Chen et al., 2006), in which chunking is regarded as a sequence labeling task with IBO2 representation. The features used for chunking are the uni-gram and bi-gram word/POS tags with a window of size 2. The SRL task is also regarded as a sequence labeling problem. For an argument with label ARG*, we assign the label B-ARG* to its first chunk, and the label I-ARG* to its rest chunks. The chunks outside of any argument are assigned the label O. The features used for SRL are the same with those used in the one-stage method in (Sun et al., 2009).

In this paper, we employ Tiny SVM along with Yamcha (Kudo and Matsumoto, 2001) for Chinese chunking, and CRF++[2] for SRL.

3.3 Individual systems' outputs

The maximum entropy classifier used in full parsing based system and the CRF model used in shallow paring based system can both output classification probabilities. For the full parsing based system, the classification probability of the ar-

[1] http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit .html

[2] http://crfpp.sourceforge.net/

gument classification stage is used as the argument's probability. Whereas for the shallow parsing based system, an argument is usually comprised of multiple chunks. For example, an argument with label ARG0 may contain three chunks labeled as: B-ARG0, I-ARG0, I-ARG0. And each chunk has a label probability. Thus we have three probabilities p_1, p_2, p_3 for one argument. In this case, we use the geometric mean of individual chunks' probabilities $(p_1 \cdot p_2 \cdot p_3)^{1/3}$ as the argument's probability.

As illustrated in Figure 1, in an individual system's output, each argument has three attributes: its location in sentence loc, represented by the number of its first word and last word; its semantic role label l; and its probability p.

Sent:	外商 投资 企业	成为	中国 外贸 重要 增长点
Args:	[ARG0]	[pred]	[ARG1]
loc:	(0, 2)		(4, 7)
l:	ARG0		ARG1
p:	0.94		0.92

Figure 1: Three attributes of an output argument: location loc, label l, and probability p.

So each argument outputted by a system is a triple (loc, l, p). For example, the ARG0 in Figure 1 is $((0, 2), \text{ARG0}, 0.94)$. Because the outputs of baseline systems are to be combined, we call such triple a **candidate** for combination.

4 Approach Overview

As illustrated in Figure 2, the architecture of our system consists of a candidates generation stage, a weighted merging stage, and a combination stage. In the candidates generation stage, the baseline systems are run individually and their outputs are collected. We use 2-best parse trees of Berkeley parser (Petrov and Klein, 2007) and 1-best parse tree of Bikel parser (Bikel, 2004) and Stanford parser (Klein and Manning, 2003) as inputs to the full parsing based system. The second best parse tree of Berkeley parser is used here for its good quality. So together we have four different outputs from the full parsing based system. From the shallow parsing based system, we have only one output.

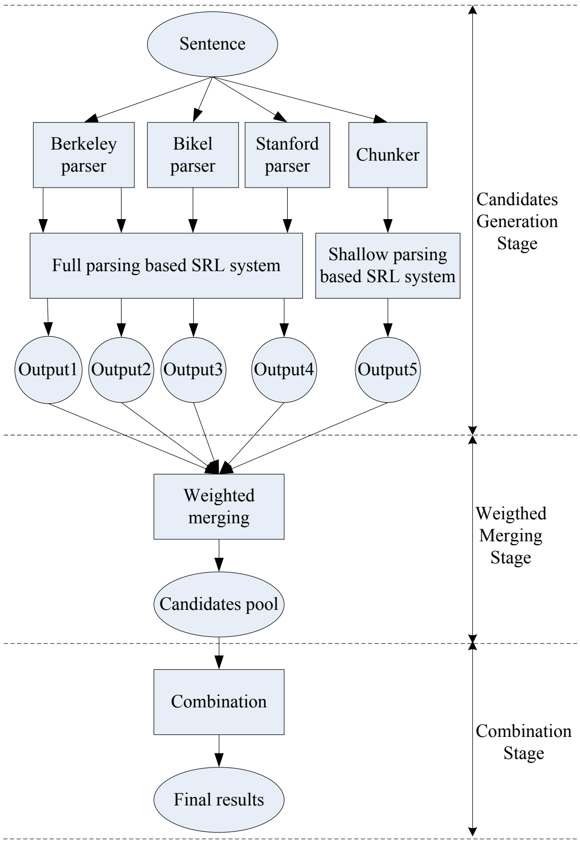


Figure 2: The overall architecture of our system.

In the weighted merging stage, each system output is assigned a weight according to our prior knowledge obtained on the development set. Details about how to obtain appropriate weights will be explained in Section 6. Then all candidates with the same loc and l are merged to one by weighted summing their probabilities. Specifically, suppose that there are n system outputs to be combined, with the i-th output's weight to be w_i. And the candidate in the i-th output with loc and l is (loc, l, p_i) (If there is no candidate with loc and l in the i-th output, p_i is 0.). Then the merged candidate is (loc, l, p), where $p = \sum_{i=1}^{n} w_i p_i$.

After the merging stage, a pool of merged candidates is obtained. In the combination stage, candidates in the pool are combined to form a consistent SRL result. Greedy and integer linear programming combination methods are experimented in this paper.

5 Combination Methods

5.1 Global constraints

When combining the outputs, two global constraints are enforced to resolve the conflict between outputs. These two constraints are:

1. *No duplication*: There is no duplication for key arguments: ARG0 $\sim$ ARG5.

2. *No overlapping*: Arguments cannot overlap with each other.

We say two argument candidates **conflict** with each other if they do not satisfy the two constraints above.

5.2 Two combination methods

Under these constraints, two methods are explored to combine the outputs. The first one is a greedy method. In this method, candidates with probability below a threshold are deleted at first. Then the remaining candidates are inspected in descending order according to their probabilities. And each candidate will be put into a solution set if it does not conflict with candidates already in the set. This greedy combination method is very simple and has been adopted in previous research (Pradhan et al., 2005; Màrquez et al., 2005).

The second combination method is integer linear programming (ILP) method. ILP method was first applied to SRL in (Punyakanok et al., 2004). Here we formulate an ILP model whose form is different from the model in (Punyakanok et al., 2004; Koomen et al., 2005). For convenience, we denote the whole label set as $\{l_1, l_2, \dots, l_n\}$. And let $l_1 \sim l_6$ stand for the key argument labels ARG0 $\sim$ ARG5 respectively. Suppose there are m different locations, denoted as $loc_1, \dots, loc_m$, among all candidates in the pool. And the probability of assigning l_j to loc_i is p_{ij}. A binary variable x_{ij} is defined as:

$$x_{ij} = \begin{cases} 1 & \text{if } loc_i \text{ is assigned label } l_j, \\ 0 & \text{otherwise.} \end{cases}$$

The objective of the ILP model is to maximize the sum of arguments' probabilities:

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} (p_{ij} - T) x_{ij} \qquad (1)$$

where T is a threshold to prevent including too many candidates in solution. T is similar to the threshold in greedy combination method. In this paper, both thresholds are empirically tuned on development data, and both are set to be 0.2.

The inequalities in equation (2) make sure that each loc is assigned at most one label.

$$\forall 1 \le i \le m : \sum_{j=1}^{n} x_{ij} \le 1 \qquad (2)$$

The inequalities in equation (3) satisfy the *No duplication* constraint.

$$\forall 1 \le j \le 6 : \sum_{i=1}^{m} x_{ij} \le 1 \qquad (3)$$

For any location loc_i, let C_i denote the index set of the locations that overlap with it. Then the *No overlapping* constraint means that if loc_i is assigned a label, i.e., $\sum_{j=1}^{n} x_{ij} = 1$, then for any $k \in C_i$, loc_k cannot be assigned any label, i.e., $\sum_{j=1}^{n} x_{kj} = 0$. A common technique in ILP modeling to form such a constraint is to use a sufficiently large auxiliary constant M. And the constraint is formulated as:

$$\forall 1 \le i \le m : \sum_{k \in C_i} \sum_{j=1}^{n} x_{kj} \le (1 - \sum_{j=1}^{n} x_{ij}) M \qquad (4)$$

In this case, M only needs to be larger than the number of candidates to be combined. In this paper, $M = 500$ is large enough. And we employ lpsolve[3] to solve the ILP model.

Note that the form of the ILP model in this paper is different from that in (Punyakanok et al., 2004; Koomen et al., 2005) in three aspects: (1) A special label class *null*, which means no label is assigned, was added to the label set in (Punyakanok et al., 2004; Koomen et al., 2005). Whereas no such special class is needed in our model, because if no label is assigned to loc_i, $\sum_{j=1}^{n} x_{ij} = 0$ would simply indicate this case. This makes our model contain fewer variables. (2) Without *null* class in our model, we need to use a different technique to formulate the *No-overlapping* constraint. (3) In order to compare

[3] http://lpsolve.sourceforge.net/

with the greedy combination method, the ILP model in this paper conforms to exactly the same constraints as the greedy method. Whereas many more global constraints were taken into account in (Punyakanok et al., 2004; Koomen et al., 2005).

6 Train Minimum Error Weights

The idea of minimum error weighting is straightforward. Individual outputs $O_1, O_2, \ldots, O_n$ are assigned weights $w_1, w_2, \ldots, w_n$ respectively. These weights are normalized, i.e., $\sum_{i=1}^{n} w_i = 1$. An output's weight can be seen as the confidence we have in that output. It is a kind of prior knowledge we have about that output. We can gain this prior knowledge on the development set. As long as the data of the development set and the test set are similar, this prior knowledge should be able to help to guide SRL combination on test set. In this section, we discuss how to obtain appropriate weights.

6.1 Training model

Suppose the golden answer and SRL result on development set are d and r respectively. An **error function** $Er(r, d)$ is a function that measures the error contained in r in reference to d. An error function can be defined as the number of wrong arguments in r. It can also be defined using precision, recall, or F_1 score. For example, $Er(r, d) = 1 - Precision(r, d)$, or $Er(r, d) = 1 - F_1(r, d)$. Smaller value of error function means less error in r.

The combination process can also be seen as a function, which maps the outputs and weights to the combined result r: $r = Comb(O_1^n, w_1^n)$. Therefore, the error function of our system on development set is:

$$Er(r, d) = Er(Comb(O_1^n, w_1^n), d) \qquad (5)$$

From equation (5), it can be seen that: Given development set d, if the outputs to be combined O_1^n and the combination method $Comb$ are fixed, the error function is just a function of the weights. So we can obtain appropriate weights by minimizing the error function:

$$\hat{w_1^n} = \arg\min_{w_1^n} Er(Comb(O_1^n, w_1^n), d) \qquad (6)$$

6.2 Training algorithm

Algorithm 1 Powell Training Algorithm.

```
 1: Input : Error function $Er(\boldsymbol{w})$.
 2: Initialize $n$ directions $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_n$, and
    a start point $\boldsymbol{w}$ in $R^n$.
 3: Set termination threshold $\delta$.
 4: do:
 5:     $\boldsymbol{w}_1 \leftarrow \boldsymbol{w}$
 6:     for $i \leftarrow 1, \ldots, n$:
 7:         $\alpha_i \leftarrow \arg\min_{\alpha} f(\boldsymbol{w}_i + \alpha \boldsymbol{d}_i)$
 8:         $\boldsymbol{w}_{i+1} \leftarrow \boldsymbol{w}_i + \alpha_i \boldsymbol{d}_i$
 9:     $\boldsymbol{d}_{n+1} \leftarrow \boldsymbol{w}_{n+1} - \boldsymbol{w}$
10:     $\alpha^* \leftarrow \arg\min_{\alpha} f(\boldsymbol{w} + \alpha \boldsymbol{d}_{n+1})$
11:     $\boldsymbol{w}' \leftarrow \boldsymbol{w} + \alpha^* \boldsymbol{d}_{n+1}$
12:     $\Delta Er \leftarrow Er(\boldsymbol{w}) - Er(\boldsymbol{w}')$
13:     $i \leftarrow \arg\max_{1 \leq j \leq n} Er(\boldsymbol{w}_j) - Er(\boldsymbol{w}_{j+1})$
14:     if $(\alpha^*)^2 \geq \frac{\Delta Er}{Er(\boldsymbol{w}_i) - Er(\boldsymbol{w}_{i+1})}$:
15:         for $j \leftarrow i, \ldots, n$:
16:             $\boldsymbol{d}_j \leftarrow \boldsymbol{d}_{j+1}$
17:     $\boldsymbol{w} \leftarrow \boldsymbol{w}'$
18: while $\Delta Er > \delta$
19: Output: The minimum error weights $\boldsymbol{w}$.
```

There are two difficulties to solve the optimization problem in equation 6. The first one is that the error function cannot be written to an analytical form. This is because the $Comb$ function, which stands for the combination process, cannot be written as an analytical formula. So the problem cannot be solved using canonical gradient-based optimization algorithms, because the gradient function cannot be derived. The second difficulty is that, according to our experience, the error function has many local optima, which makes it difficult to find a global optima.

To resolve the first difficulty, Modified Powell's method (Yuan, 1993) is employed to solve the optimization problem. Powell's method is a heuristic search method that does not require the objective function to have an explicit analytical form. The training algorithm is presented in Algorithm 1. In Algorithm 1, the line search problem in steps 7 and 10 is solved using Brent's method (Yuan, 1993). And the temination threshold δ is empirically set to be 0.001 in this paper.

To resolve the second difficulty, we perform multiple searches using different start points, and then choose the best solution found.

7 Experiments

7.1 Experimental setup

We use Chinese Proposition Bank (CPB) 1.0 and Chinese Tree Bank (CTB) 5.0 of Linguistic Data Consortium corpus in our experiments. The training set is comprised of 648 files(chtb_081.fid to chtb_885.fid). The development set is comprised of 40 files(chtb_041.fid to chtb_080.fid). The test set is comprised of 72 files(chtb_001.fid to chtb_040.fid and chtb_900.fid to chtb_931.fid).

The same data setting has been used in (Xue, 2008; Ding and Chang, 2008; Sun et al., 2009). Sun et al. (2009) used sentences with golden segmentation and POS tags as input to their SRL system. However, we use sentences with only golden segmentation as input. Then we perform automatic POS tagging using Stanford POS tagger (Toutanova et al., 2003). In (Xue, 2008), the parser used by the SRL system is trained on the training and development set plus 275K words of broadcast news. In this paper, all parsers used by the full parsing based system are trained on the training set plus the broadcast news portion of CTB6.0. And the chunker used in the shallow parsing based system is trained just on the training set.

7.2 Individual outputs' performance

In this paper the four outputs of the full parsing based system are represented by FO1 $\sim$ FO4 respectively. Among them, FO1 and FO2 are the outputs using the first and second best parse trees of Berkeley parser, FO3 and FO4 are the outputs using the best parse trees of Stanford parser and Bikel parser respectively. The output of the shallow parsing based system is represented by SO. The individual outputs' performance on development and test set are listed in Table 1.

From Table 1 we can see that the performance of individual outputs are similar on development set and test set. On both sets, the F_1 scores of individual outputs are in the same order: FO1 $>$ FO2 $>$ SO $>$ FO3 $>$ FO4.

Data set	Outputs	$P(\%)$	$R(\%)$	F_1
development	FO1	**79.17**	**72.09**	**75.47**
	FO2	77.89	70.56	74.04
	FO3	72.57	67.02	69.68
	FO4	75.60	63.45	69.00
	SO	73.72	67.35	70.39
test	FO1	**80.75**	**70.98**	**75.55**
	FO2	79.44	69.37	74.06
	FO3	73.95	66.37	70.00
	FO4	75.89	63.26	69.00
	SO	75.69	67.90	71.59

Table 1: The results of individual systems on development and test set.

7.3 Combining outputs of full parsing based system

In order to investigate the benefit that the full parsing based system can get from using multiple parsers, we combine the four outputs FO1 $\sim$ FO4. The combination results are listed in Table 2. In tables of this paper, "Grd" and "ILP" stand for greedy and ILP combination methods respectively, and "+MEW" means the combination is performed with MEW strategy.

	$P(\%)$	$R(\%)$	F_1
Grd	**82.68**	73.36	77.74
ILP	82.21	73.93	77.85
Grd+MEW	81.30	75.38	78.23
ILP+MEW	81.27	**75.74**	**78.41**

Table 2: The results of combining outputs of full parsing based system on test set.

	Er	FO1	FO2	FO3	FO4
Grd	$1 - F_1$	0.31	0.16	0.30	0.23
ILP	$1 - F_1$	0.33	0.10	0.27	0.30

Table 3: The minimum error weights for the results in Table 2.

From Table 2 and Table 1, we can see that, without MEW strategy, the F_1 score of combination result is about 2.3% higher than the best individual output. With MEW strategy, the F_1 score is improved about 0.5% further. That is to say, with MEW strategy, the benefit of combination is improved by about 20%. Therefore, the effect of MEW is very encouraging.

Here the error function for MEW training is chosen to be $1 - F_1$. And the trained weights for greedy and ILP methods are listed in Table 3 separately. In tables of this paper, the column Er corresponds to the error function used for MEW strategy.

7.4 Combining all outputs

We have also combined all five outputs. The results are listed in Table 4. Compared with the results in Table 2, we can see that the combination results is largely improved, especially the recall.

	P(%)	R(%)	F_1
Grd	**83.64**	75.32	79.26
ILP	83.31	75.71	79.33
Grd+MEW	83.34	77.47	80.30
ILP+MEW	83.02	**78.03**	**80.45**

Table 4: The results of combining all outputs on test set.

From Table 4 and Table 1 we can see that without MEW strategy, the F_1 score of combination result is about 3.8% higher than the best individual output. With MEW, the F_1 score is improved further by more than 1%. That means the benefit of combination is improved by over 25% with MEW strategy.

Here the error function for MEW training is still $1 - F_1$, and the trained weights are listed in Table 5.

	Er	FO1	FO2	FO3	FO4	SO
Grd	$1 - F_1$	0.23	0.12	0.23	0.20	0.22
ILP	$1 - F_1$	0.24	0.08	0.22	0.21	0.25

Table 5: The minimum error weights for the results in Table 4.

7.5 Using alternative error functions for minimum error weights training

In previous experiments, we use $1 - F_1$ as error function. As pointed out in Section 6, the definition of error function is very general. So we have experimented with two other error functions, which are $1 - Precision$, and $1 - Recall$. Obviously, these two error functions favor precision and recall separately. The results of combining all five outputs using these two error functions are listed in Table 6, and the trained weights are listed in Table 7.

	Er	P(%)	R(%)	F_1
Grd+MEW	$1 - P$	85.31	73.42	78.92
ILP+MEW	$1 - P$	**85.62**	72.76	78.67
Grd+MEW	$1 - R$	81.94	77.55	79.68
ILP+MEW	$1 - R$	79.74	**78.34**	79.03

Table 6: The results of combining all outputs with alternative error functions.

	Er	FO1	FO2	FO3	FO4	SO
Grd	$1 - P$	0.25	0.24	0.22	0.22	0.07
ILP	$1 - P$	0.30	0.26	0.20	0.15	0.09
Grd	$1 - R$	0.21	0.10	0.17	0.15	0.37
ILP	$1 - R$	0.24	0.04	0.10	0.22	0.39

Table 7: The minimum error weights for the results in Table 6.

From Table 6 and Table 4, we can see that when $1 - Precison$ is used as error function, the precision of combination result is largely improved. But the recall decreases a lot. Similar effect of the error function $1 - Recall$ is also observed.

The results of this subsection reflect the flexibility of MEW strategy. This flexibility comes from the generality of the definition of error function. The choice of error function gives us some control over the results we want to get. We can define different error functions to favor precision, or recall, or some error counts such as the number of misclassified arguments.

7.6 Discussion

In this paper, the greedy and ILP combination methods conform to the same simple constraints specified in Section 5. From the experiment results, we can see that ILP method generates slightly better results than greedy method.

In Subsection 7.4, we see that combining all outputs using ILP method with MEW strategy yields 4.90% improvement on F_1 score over the best individual output FO1. In order to understand each output's contribution to the improvement over FO1. We compare the differences between outputs.

Let C_O denote the set of correct arguments in an output O. Then we get the following statistics when comparing two outputs A and B: (1) the number of common correct arguments in A and B, i.e., $|C_A \cap C_B|$; (2) the number of correct arguments in A and not in B, i.e., $|C_A \setminus C_B|$; (3) the number of correct arguments in B and not in A, i.e., $|C_B \setminus C_A|$. The comparison results between some outputs on test set are listed in Table 8. In this table, UF stands for the union of the 4 outputs FO1 $\sim$ FO4.

A	B	$\lvert C_A \cap C_B \rvert$	$\lvert C_A \setminus C_B \rvert$	$\lvert C_B \setminus C_A \rvert$
FO1	FO2	5498	508	372
	FO3	5044	962	552
	FO4	4815	1191	512
	SO	4826	1180	920
UF	SO	5311	1550	435

Table 8: Comparison between outputs on test set.

From Table 8 we can see that the output SO has 4826 common correct arguments with FO1, which is relatively small. And, more importantly, SO contains 920 correct arguments not in FO1, which is much more than any other output contains. Therefore, SO is more complementary to FO1 than other outputs. On the contrary, FO2 is least complementary to FO1. Even compared with the union of FO1 $\sim$ FO4, SO still contains 435 correct arguments not in the union. This shows that the output of shallow parsing based system is a good complement to the outputs of full parsing based system. This explains why recall is largely improved when SO is combined in Subsection 7.4. From the analysis above we can also see that the weights in Table 5 are quite reasonable. In Table 5, SO is assigned the largest weight and FO2 is assigned the smallest weight.

In Subsection 7.3, the MEW strategy improves the benefit of combination by about 20%. And in Subsection 7.4, the MEW strategy improves the benefit of combination by over 25%. This shows that the MEW strategy is very effective for Chinese SRL combination.

To our best knowledge, no results on Chinese SRL combination has been reported in the literature. Therefore, to compare with previous results, the top two results of single SRL system in the literature and the result of our combination system on this data set are listed in Table 9. For the results in Table 9, the system of Sun et al. uses sentences with golden POS tags as input. Xue's system and our system both use sentences with automatic POS tags as input. The result of Sun et al. (2009) is the best reported result on this data set in the literature.

	POS	P(%)	R(%)	F_1
(Xue, 2008)	auto	76.8	62.5	68.9
(Sun et al., 2009)	gold	79.25	69.61	74.12
Ours	auto	**83.02**	**78.03**	**80.45**

Table 9: Previous best single system's results and our combination system's result on this data set.

8 Conclusions

In this paper, we propose a minimum error weighting strategy for SRL combination and investigate the benefit that Chinese SRL can get from combination. We assign different weights to different system outputs and add a weighted merging stage to conventional SRL combination system architecture. And we also propose a method to train these weights on development set. We evaluate the MEW strategy on Chinese Propbank data set with greedy and ILP combination methods.

Our experiments have shown that the MEW strategy is very effective for Chinese SRL combination, and the benefit of combination can be improved over 25% with this strategy. And also, the MEW strategy is very flexible. With different definitions of error function, this strategy can favor precision, or recall, or F_1 score. The experiments have also shown that Chinese SRL can benefit a lot from combination, especially when systems based on different syntactic views are combined. The SRL result with the highest F_1 score in this paper is generated by ILP combination together with MEW strategy. In fact, the MEW strategy is easy to incorporate with other combination methods, just like incorporating with the greedy and ILP combination methods in this paper.

Acknowledgment

The research work has been partially funded by the Natural Science Foundation of China under Grant No. 60975053, 90820303 and 60736014, the National Key Technology R&D Program under Grant No. 2006BAH03B02, the Hi-Tech Research and Development Program ("863" Program) of China under Grant No. 2006AA010108-4, and also supported by the China-Singapore Institute of Digital Media (CSIDM) project under grant No. CSIDM-200804.

References

Daniel Bikel. 2004. Intricacies of Collins Parsing Model. *Computational Linguistics*, 30(4):480-511.

Wanxiang Che, Min Zhang, Ai Ti Aw, Chew Lim Tan, Ting Liu, and Sheng Li. 2008. Using a Hybrid Convolution Tree Kernel for Semantic Role Labeling. *ACM Transactions on Asian Language Information Processing*, 2008, 7(4).

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. An empirical study of Chinese chunking. In *Proceedings of COLING/ACL-2006*.

Weiwei Ding and Baobao Chang. 2008. Improving Chinese Semantic Role Classification with Hierarchical Feature Selection Strategy. In *Proceedings of EMNLP-2008*.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue and Yi Zhang. 2009. The CoNLL-2009 Shared Task: Syntactic and Semantic Dependencies in Multiple Languages. In *Proceedings of CoNLL-2009*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of ACL-2003*.

Peter Koomen, Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2005. Generalized Inference with Multiple Semantic Role Labeling Systems. In *Proceedings of CoNLL-2005 shared task*.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with Support Vector Machines. In *Proceedings of NAACL-2001*.

Junhui Li, Guodong Zhou, Hai Zhao, Qiaoming Zhu, and Peide Qian. 2009. Improving Nominal SRL in Chinese Language with Verbal SRL Information and Automatic Predicate Recognition. In *Proceedings of EMNLP-2009*.

Lluís Màrquez, Mihai Surdeanu, Pere Comas, and Jordi Turmo. 2005. A Robust Combination Strategy for Semantic Role Labeling. In *Proceedings of EMNLP-2005*.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized parsing. In *Proceedings of ACL-2007*.

Sameer S. Pradhan, Wayne Ward, Kadri Hacioglu, James H. Martin, and Daniel Jurafsky. 2005. Semantic Role Labeling Using Different Syntactic Views. In *Proceedings of ACL-2005*.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. 2004. Semantic Role Labeling via Integer Linear Programming Inference. In *Proceedings of COLING-2004*.

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantic parsing of Chinese. In *Proceedings of NAACL-2004*.

Weiwei Sun, Zhifang Sui, Meng Wang, and Xin Wang. 2009. Chinese Semantic Role Labeling with Shallow Parsing. In *Proceedings of EMNLP-2009*.

Mihai Surdeanu, Lluís Màrquez, Xavier Carreras, and Pere R. Comas. 2007. Combination Strategies for Semantic Role Labeling. *Journal of Artificial Intelligence Research (JAIR)*, 29:105-151.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A Global Joint Model for Semantic Role Labeling. *Computational Linguistics*, 34(2): 145-159.

Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of HLT-NAACL-2003*.

Nianwen Xue. 2008. Labeling Chinese Predicates with Semantic Roles. *Computational Linguistics*, 34(2): 225-255.

Yaxiang Yuan. 1993. *Numerical Methods for Nonlinear Programming*. Shanghai Scientific and Technical Pulishers, Shanghai.

Detecting Speech Repairs Incrementally Using a Noisy Channel Approach

Simon Zwarts, Mark Johnson and Robert Dale
Centre for Language Technology
Macquarie University
{simon.zwarts|mark.johnson|robert.dale}@mq.edu.au

Abstract

Unrehearsed spoken language often contains disfluencies. In order to correctly interpret a spoken utterance, any such disfluencies must be identified and removed or otherwise dealt with. Operating on transcripts of speech which contain disfluencies, our particular focus here is the identification and correction of speech repairs using a noisy channel model. Our aim is to develop a high-accuracy mechanism that can identify speech repairs in an incremental fashion, as the utterance is processed word-by-word.

We also address the issue of the evaluation of such incremental systems. We propose a novel approach to evaluation, which evaluates performance in detecting and correcting disfluencies incrementally, rather than only assessing performance once the processing of an utterance is complete. This demonstrates some shortcomings in our basic incremental model, and so we then demonstrate a technique that improves performance on the detection of disfluencies as they happen.

1 Introduction

One of the most obvious differences between written language and spoken language is the fact that the latter presents itself incrementally over some time period. Most natural language processing applications operate on complete sentences; but for real time spontaneous speech, there are potential benefits to incrementally processing the input so that a system can stay responsive and interact directly before a speaker's utterance is complete. Work in psycholinguistics supports the view that the human parsing mechanism works incrementally, with partial semantic interpretations being produced before the complete utterance has been heard (Marslen-Wilson, 1973). Our interest is in developing similarly incremental processing techniques for natural language interpretation, so that, for example, a speech recognizer might be able to interject during a long utterance to object, cut the speaker short, or correct a mistaken assumption; such a mechanism is even required for the appropriate timing of backchannel signals. Additionally the incremental nature of the model allows potential application of this model in speech recognition models.

Another feature of unrehearsed spoken language that has no obvious correlate in written language is the presence of disfluencies.[1] Disfluencies are of different types, ranging from simple filled pauses (such as *um* and *uh*) to more complicated structures where the sequence of words that make up the utterance is 'repaired' while it is being produced. Whereas simpler disfluencies may be handled by simply deleting them from the sequence of words under consideration, the editing terms in a speech repair are part of the utterance, and therefore require more sophisticated processing.

There are three innovations in the present paper. First, we demonstrate that a noisy channel model of speech repairs can work accurately in an incremental fashion. Second, we provide an approach to the evaluation of

[1] Although some disfluencies can be considered grammatical errors, they are generally quite distinct in both cause and nature from the kinds of grammatical errors found in written text.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1371–1378,
Beijing, August 2010

such an incremental model. Third, we tackle the problem of the early detection of speech repairs, and demonstrate a technique that decreases the latency (as measured in tokens) involved in spotting that a disfluency has occurred.

The rest of the paper is structured as follows. Section 2 provides some background on speech repairs and existing approaches to handling them, including Johnson and Charniak's (2004) model, which we use as a starting point for our incremental model. Section 3 describes our model in detail, focusing on the noisy channel model and the incremental component of this model. Section 4 introduces some considerations that arise in the development of techniques for the evaluation of incremental disfluency detection; we then provide a quantitative assessment of our performance using these techniques. Our evaluation reveals that our basic incremental model does not perform very well at detecting disfluencies close to where they happen, so in Section 5 we present a novel approach to optimise detection of these disfluencies as early as possible. Finally Section 6 concludes and discusses future work.

2 Speech Repairs

We adopt the terminology and definitions introduced by Shriberg (1994) to discuss disfluency. We are particularly interested in what are called **repairs**. These are the hardest types of disfluency to identify since they are not marked by a characteristic vocabulary. Shriberg (1994) identifies and defines three distinct parts of a repair, referred to as the **reparandum**, the **interregnum** and the **repair**. Consider the following utterance:

$$\begin{array}{c} \textit{I want a flight } \overbrace{\textit{to Boston,}}^{\text{reparandum}} \\ \underbrace{\textit{uh, I mean}}_{\text{interregnum}}\underbrace{\textit{to Denver}}_{\text{repair}} \textit{ on Friday} \end{array} \qquad (1)$$

The reparandum *to Boston* is the part of the utterance that is being edited out; the interregnum *uh* is a filler, which may not always be present; and the repair *to Denver* replaces the reparandum.

Given an utterance that contains such a repair, we want to be able to correctly detect the start and end positions of each of these three components. We can think of each word in an utterance as belonging to one of four categories: fluent material, reparandum, interregnum, or repair. We can then assess the accuracy of techniques that attempt to detect disfluencies by computing precision and recall values for the assignment of the correct categories to each of the words in the utterance, as compared to the gold standard as indicated by annotations in the corpus.

An alternative means of evaluation would be to simply generate a new signal with the reparandum and filler removed, and compare this against a 'cleaned-up' version of the utterance; however, Core and Schubert (1999) argue that, especially in the case of speech repairs, it is important not to simply throw away the disfluent elements of an utterance, since they can carry meaning that needs to be recovered for proper interpretation of the utterance. We are therefore interested in the first instance in a model of speech error *detection*, rather than a model of *correction*.

Johnson and Charniak (2004) describe such a model, using a noisy-channel based approach to the detection of the start and end points of reparanda, interregna and repairs. Since we use this model as our starting point, we provide a more detailed explanation in Section 3.

The idea of using a noisy channel model to identify speech repairs has been explored for languages other than English. Honal and Schultz (2003) use such a model, comparing speech disfluency detection in spontaneous spoken Mandarin against that in English. The approach performs well in Mandarin, although better still in English.

Both the models just described operate on transcripts of completed utterances. Ideally, however, when we deal with speech we would like to process the input word by word as it is received. Being able to do this would enable tighter integration in both speech recognition and interpretation, which might in turn improve overall accuracy.

The requirement for incrementality is recognised by Schuler et al. (2010), who employ an incremental Hierarchical Hidden Markov Model (HHMM) to detect speech disfluencies. The HHMM is trained on manually annotated parse trees which are transformed by a right corner transformation; the HHMM is then used in an incremental fashion on unseen data, growing the parse structure each time a new token comes in. Special subtrees in this parse can carry a marker indicating that the span of the subtree consists of tokens corresponding to a speech disfluency. Schuler et al.'s approach thus provides scope for detecting disfluencies in an incremental fashion. However, their reported accuracy scores are not as good as those of Johnson and Charniak (2004): they report an F-score of 0.690 for their HHMM+RCT model, as compared to 0.797 for Johnson and Charniak's parser model.

Our aim in this paper, then, is to investigate whether it is possible to adapt Johnson and Charniak's model to process utterances incrementally, without any loss of accuracy. To define the incremental component more precisely, we investigate the possibility of marking the disfluencies as soon as possible during the processing of the input. Given two models that provide comparable accuracy measured on utterance completion, we would prefer a model which detects disfluencies earlier.

3 The Model

In this section, we describe Johnson and Charniak's (2004) noisy channel model, and show how this model can be made incremental.

As a data set to work with, we use the Switchboard part of the Penn Treebank 3 corpus. The Switchboard corpus is a corpus of spontaneous conversations between two parties. In Penn Treebank 3, the disfluencies are manually annotated. Following Johnson and Charniak (2004), we use all of sections 2 and 3 for training; we use conversations 4[5-9]* for a held-out training set; and conversations 40*, 41[0-4]* and 415[0-3]* as the held-out test set.

3.1 The Noisy Channel Model

To find the repair disfluencies a noisy channel model is used. For an observed utterance with disfluencies, y, we wish to find the most likely source utterance, $\hat{x}$, where:

$$\begin{aligned} \hat{x} &= argmax_x \; p(x \mid y) \qquad (2) \\ &= argmax_x \; p(y \mid x) \, p(x) \end{aligned}$$

Here we have a channel model $p(y|x)$ which generates an utterance y given a source x and a language model $p(x)$. We assume that x is a substring of y, i.e., the source utterance can be obtained by marking words in y as a disfluency and effectively removing them from this utterance.

Johnson and Charniak (2004) experiment with variations on the language model; they report results for a bigram model, a trigram model, and a language model using the Charniak Parser (Charniak, 2001). Their parser model outperforms the bigram model by 5%. The channel model is based on the intuition that a reparandum and a repair are generally very alike: a repair is often almost a copy of the reparandum. In the training data, over 60% of the words in a reparandum are lexically identical to the words in the repair. Example 1 provides an example of this: half of the repair is lexically identical to the reparandum. The channel model therefore gives the highest probability when the reparandum and repair are lexically equivalent. When the potential reparandum and potential repair are not identical, the channel model performs deletion, insertion or substitution. The probabilities for these operations are defined on a lexical level and are derived from the training data. This channel model is formalised using a Synchronous Tree Adjoining Grammar (STAG) (Shieber and Schabes, 1990), which matches words from the reparandum to the repair. The weights for these STAG rules are learnt from the training text, where reparanda and repairs are aligned to each other using a minimum edit-distance string aligner.

For a given utterance, every possible utterance position might be the start of a reparandum, and every given utterance position thereafter might be the start of a repair (to limit complexity, a maximum distance between these two points is imposed). Every disfluency in turn can have an arbitrary length (again up to some maximum to limit complexity). After every possible disfluency other new reparanda and repairs might occur; the model does not attempt to generate crossing or nested disfluencies, although they do very occasionally occur in practice. To find the optimal selection for reparanda and repairs, all possibilities are calculated and the one with the highest probability is selected. A chart is filled with all the possible start and end positions of reparanda, interregna and repairs; each entry consists of a tuple $\langle rm_{\text{begin}}, ir_{\text{begin}}, rr_{\text{begin}}, rr_{\text{end}}\rangle$, where rm is the reparandum, ir is the interregnum and rr is the repair. A Viterbi algorithm is used to find the optimal path through the utterance, ranking each chart entry using the language model and channel model. The language model, a bigram model, can be easily calculated given the start and end positions of all disfluency components. The channel model is slightly more complicated because an optimal alignment between reparandum and repair needs to be calculated. This is done by extending each partial analysis by adding a word to the reparandum, the repair or both. The start position and end position of the reparandum and repair are given for this particular entry. The task of the channel model is to calculate the highest probable alignment between reparandum and repair. This is done by initialising with an empty reparandum and repair, and 'growing' the analysis one word at a time. Using a similar approach to that used in calculating the edit-distance between reparandum and repair, the reparandum and repair can both be extended with one of four operations: deletion (only the reparandum grows), insertion (only the repair grows), substitution (both grow), or copy (both grow). When the reparandum and the repair have their length corresponding to the current entry in the chart, the channel probability can be calculated. Since there are multiple alignment possibilities, we use dynamic programming to select the most probable solutions. The probabilities for insertion, deletion or substitution are estimated from the training corpus. We use a beam-search strategy to find the final optimum when combining the channel model and the language model.

3.2 Incrementality

Taking Johnson and Charniak's model as a starting point, we would like to develop an incremental version of that algorithm. We simulate incrementality by maintaining for each utterance to be processed an **end-of-prefix boundary**; tokens after this boundary are not available for the model to use. At each step in our incremental model, we advance this boundary by one token (the **increment**), until finally the entire utterance is available. We make use of the notion of a **prefix**, which is a substring of the utterance consisting of all tokens up to this boundary marker.

Just as in the non-incremental model, we keep track of all the possible reparanda and repairs in a chart. Every time the end-of-prefix boundary advances, we update the chart: we add all possible disfluencies which have the end position of the repair located one token before the end-of-prefix boundary, and we add all possible start points for the reparandum, interregna and repair, and end points for the reparandum and interregna, given the ordering constraints of these components.

In our basic incremental model, we leave the remainder of the algorithm untouched. When the end-of-prefix boundary reaches the end of the utterance, and thus the entire utterance is available, this model results in an identical analysis to that provided by the non-incremental model, since the chart contains identical entries, although calculated in a different order. Intuitively, this model should perform well when the current prefix is very close to being a complete utterance; and it should perform less well when a potential disfluency is still under construction, since these situations are not typically found in the training data. We will return to this point further below.

We do not change the training phase of the model and we assume that the optimal values found for the non-incremental model are also optimal for the incremental model, since most weights which need to be learned are based on lexical values. Other weights are bigram based values, and values dealing with unknown tokens (i.e., tokens which occur in the test data, but not in the training data); it is not unreasonable to assume these weights are identical or very similar in both the incremental and the non-incremental model.

4 Evaluation Models and Their Application

As well as evaluating the accuracy of the analysis returned at the end of the utterance, it seems reasonable to also evaluate how quickly and accurately an incremental algorithm detects disfluencies on a word-by-word basis as the utterance is processed. In this section, we provide the methodological background to our approach, and in Section 5.2 we discuss the performance of our model when evaluated in this way.

Incremental systems are often judged solely on the basis of their output when the utterance being processed is completed. Although this does give an insight into how well a system performs overall, it does not indicate how well the incremental aspects of the mechanism perform. In this section we present an approach to the evaluation of a model of speech repair detection which measures the performance of the incremental component.

One might calculate the accuracy over all prefixes using a simple word accuracy score. However, because each prefix is a superstring of each previous prefix, such a calculation would not be fair: tokens that appear in early in the utterance will be counted more often than tokens that appear later in the utterance. In theory, the analysis of the early tokens can change at each prefix, so arguably it would make sense to reevaluate the complete analysis so far at every step. In practice, however, these changes do not happen, and so this measurement would not reflect the performance of the system correctly.

Our approach is to define a measure of **responsiveness**: that is, how soon is a disfluency detected? We propose to measure responsiveness in two ways. The **time-to-detection** score indicates how many tokens following a disfluency are read before the given disfluency is marked as one; the **delayed accuracy** score looks n tokens back from the boundary of the available utterance and, when there is a gold standard disfluency-marked token at that distance, counts how often these tokens are marked correctly.

We measure the time-to-detection score by two numbers, corresponding to the number of tokens from the start of the reparandum and the number of tokens from the start of the repair. We do this because disfluencies can be of different lengths. We assume it is unlikely that a disfluency will be found before the reparandum is completed, since the reparandum itself is often fluent. We measure the time-to-detection by the first time a given disfluency appears as one.

Since the model is a statistical model, it is possible that the most probable analysis marks a given word at position j as a disfluency, while in the next prefix the word in the same position is now no longer marked as being disfluent. A prefix later this word might be marked as disfluent again. This presents us with a problem. How do we measure when this word was correctly identified as disfluent: the first time it was marked as such or the second time? Because of the possibility of such oscillations, we take the first marking of the disfluency as the measure point. Disfluencies which are never correctly detected are not part of the time-to-detection score.

Since the evaluation starts with disfluencies found by the model, this measurement has precision-like properties only. Consequently, there are easy ways to inflate the score artificially at the cost of recall. We address this by also calculating the delayed accuracy. This is calculated at each prefix by looking back n tokens from the prefix boundary, where $n = 0$ for the prefix boundary. For each n we calculate the accuracy score at that point over all prefixes. Each token is only assessed once given a set value of n, so we do not suffer from early prefixes being assessed more often. However, larger values of n do not take all tokens into account, since the last y tokens of an utterance will not play a part in the accuracy when $y < n$. Since we evaluate given a gold standard disfluency, this measurement has recall-like properties.

Together with the final accuracy score over the entire utterance, the time-to-detection and delayed accuracy scores provide different insights and together give a good measurement of the responsiveness and performance of the model.

Our incremental model has the same final accuracy as the original non-incremental model; this corresponds to an F-score (harmonic mean) of 0.778 on a word basis.

We found the average time to detection, measured in tokens for this model to be 8.3 measured from the start of reparandum and 5.1 from the start of repair. There are situations where disfluencies can be detected before the end of the repair; by counting from the start rather than the end of the disfluency components, we provide a way of scoring in such cases. To provide a better insight into what is happening, we also report the average distance since the start of the reparandum. We find that the time to detect is larger than the average repair length; this implies that, under this particular model, most disfluencies are only detected after the repair is finished. In fact the difference is greater than 1, which means that in most cases it takes one more token after the repair before the model identifies the disfluency.

Table 1 shows the delayed accuracy. We can see that the score first rises quickly after which the increases become much smaller. As mentioned above, a given disfluency detection in theory might oscillate. In practice, however, oscillating disfluencies are very rare, possibly because a bigram model operates on a very local level. Given that oscillation is rare, a quick stabilisation of the score indicates that, when we correctly detect a disfluency, this happens rather quickly after the disfluency has completed, since the accuracy for the large n is calculated over the same tokens as the accuracy for the smaller n (although not in the same prefix).

5 Disfluencies around Prefix Boundaries

5.1 Early detection algorithm

Our model uses a language model and a channel model to locate disfluencies. It calculates a language model probability for the utterance with the disfluency taken out, and it calculates the probability of the disfluency itself with the STAG channel model.

Consider the following example utterance fragment where a repair disfluency occurs:

$$\dots w_i \ \overbrace{rn_{i+1}\ rn_{i+2}}^{\text{reparandum}}\ \overbrace{rr_{i+3}\ rr_{i+4}}^{\text{repair}}\ w_{i+5} \dots \tag{3}$$

Here, the subscripts indicate token position in sequence; w is a token outside the disfluency; and rn is a reparandum being repaired by the repair rr. The language model estimates the continuation of the utterance without the disfluency. The model considers whether the utterance continuation after the disfluency is probable given the language model; the relevant bigram here is $p(rr_{i+3}|w_i)$, continuing with $p(rr_{i+4}|rr_{i+3})$. However, under the incremental model, it is possible the utterance has only been read as far as token $i + 3$, in which case the probability $p(w_{i+4}|w_{i+3})$ is undefined.

We would like to address the issue of looking beyond a disfluency under construction. We assume the issue of not being able to look for an utterance continuation after the repair component of the disfluency can be found back in the incremental model scores. A disfluency is usually only detected after the disfluency is completely uttered, and always requires one

n tokens back	**1**	**2**	**3**	**4**	**5**	**6**
accuracy	0.500	0.558	0.631	0.665	0.701	0.714

Table 1: delayed accuracy, n tokens back from the end of prefixes

n tokens back	**1**	**2**	**3**	**4**	**5**	**6**
accuracy	0.578	0.633	0.697	0.725	0.758	0.770

Table 2: delayed accuracy under the updated model

more token in the basic model. In the given instance this means it is unlikely that we will detect the disfluency before $i + 5$.

In order to make our model more responsive, we propose a change which makes it possible for the model to calculate channel probabilities and language model probabilities before the repair is completed. Assuming we have not yet reached the end of utterance, we would like to estimate the continuation of the utterance with the relevant bigram $p(rr_{i+4}|rr_{i+3})$. Since rr_{i+4} is not yet available we cannot calculate this probability. The correct thing to do is to sum over all possible continuations, including the end of utterance token (for the complete utterance, as opposed to the current prefix). This results in the following bigram estimation:

$$\sum_{t \in \text{vocabulary}} p(t|w_i) \tag{4}$$

This estimation is not one we need to derive from our data set, since p is a true probability. In this case, the sum over all possible continuations (this might include an end of utterance marker, in which case the utterance is already complete) equals 1. We therefore modify the algorithm so that it takes this into account. This solves the problem of the language model assessing the utterance with the disfluency cut out, when nothing from the utterance continuation after a disfluency is available.

The other issue which needs to be addressed is the alignment of the reparandum with the repair when the repair is not yet fully available. Currently the model is encouraged to align the individual tokens of the reparandum with those of the repair. The algorithm has lower estimations when the reparandum cannot be fully aligned with the repair because the reparandum and repair differ considerably in length.

We note that most disfluencies are very short: reparanda and repairs are often only one or two tokens each in length, and the interregnum is often empty. To remove the penalty for an incomplete repair, we allow the repair to grow one token beyond the prefix boundary; given the relative shortness of the disfluencies, this seems reasonable. Since this token is not available, we cannot calculate the lexical substitution value. Instead we define a new operation in the channel model: in addition to deletion, insertion, copy, and substitution, we add an additional substitution operation, the **incremental completion substitution**. This operation does not compete with the copy operation or the normal substitution operation, since it is only defined when the last token of the repair falls at the prefix boundary.

5.2 Results for the Early detection algorithm

The results of these changes are reflected in new time-to-detection and delayed accuracy scores. Again we calculated the time-to-detection, and found this to be 7.5 from the start of reparandum and 4.6 from the start of repair. Table 2 shows the results under the new early completion model using the delayed accuracy method. We see that the updated model has lower time-to-detection scores (close to a full token earlier); for delayed accuracy, we note that the scores stabilise in a similar fashion, but the scores for the updated model rise slightly more quickly.

6 Conclusions and Future Work

We have demonstrated an incremental model for finding speech disfluencies in spoken language transcripts. When we consider complete utterances, the incremental model provides identical results to those of a non-incremental model that delivers state-of-the-art accuracy in speech repair detection. We have investigated a number of measures which allow us to evaluate the model on an incremental level. Most disfluencies are identified very quickly, typically one or two tokens after the disfluency has been completed. We addressed the problems of the model around the end of prefix boundaries. These are repairs which are either still in the process of being uttered or have just been completed. We have addressed this issue by making some changes to how the model deals with prefix boundaries, and we have shown that this improves the responsiveness of the model.

The work reported in this paper uses a n-gram model as a language model and a STAG based model for the repair. We would like to replace the n-gram language model with a better language model. Previous work (Johnson and Charniak, 2004) has shown that disfluency detection can be improved by replacing the n-gram language model with a statistical parser. Besides a reported 5% accuracy improvement, this also provides a structural analysis, something which an n-gram model does not. We would like to investigate a similar extension in our incremental approach, which will require the integration of an incremental statistical parser with our noisy channel model. While transcripts of spoken texts come with manually annotated sentence boundaries, real time spoken language does not. The language model in particular takes these sentence boundaries into account. We therefore propose to investigate the properties of this model when sentence boundaries are removed.

Acknowledgements

This work was supported by the Australian Research Council as part of the Thinking Head Project, ARC/NHMRC Special Research Initiative Grant # TS0669874. We thank the anonymous reviewers for their helpful comments.

References

Charniak, Eugene. 2001. Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 124–131.

Core, Mark and Lenhart Schubert. 1999. A model of speech repairs and other disruptions. In *AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*, pages 48–53.

Honal, Matthias and Tanja Schultz. 2003. Correction of Disfluencies in Spontaneous Speech using a Noisy-Channel Approach. In *Proceedings of the 8th Eurospeech Conference.*

Johnson, Mark and Eugene Charniak. 2004. A tag-based noisy channel model of speech repairs. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 33–39.

Marslen-Wilson, W. 1973. Linguistic structure and speech shadowing at very short latencies. *Nature*, 244:522–533.

Schuler, William, Samir AbdelRahman, Tim Miller, and Lane Schwartz. 2010. Broad-Coverage Parsing using Human-Like Memory Constraints. *Computational Linguistics*, 36(1):1–30.

Shieber, Stuart M. and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, pages 253–258.

Shriberg, Elizabeth. 1994. *Preliminaries to a Theory of Speech Disuencies.* Ph.D. thesis, University of California, Berkeley.

Syntactic Scope Resolution in Uncertainty Analysis

Lilja Øvrelid♠♣ and **Erik Velldal**♣ and **Stephan Oepen**♣
♣ University of Oslo, Department of Informatics
♠Universität Potsdam, Institut für Linguistik
ovrelid@uni-potsdam.de and erikve@ifi.uio.no and oe@ifi.uio.no

Abstract

We show how the use of syntactic structure enables the resolution of hedge scope in a hybrid, two-stage approach to uncertainty analysis. In the first stage, a Maximum Entropy classifier, combining surface-oriented and syntactic features, identifies cue words. With a small set of hand-crafted rules operating over dependency representations in stage two, we attain the best overall result (in terms of both combined ranks and average F_1) in the 2010 CoNLL Shared Task.

1 Background—Motivation

Recent years have witnessed an increased interest in the analysis of various aspects of sentiment in natural language (Pang & Lee, 2008). The subtask of *hedge resolution* deals with the analysis of uncertainty as expressed in natural language, and the linguistic means (so-called hedges) by which speculation or uncertainty are expressed. Information of this kind is of importance for various mining tasks which aim at extracting factual data. Example (1), taken from the BioScope corpus (Vincze, Szarvas, Farkas, Móra, & Csirik, 2008), shows a sentence where uncertainty is signaled by the modal verb *may*.[1]

(1) {The unknown amino acid ⟨may⟩ be used by these species}.

The topic of the Shared Task at the 2010 Conference for Natural Language Learning (CoNLL) is hedge detection in biomedical literature—in a sense 'zooming in' on one particular aspect of the broader BioNLP Shared Task in 2009 (Kim, Ohta, Pyysalo, Kano, & Tsujii, 2009). It involves two subtasks: Task 1 is described as *learning to detect sentences containing uncertainty*; the objective of Task 2 is *learning to resolve the in-sentence scope of hedge cues* (Farkas, Vincze, Mora, Csirik, & Szarvas, 2010). The organizers further suggest: *This task falls within the scope of semantic analysis of sentences exploiting syntactic patterns [...].*

The utility of syntactic information within various approaches to sentiment analysis in natural language has been an issue of some debate (Wilson, Wiebe, & Hwa, 2006; Ng, Dasgupta, & Arifin, 2006), and the potential contribution of syntax clearly varies with the specifics of the task. Previous work in the hedging realm has largely been concerned with cue detection, i.e. identifying uncertainty cues such as *may* in (1), which are predominantly individual tokens (Medlock & Briscoe, 2007; Kilicoglu & Bergler, 2008). There has been little previous work aimed at actually resolving the scope of such hedge cues, which presumably constitutes a somewhat different and likely more difficult problem. Morante and Daelemans (2009) present a machine-learning approach to this task, using token-level, lexical information only. To this end, CoNLL 2010 enters largely uncharted territory, and it remains to be seen (a) whether syntactic analysis indeed is a necessary component in approaching this task and, more generally, (b) to what degree the specific task setup can inform us about the strong and weak points in current approaches and technology.

In this article, we investigate the contribution of syntax to hedge resolution, by reflecting on our experience in the CoNLL 2010 task.[2] Our CoNLL system submission ranked fourth (of 24) on Task 1 and third (of 15) on Task 2, for an overall best average result (there appears to be very limited overlap among top performers for the two subtasks).

[1] In examples throughout this paper, angle brackets highlight hedge cues, and curly braces indicate the scope of a given cue, as annotated in BioScope.

[2] It turns out, in fact, that all the top-performing systems in Task 2 of the CoNLLShared Task rely on syntactic information provided by parsers, either in features for machine learning or as input to manually crafted rules (Morante, Asch, & Daelemans, 2010; Rei & Briscoe, 2010).

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1379–1387,
Beijing, August 2010

	Sentences	Hedged Sentences	Cues	Multi-Word Cues	Tokens	Cue Tokens
Abstracts	11871	2101	2659	364	309634	3056
Articles	2670	519	668	84	68579	782
Total	14541	2620	3327	448	378213	3838

Table 1: Summary statistics for the Shared Task training data.

This article transcends our CoNLL system description (Velldal, Øvrelid, & Oepen, 2010) in several respects, presenting updated and improved cue detection results (§ 3 and § 4), focusing on the role of syntactic information rather than on machine learning specifics (§ 5 and § 6), providing an analysis and discussion of Task 2 errors (§ 7), and generally aiming to gauge the value of available annotated data and processing tools (§ 8). We present a hybrid, two-level approach for hedge resolution, where a statistical classifier detects cue words, and a small set of manually crafted rules operating over syntactic structures resolve scope. We show how syntactic information—produced by a data-driven dependency parser complemented with information from a 'deep', hand-crafted grammar—contributes to the resolution of in-sentence scope of hedge cues, discussing various types of syntactic constructions and associated scope detection rules in considerable detail. We furthermore present a manual error analysis, which reveals remaining challenges in our scope resolution rules as well as several relevant idiosyncrasies of the preexisting BioScope annotation.

2 Task, Data, and System Basics

Task Definition and Evaluation Metrics Task 1 is a binary sentence classification task: identifying utterances as being *certain* or *uncertain*. Following common practice, this subtask is evaluated in terms of precision, recall, and F_1 for the 'positive' class, i.e. *uncertain*. In our work, we approach Task 1 as a byproduct of the full hedge resolution problem, labeling a sentence as *uncertain* if it contains at least one token classified as a hedge cue. In addition to the sentence-level evaluation for Task 1, we also present precision, recall, and F_1 for the cue-level.

Task 2 comprises two subtasks: cue detection and scope resolution. The official CoNLL evaluation does not tease apart these two aspects of the problem, however: Only an exact match of both the cue and scope bracketing (in terms of substring positions) will be counted as a success, again quantified in terms of precision, recall, and F_1. Discussing our results below, we report cue detection and scope resolution performance separately, and further put scope results into perspective against an upper bound based on the gold-standard cue annotation.

Besides the primary biomedical domain data, some annotated Wikipedia data was provided for Task 1, and participating systems are classified as *in-domain* (using exclusively the domain-specific data), *cross-domain* (combining both types of training data), or *open* (utilizing additional uncertainty-related resources). In our work, we focus on the interplay of syntax and the more challenging Task 2; we ignored the Wikipedia track in Task 1. Despite our using general NLP tools (see below), our system falls into the most restrictive, *in-domain* category.

Training and Evaluation Data The training data for the CoNLL 2010 Shared Task is taken from the BioScope corpus (Vincze et al., 2008) and consists of 14,541 'sentences' (or other root-level utterances) from biomedical abstracts and articles (see Table 1).[3] The BioScope corpus provides annotation for hedge cues as well as their scope. According to the annotation guidelines (Vincze et al., 2008), the annotation adheres to a principle of minimalism when it comes to hedge cues, i.e. the minimal unit expressing hedging is annotated. The inverse is true of scope annotations, which adhere to a principle of maximal scope—meaning that scope should be set to the largest syntactic

[3] As it was known beforehand that evaluation would draw on full articles only, we put more emphasis on the article subset of the training data, for example in cross validation testing and manual diagnosis of errors.

ID	FORM	LEMMA	POS	FEATS	HEAD	DEPREL	XHEAD	XDEP
1	The	the	DT	_	4	NMOD	4	SPECDET
2	unknown	unknown	JJ	degree:attributive	4	NMOD	4	ADJUNCT
3	amino	amino	JJ	degree:attributive	4	NMOD	4	ADJUNCT
4	acid	acid	NN	pers:3|case:nom|num:sg|ntype:common	5	SBJ	3	SUBJ
5	may	may	MD	mood:ind|subcat:MODAL|tense:pres|clauseType:decl	0	ROOT	0	ROOT
6	be	be	VB	_	5	VC	7	PHI
7	used	use	VBN	subcat:V-SUBJ-OBJ|vtype:main|passive:+	6	VC	5	XCOMP
8	by	by	IN	_	7	LGS	9	PHI
9	these	these	DT	deixis:proximal	10	NMOD	10	SPECDET
10	species	specie	NNS	num:pl|pers:3|case:obl|common:count|ntype:common	8	PMOD	7	OBL-AG
11	.	.	.	_	5	P	0	PUNC

Table 2: Stacked dependency representation of example (1), with MaltParser and XLE annotations.

unit possible.

For evaluation purposes, the task organizers provided newly annotated biomedical articles, following the same general BioScope principles. The CoNLL 2010 evaluation data comprises 5,003 additional utterances (138,276 tokens), of which 790 are annotated as hedged. The data contains a total of 1033 cues, of which 87 are so-called multiword cues (i.e. cues spanning multiple tokens), comprising 1148 cue tokens altogether.

Stacked Dependency Parsing For syntactic analysis we employ the open-source MaltParser (Nivre, Hall, & Nilsson, 2006), a platform for data-driven dependency parsing. For improved accuracy and portability across domains and genres, we make our parser incorporate the predictions of a large-scale, general-purpose LFG parser—following the work of Øvrelid, Kuhn, and Spreyer (2009). A technique dubbed *parser stacking* enables the data-driven parser to learn, not only from gold standard treebank annotations, but from the output of another parser (Nivre & McDonald, 2008). This technique has been shown to provide significant improvements in accuracy for both English and German (Øvrelid et al., 2009), and a similar setup employing an HPSG grammar has been shown to increase domain independence in data-driven dependency parsing (Zhang & Wang, 2009). The stacked parser combines two quite different approaches—data-driven dependency parsing and 'deep' parsing with a handcrafted grammar—and thus provides us with a broad range of different types of linguistic information for the hedge resolution task.

MaltParser is based on a deterministic parsing strategy in combination with treebank-induced classifiers for predicting parse transitions. It supports a rich feature representation of the parse history in order to guide parsing and may easily be extended to take additional features into account. The procedure to enable the data-driven parser to learn from the grammar-driven parser is quite simple. We parse a treebank with the XLE platform (Crouch et al., 2008) and the English grammar developed within the ParGram project (Butt, Dyvik, King, Masuichi, & Rohrer, 2002). We then convert the LFG output to dependency structures, so that we have two parallel versions of the treebank—one gold standard and one with LFG annotation. We extend the gold standard treebank with additional information from the corresponding LFG analysis and train MaltParser on the enhanced data set.

Table 2 shows the enhanced dependency representation of example (1) above, taken from the training data. For each token, the parsed data contains information on the word form, lemma, and part of speech (PoS), as well as on the head and dependency relation in columns 6 and 7. The added XLE information resides in the FEATS column, and in the XLE-specific head and dependency columns 8 and 9. Parser outputs, which in turn form the basis for our scope resolution rules discussed in Section 5, also take this same form. The parser employed in this work is trained on the Wall Street Journal sections 2 – 24 of the Penn Treebank (PTB), converted to dependency format (Johansson & Nugues, 2007) and extended with XLE features, as described above. Parsing uses the arc-eager mode of MaltParser and an SVM with a polynomial kernel. When tested using 10-fold cross validation on the enhanced PTB, the parser achieves a labeled accuracy score of 89.8.

PoS Tagging and Domain Variation Our parser is trained on financial news, and although stacking with a general-purpose LFG parser is expected to aid domain portability, substantial differences in domain and genre are bound to negatively affect syntactic analysis (Gildea, 2001). MaltParser presupposes that inputs have been PoS tagged, leaving room for variation in preprocessing. On the one hand, we aim to make parser inputs maximally similar to its training data (i.e. the conventions established in the PTB); on the other hand we wish to benefit from specialized resources for the biomedical domain.

The GENIA tagger (Tsuruoka et al., 2005) is particularly relevant in this respect (as could be the GENIA Treebank proper[4]). However, we found that GENIA tokenization does not match the PTB conventions in about one out of five sentences (for example wrongly splitting tokens like '`390,926`' or '`Ca(2+)`'); also in tagging proper nouns, GENIA systematically deviates from the PTB. Hence, we adapted an in-house tokenizer (using cascaded finite-state rules) to the CoNLL task, run two PoS taggers in parallel, and eclectically combine annotations across the various preprocessing components—predominantly giving precedence to GENIA lemmatization and PoS hypotheses.

To assess the impact of improved, domain-adapted inputs on our hedge resolution system, we contrast two configurations: first, running the parser in the exact same manner as Øvrelid, Kuhn, and Spreyer (2010), we use TreeTagger (Schmid, 1994) and its standard model for English (trained on the PTB) for preprocessing; second, we give as inputs to the parser our refined tokenization and merged PoS tags, as described above. When evaluating the two modes of preprocessing on the articles subset of the training data, and using gold-standard cues, our system for resolving cue scopes (presented in § 5) achieves an F_1 of 66.31 with TreeTagger inputs, and 72.30 using our refined tokenization and tagger combination. These results underline the importance of domain adaptation for accurate syntactic analysis, and in the following we assume our hybrid in-house setup.

3 Stage 1: Identifying Hedge Cues

For the task of identifying hedge cues, we developed a binary maximum entropy (MaxEnt) classifier. The identification of cue words is used for (a) classifying sentences as certain/uncertain (Task 1), and (b) providing input to the syntactic rules that we later apply for resolving the in-sentence scope of the cues (Task 2). We also report evaluation scores for the sub-task of cue detection in isolation.

As annotated in the training data, it is possible for a hedge cue to span multiple tokens, e.g. as in *whether or not*. The majority of the multi-word cues in the training data are very infrequent, however, most occurring only once, and the classifier itself is not sensitive to the notion of multi-word cues. Instead, the task of determining whether a cue word forms part of a larger multi-word cue, is performed in a separate post-processing step (applying a heuristic rule targeted at only the most frequently occurring patterns of multi-word cues in the training data).

During development, we trained cue classifiers using a wide variety of feature types, both syntactic and surface-oriented. In the end, however, we found n-gram-based lexical features to have the greatest contribution to classifier performance. Our best-performing classifier so far (see 'Final' in Table 3) includes the following feature types: n-grams over forms (up to 2 tokens to the right), n-grams over base forms (up to 3 tokens left and right), PoS (from GENIA), subcategorization frames (from XLE), and phrase-structural coordination level (from XLE). Our CoNLL system description includes more details of the various other feature types that we experimented with (Velldal et al., 2010).

4 Cue Detection Evaluation

Table 3 summarizes the performance of our MaxEnt hedge cue classifier in terms of precision, recall and F_1, computed using the official Shared Task scorer script. The sentence-level scores correspond to Task 1 of the Shared Task, and the cue-level scores are based on the exact-match counts for full hedge cues (possibly spanning multiple tokens).

[4] Although the GENIA Treebank provides syntactic annotation in a form inspired by the PTB, it does not provide function labels. Therefore, our procedure for converting from constituency to dependency requires non-trivial adaptation before we can investigate the effects of retraining the parser against GENIA.

	Sentence Level			Cue Level		
Configuration	**Prec**	**Rec**	$\mathbf{F}_1$	**Prec**	**Rec**	$\mathbf{F}_1$
Baseline, Development	79.25	79.45	79.20	77.37	71.70	74.43
Final, Development	91.39	86.78	89.00	90.18	79.47	84.49
Final, Held-Out	85.61	85.06	85.33	81.97	76.41	79.10

Table 3: Isolated evaluation of the hedge cue classifier.

As the CoNLL test data was known beforehand to consist of articles only, in 10-fold cross validation for classifier development we tested exclusively against the articles segment, while always including all sentences from the abstracts in the training set. This corresponds to the development results in Table 3, while the held-out results are for the official Shared Task evaluation data (training on all the available training data). A model using only unigram features serves as a baseline.

5 Stage 2: Resolving Scope

Hedge scope may vary quite a lot depending on linguistic properties of the cue in question. In our approach to scope resolution we rely heavily on syntactic information, taken from the dependency structures proposed by both MaltParser and XLE, as well as on various additional features relating to specific syntactic constructions.

We constructed a small set of heuristic rules which define the scope for each cue detected in Stage 1. In developing these rules, we made use of the information provided by the guidelines for scope annotation in the BioScope corpus (Vincze et al., 2008), combined with manual inspection of the training data in order to further generalize over the phenomena discussed by Vincze et al. (2008) and work out interactions of constructions for various types of cues.

The rules take as input a parsed sentence which has been further tagged with hedge cues. They operate over the dependency structures and additional features provided by the parser. Default scope is set to start at the cue word and span to the end of the sentence (modulo punctuation), and this scope also provides the baseline for the evaluation of our rules. In the following, we discuss broad classes of rules, organized by categories of hedge cues. As there is no explicit representation of phrase or clause boundaries in our dependency universe, we assume a set of functions over dependency graphs, for example finding the left- or rightmost (direct) *dependent* of a given node, or transitively selecting left- or rightmost *descendants*.

Coordination The dependency analysis of coordination provided by our parser makes the first conjunct the head of the coordination. For cues that are coordinating conjunctions (PoS tag CC), such as *or*, we define the scope as spanning the whole coordinate structure, i.e. start scope is set to the leftmost dependent of the head of the coordination, e.g., *roX* in (2), and end scope is set to its rightmost dependent (conjunct), e.g., *RNAs* in (2). This analysis provides us with coordinations at various syntactic levels, such as NP and $\overline{\text{N}}$ (2), AP and AdvP, or VP (3):

(2) [...] the {roX genes ⟨or⟩ RNAs} recruit the entire set of MSL proteins [...]

(3) [...] the binding interfaces are more often {kept ⟨or⟩ even reused} rather than lost in the course of evolution.

Adjectives We distinguish between adjectives (JJ) in *attributive* (NMOD) function and adjectives in *predicative* (PRD) function. Attributive adjectives take scope over their (nominal) head, with all its dependents, as in (4) and (5):

(4) The {⟨possible⟩ selenocysteine residues} are shown in red, [...]

(5) Extensive analysis of the flanks failed to show any hallmarks of {⟨putative⟩ transposons that might be associated with this RAG1-like protein}, [...]

For adjectives in a predicative function the scope includes the subject argument of the head verb (the copula), as well as a (possible) clausal argument, as in (6). The scope does not, however, include expletive subjects, as in (7).

(6) Therefore, {the unknown amino acid, if it is encoded by a stop codon, is ⟨unlikely⟩ to exist in the current databases of microbial genomes}.

(7) For example, it is quite {⟨likely⟩ that there exists an extremely long sequence that is entirely unique to U}.

Verbs The scope of verbal cues is a bit more complex and depends on several factors. In our rules, we distinguish *passive* usages from active usages, *raising* verbs from non-raising verbs, and the presence or absence of a subject-control embedding context. The scopes of both passive and raising verbs include the subject argument of their head verb, as in (8) and (9), unless it is an expletive pronoun, as in (10).

(8) {Interactions determined by high-throughput methods are generally ⟨considered⟩ to be less reliable than those obtained by low-throughput studies} 1314 and as a consequence [...]

(9) {Genomes of plants and vertebrates ⟨seem⟩ to be free of any recognizable Transib transposons} (Figure 1).

(10) It has been {⟨suggested⟩ that unstructured regions of proteins are often involved in binding interactions, particularly in the case of transient interactions} 77.

In the case of subject control involving a hedge cue, specifically modals, subject arguments are included in scopes where the controller heads a passive construction or a raising verb, as in example (1) above, repeated here for convenience:

(11) {The unknown amino acid ⟨may⟩ be used by these species}.

In general, the end scope of verbs should extend over the minimal clause that contains the verb in question. In terms of dependency structures, we define the clause boundary as comprising the chain of descendants of a verb which is not intervened by a token with a higher attachment in the graph than the verb in question. In example (8) for instance, the sentence-level conjunction *and* marks the end of the clause following the cue *considered*.

Prepositions and Adverbs Cues that are tagged as prepositions (including some complementizers) take scope over their argument, with all its descendants, (12). Adverbs take scope over their head with all its (non-subject) syntactic descendants (13).

	Configuration	**F$_1$**
BSP	Default, Gold Cues	45.21
	Rules, Gold Cues	72.31
	Rules, System Cues	64.77
BSE	Rules, Gold Cues	66.73
	Rules, System Cues	55.75

Table 4: Evaluation of scope resolution rules.

(12) {⟨Whether⟩ the codon aligned to the inframe stop codon is a nonsense codon or not} was neglected at this stage.

(13) These effects are {⟨probably⟩ mediated through the 1,25(OH)2D3 receptor}.

Multi-Word Cues In the case of multi-word cues, such as *indicate that* or *either ... or*, we need to determine the head of the multi-word unit. We then set the scope of the whole unit to the scope of the head token.

As an illustration of rule processing, consider our running example (11), with its syntactic analysis as shown in Table 2 above. This example invokes a variety of syntactic properties, including parts of speech, argumenthood, voice etc. Initially, the scope of the hedge cue is set to default scope. Then the subject control rule is applied, which checks the properties of the verbal argument *used*, going through a chain of verbal dependents from the modal verb. Since it is marked as passive in the LFG analysis, the start scope is set to include the subject of the cue word (the leftmost descendant in its `SBJ` dependent).

6 Rule Evaluation

Table 4 summarizes scope resolution performance (viewed as a an isolated subtask) for various configurations, both against the articles section of the CoNLL training data (dubbed BSP) and against the held-out evaluation data (BSE). First of all, we note that the ‘default scope’ baseline is quite strong: unconditionally extending the scope of a cue to the end of the sentence yields an F$_1$ of 45.21. Given gold standard cue information, our scope rules improve on the baseline by 27 points on the articles section of the data set, for an F$_1$ of 72.31; with system-assigned hedge cues, our rules still achieve an F_1 of 64.77. Note that scope resolution scores based on classified cues also yield the end-to-end system evaluation for Task 2.

The bottom rows of Table 4 show the evaluation of scope rules on the CoNLL held-out test data. Using system cues, scope resolution on the held-out data scores at 55.75 F_1. Comparing to the result on the (articles portion of the) training data, we observe a substantial drop in performance (of six points with gold-standard cues, nine points with system cues). There are several possible explanations for this effect. First of all, there may well be a certain degree of overfitting of our rules to the training data. The held-out data may contain hedging constructions that are not covered by our current set of scope rules, or annotation of parallel constructions may in some cases differ in subtle ways (see § 7 below). Moreover, scope resolution performance is of course influenced by cue detection (see Table 3). The cue-level F_1 of our system on the held-out data set is 79.10, compared to 84.49 (using cross validation) on the training data. This drop in cue-level performance appears to affect classification precision far more than recall. Of course, given that our heuristics for identifying multi-word cues were based on patterns extracted from the training data, some loss in the cue-level score was expected.

7 Error Analysis

To start shedding some light on the significance of our results, we performed a manual error analysis on the article portion of the training material (BSP), with two of the authors (trained linguists) working in tandem. Using gold-standard cues, our scope resolution rules fail to exactly replicate the target annotation in 185 (of 668) cases, corresponding to 72.31 F_1 in Table 4 above. Our evaluators reviewed and discussed these 185 cases, classifying 156 (84 %) as genuine system errors, 22 (12 %) as likely[5] annotation errors, and a remaining seven cases as involving controversial or seemingly arbitrary decisions.

The two most frequent classes of system errors pertain (a) to the recognition of phrase and clause boundaries and (b) to not dealing successfully with relatively superficial properties of the text. Examples (14) and (15) illustrate the first class of errors, where in addition to the gold-standard annotation we use vertical bars (‘|’) to indicate scope predictions of our system.

(14) [...] {the reverse complement |mR of m will be ⟨considered⟩ to be [...]|}

(15) This |{⟨might⟩ affect the results} if there is a systematic bias on the composition of a protein interaction set|.

In our syntax-driven approach to scope resolution, system errors will almost always correspond to a failure in determining constituent boundaries, in a very general sense. However, specifically example (15) is indicative of a key challenge in this task, where adverbials of condition, reason, or contrast frequently attach within the dependency domain of a hedge cue, yet are rarely included in the scope annotation.

Example (16) demonstrates our second frequent class of system errors. One in six items in the BSP training data contains a sentence-final parenthesized element or trailing number, as for example (2), (9), or (10) above; most of these are bibliographic or other in-text references, which are never included in scope annotation. Hence, our system includes a rule to ‘back out’ from trailing parentheticals; in examples like (16), however, syntax does not make explicit the contrast between an in-text reference vs. another type of parenthetical.

(16) More specifically, {|the bristle and leg phenotypes are ⟨likely⟩ to result from reduced signaling by Dl| (and not by Ser)}.

Moving on to apparent annotation errors, the rules for inclusion (or not) of the subject in the scope of verbal hedge cues and decisions on boundaries (or internal structure) of nominals

[5] In some cases, there is no doubt that annotation is erroneous, i.e. in violation of the available annotation guidelines (Vincze et al., 2008) or in conflict with otherwise unambiguous patterns. In other cases, however, judgments are necessarily based on generalizations made by the evaluators, i.e. assumptions about the underlying system and syntactic analyses implicit in the BioScope annotations. Furthermore, selecting items for manual analysis that do not align with the predictions made by our scope resolution rules is likely to bias our sample, such that our estimated proportion of 12 % annotation errors cannot be used to project an overall error rate.

seem problematic—as illustrated in examples (17) to (22).[6]

(17) [...] and |this is also {⟨thought⟩ to be true for the full protein interaction networks we are modeling}|.

(18) [...] {Neur |⟨can⟩ promote Ser signaling|}.

(19) |Some of the domain pairs {⟨seem⟩ to mediate a large number of protein interactions, thus acting as reusable connectors}|.

(20) One {|⟨possible⟩ explanation| is functional redundancy with the mouse Neur2 gene}.

(21) [...] |redefinition of {one of them is ⟨feasible⟩}|.

(22) |The {Bcl-2 family ⟨appears⟩ to function [...]}|.

Finally, the difficult corner cases invoke non-constituent coordination, ellipsis, or NP-initial focus adverbs—and of course interactions of the phenomena discussed above. Without making the syntactic structures assumed explicit, it is often very difficult to judge such items.

8 Reflections — Outlook

Our combination of stacked dependency parsing and hand-crafted scope resolution rules proved adequate for the CoNLL 2010 competition, confirming the central role of syntax in this task. With a comparatively small set of rules (implemented in a few hundred lines of code), constructed through roughly two full weeks of effort (studying BioScope annotations and developing rules), our CoNLL system achieved an end-to-end F_1 of 55.33 on Task 2.[7] The two submissions with better results (at 57.32 and 55.65 F_1) represent groups who have pioneered the hedge analysis task in previous years (Morante et al., 2010; Rei & Briscoe, 2010). Scores for other 'in-domain' participants range from 52.24 to 2.15 F_1. Doubtless there is room for straightforward extension: for example retraining our parser on the GENIA Treebank, further improving the cue classifier, and refining scope resolution rules in the light of the error analysis above.

At the same time, we remain mildly ambivalent about the long-term impact of some of the specifics of the 2010 CoNLL task. Shared tasks (i.e. system bake-offs) have become increasingly popular in past years, and in some subfields (e.g. IE, SMT, or dependency parsing) high-visibility competitions can shape community research agendas. Hence, even at this early stage, it seems appropriate to reflect on the possible conclusions to be drawn from the 2010 hedge resolution task. First, we believe the harsh 'exact substring match' evaluation metric underestimates the degree to which current technology can solve this problem; furthermore, idiosyncratic, string-level properties (e.g. the exact treatment of punctuation or parentheticals) may partly obscure the interpretation of methods used and corresponding system performance.

These effects are compounded by some concerns about the quality of available annotation. Even though we tried fine-tuning our cross validation testing to the nature of the evaluation data (comprising only articles), our system performs substantially worse on the newly annotated CoNLL test data, in both stages.[8] In our view, the annotation of hedge cues and scopes ideally would be overtly related to at least some level of syntactic annotation—as would in principle be possible for the segment of BioScope drawing on the abstracts of the GENIA Treebank.

Acknowledgements

We are grateful to the organizers of the 2010 CoNLL Shared Task and creators of the BioScope resource; first, for engaging in these kinds of community service, and second for many in-depth discussions of annotation and task details. We thank our colleagues at the Universities of Oslo and Potsdam for their comments and support.

[6] Like in the presentation of system errors, we include scope predictions of our own rules here too, which we believe to be correct in these cases. Also in this class of errors, we find the occasional 'uninteresting' mismatch, for example related to punctuation marks and inconsistencies around parentheses.

[7] In § 4 and § 6 above, we report scores for a slightly improved version of our system, where (after the official CoNLL submission date) we eliminated a bug related to the treatment of sentence-initial whitespace in the XML annotations. At an end-to-end F_1 of 55.75, this system would outrank the second best performer in Task 2.

[8] We are leaving open the possibility to further refine our system; we have therefore abstained from an error analysis on the evaluation data so far.

References

Butt, M., Dyvik, H., King, T. H., Masuichi, H., & Rohrer, C. (2002). The Parallel Grammar Project. In *Proceedings of COLING workshop on grammar engineering and evaluation* (pp. 1 – 7). Taipei, Taiwan.

Crouch, D., Dalrymple, M., Kaplan, R., King, T., Maxwell, J., & Newman, P. (2008). *XLE documentation.* Palo Alto, CA. (Palo Alto Research Center)

Farkas, R., Vincze, V., Mora, G., Csirik, J., & Szarvas, G. (2010). The CoNLL 2010 Shared Task: Learning to detect hedges and their scope in natural language text. In *Proceedings of the 14th Conference on Natural Language Learning.* Uppsala, Sweden.

Gildea, D. (2001). Corpus variation and parser performance. In *Proceedings of the 2001 conference on Empirical Methods in Natural Language Processing* (pp. 167 – 202). Pittsburgh, PA.

Johansson, R., & Nugues, P. (2007). Extended constituent-to-dependency conversion for English. In J. Nivre, H.-J. Kaalep, & M. Koit (Eds.), *Proceedings of NODALIDA 2007* (p. 105-112). Tartu, Estonia.

Kilicoglu, H., & Bergler, S. (2008). Recognizing speculative language in biomedical research articles: A linguistically motivated perspective. In *Proceedings of the BioNLP 2008 Workshop.* Columbus, OH, USA.

Kim, J.-D., Ohta, T., Pyysalo, S., Kano, Y., & Tsujii, J. (2009). Overview of BioNLP 2009 Shared Task on event extraction. In *Proceedings of the BioNLP 2009 workshop companion volume for shared task* (pp. 1 – 9). Boulder, CO: Association for Computational Linguistics.

Medlock, B., & Briscoe, T. (2007). Weakly supervised learning for hedge classification in scientific literature. In *Proceedings of the 45th Meeting of the Association for Computational Linguistics* (pp. 992 – 999). Prague, Czech Republic: Association for Computational Linguistics.

Morante, R., Asch, V. V., & Daelemans, W. (2010). Memory-based resolution of in-sentence scope of hedge cues. In *Proceedings of the 14th Conference on Natural Language Learning* (pp. 40 – 47). Uppsala, Sweden.

Morante, R., & Daelemans, W. (2009). Learning the scope of hedge cues in biomedical texts. In *Proceedings of the BioNLP 2009 Workshop* (pp. 28 – 36). Boulder, Colorado.

Ng, V., Dasgupta, S., & Arifin, S. M. N. (2006). Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics.* Sydney, Australia.

Nivre, J., Hall, J., & Nilsson, J. (2006). MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation* (p. 2216-2219). Genoa, Italy.

Nivre, J., & McDonald, R. (2008, June). Integrating graph-based and transition-based dependency parsers. In *Proceedings of the 46th Meeting of the Association for Computational Linguistics* (pp. 950 – 958). Columbus, Ohio.

Øvrelid, L., Kuhn, J., & Spreyer, K. (2009). Improving data-driven dependency parsing using large-scale LFG grammars. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics* (pp. 37 – 40). Singapore.

Øvrelid, L., Kuhn, J., & Spreyer, K. (2010). Cross-framework parser stacking for data-driven dependency parsing. *TAL 2010 special issue on Machine Learning for NLP*, *50*(3).

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, *2*(1-2).

Rei, M., & Briscoe, T. (2010). Combining manual rules and supervised learning for hedge cue and scope detection. In *Proceedings of the 14th Conference on Natural Language Learning* (pp. 56 – 63). Uppsala, Sweden.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *International conference on new methods in language processing* (p. 44-49). Manchester, England.

Tsuruoka, Y., Tateishi, Y., Kim, J.-D., Ohta, T., McNaught, J., Ananiadou, S., et al. (2005). Developing a robust Part-of-Speech tagger for biomedical text. In *Advances in informatics* (pp. 382 – 392). Berlin, Germany: Springer.

Velldal, E., Øvrelid, L., & Oepen, S. (2010). Resolving speculation: MaxEnt cue classification and dependency-based scope rules. In *Proceedings of the 14th Conference on Natural Language Learning.* Uppsala, Sweden.

Vincze, V., Szarvas, G., Farkas, R., Móra, G., & Csirik, J. (2008). The BioScope corpus: Annotation for negation, uncertainty and their scope in biomedical texts. In *Proceedings of the BioNLP 2008 Workshop.* Columbus, OH, USA.

Wilson, T., Wiebe, J., & Hwa, R. (2006). Recognizing strong and weak opinion clauses. *Computational Intelligence*, *22*(2), 73 – 99.

Zhang, Y., & Wang, R. (2009). Cross-domain dependency parsing using a deep linguistic grammar. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics.* Singapore.