2006

COLING • ACL

COLING • ACL 2006

21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics

Proceedings of the Conference

Volume 1

17-21 July 2006
Sydney, Australia

Production and Manufacturing by
BPA Digital
11 Evans St
Burwood VIC 3125
AUSTRALIA

Order copies of this and other ACL proceedings from:

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

ISBN 1-932432-65-5

Table of Contents

Preface: General Chair

I am honoured to write the first few words of these Proceedings, as General Chair of COLING/ACL 2006 in Sydney, Australia. As we know, this is just the third time in their history that the two traditionally major events in Computational Linguistics, COLING and ACL – organised respectively by ICCL (International Committee on Computational Linguistics) and ACL (the Association for Computational Linguistics) – are joined in one combined conference, after Stanford in 1984 and Montreal in 1998. I was lucky to attend both those wonderful events and would have never imagined to be "in charge" of the next one, the first of the new millennium!

When I accepted, I knew I didn't have real work to do in this position, apart from mediate – if necessary – among the several "real workers", the various Chairs. I must say now that my work was even easier than foreseen, because of the wonderful teamwork of all the COLING/ACL group.

In this joint Conference we have tried to maintain the spirit of both COLING and ACL, but the combination will inevitably have its own personality, in a mixture that is more than the simple sum of the two. Part of its character will be due to the location, for the first time – for both conferences – in Australia. For this reason we decided to have a member of AFNLP (the Asian Federation of Natural Language Processing) on the Advisory Board and to give particular attention and visibility to the Asia-Pacific context, communities and languages. We sincerely thank both the AFNLP-Nagao Fund for providing financial support for those presenting Asian NLP research, and ALTA (the Australasian Language Technology Association) for their local support.

It is my task here – but I should say my pleasure – to express gratitude to all those without whom this conference would not exist, and I think I can do that on behalf of all participants.

My biggest thanks go to all the Chairs, for their invaluable effort and dedication which made this Conference possible.

First of all the two Program Chairs: Claire Cardie and Pierre Isabelle, who did a tremendous job, managing so many submissions and taking care of both regular papers and posters, and the two Local Arrangements Chairs: Robert Dale and Cécile Paris, who have succeeded in keeping so many details under control, in such a smooth way as if everything were natural and effortless for them.

And all the others, for their precious, competent and hard work: the Workshops Chair: Suzanne Stevenson; the Student Workshop Chair: Rebecca Hwa; the Tutorials Chair: Claire Gardent; the Interactive Presentations Chair: James Curran; the Publications Chair: Olivia Kwong; the two Sponsorship Chairs: Steven Krauwer (International) and Dominique Estival (Australia); the Mentoring Chair: Richard Power, who kindly accepted to do this for the second time; the Publicity Chair: Tim Baldwin; the Exhibits Chair: Menno van Zaanen; the Student Volunteers coordinator: Priscilla Rasmussen, giving often advice to all of us as ACL business manager; the webmasters: Andrew Lampert and Brett Powley; and finally Judy Potter and her team from Well Done Events for managing registrations and assisting in the local organisation.

I warmly thank the Advisory Board – composed of four ICCL, four ACL, and one AFNLP members – to whom we resorted for suggestions on important and sometimes delicate issues: Sandra Carberry, Eva Hajicova, Aravind Joshi, Martin Kay, Kathleen McCoy, Martha Palmer, Priscilla Rasmussen, Benjamin T'sou, Jun'ichi Tsujii.

I express my gratitude to all the sponsors for their great support to the conference.

I thank all the organizers of the so numerous surrounding workshops, tutorials, and other co-located events – conferences, workshops, summer school – adding value to the main conference, creating altogether probably the biggest ever happening in Computational Linguistics.

My thanks to the area chairs, the reviewers, the invited speakers, the authors of the various presentations, in particular the students who enter with enthusiasm in such an exciting field, all the participants who will often make a long trip to be present at COLING/ACL 2006, and all those who contributed in many ways to a success of the conference.

And I finally thank both ICCL and ACL for having decided to join forces again in such a great enterprise. COLING/ACL 2006 will be, I'm sure, an exciting, stimulating and inspiring event for all of you.

Enjoy COLING/ACL 2006! . . . and consider that some of the youngest here do not know it yet, but they will be chairing the next joint events in a few years.

Nicoletta Calzolari
COLING/ACL 2006 General Chair
June 2006

Preface: Program Committee Co-Chairs

This conference represents just the third time in their 40+ year history that the two premier conferences in natural language processing, computational linguistics, and language technology have merged for a joint COLING/ACL event; and it's the first time that the joint conference will be held in the southern hemisphere. It is fitting then, that we received a record number of 630 submissions from 40+ countries: 39% from 13 countries in Asia, 29% from 17 countries in Europe, 25% from Canada and the United States, 4% from Australia and New Zealand, 2% from 4 countries in the Middle East, and less than 1% from South America (Brazil) and Africa (South Africa and Tunisia). Of the 630 submissions, 23% were accepted for paper presentations and an additional 20% for poster presentations.

Our rough estimate of the amount of work that went just into preparing the submissions, final versions and on-site presentations for this year's main program exceeds 32 person/years.[1] If we include workshops and EMNLP, this figure is probably doubled. Thanks to everyone who submitted their research to the conference!

Much of the work in putting together the main program of papers and posters was done, of course, by our tireless area chairs and reviewers (of which we had 19 and 384, respectively). A tribute to their joint efforts is the fact that we obtained a 100% response rate for reviews – over 100% actually, since a few crazy souls offered unsolicited or extra reviews.

COLING/ACL 2006 spans five days with the traditional COLING "excursion day" on day three. The remaining four days of the conference include plenary sessions, four parallel paper sessions, the student research workshop, and two evening poster sessions. The ACL Lifetime Achievement Award will also be bestowed on its fifth recipient in a plenary session, followed by an invited talk by the esteemed award winner. A Best Paper Award will be announced in a plenary session at the end of the conference. We would like to especially thank our two invited speakers, Daniel Marcu and Sally McConnell-Ginet.

In honor of the joint conference's location, we have planned a special Asian language event for Thursday morning that consists of paper presentations of the top four Asian language papers followed by a plenary panel focusing on issues in Asian language processing, and ending with the presentation of the Best Asian Language Paper Award. We offer special thanks to our three distinguished panelists – Pushpak Bhattacharyya, Benjamin T'sou, and Jun'ichi Tsujii – and to Aravind Joshi, who expertly organized the panel.

Finally, we thank the ACL and ICCL conference oversight committee, for advice of all sorts along the way; and Rich Gerber, the START conference system developer, who answered our countless questions at all hours of the day and night.

After all of this work, by so many people, we are very much looking forward to sitting back and enjoying the conference with you in Sydney in July!

Claire Cardie
Pierre Isabelle
June 2006

[1] We assume an average of 8 days of work to prepare each one of about 630 submissions to the COLING-ACL 2006 main program; and an average of 5 days of work to produce final versions for each one of the 267 accepted contributions.

Organizers

General Chair:

Nicoletta Calzolari, Istituto di Linguistica Computazionale – CNR, Italy

Program Committee Co-Chairs:

Claire Cardie, Cornell University, USA
Pierre Isabelle, National Research Council of Canada, Canada

Tutorials Chair:

Claire Gardent, CNRS/LORIA, France

Workshops Chair:

Suzanne Stevenson, University of Toronto, Canada

Workshops Program Committee:

Ann Copestake, University of Cambridge, UK
Pascale Fung, Hong Kong University of Science and Technology, Hong Kong
Jamie Henderson, University of Edinburgh, UK
Ingrid Zukerman, Monash University, Australia

Interactive Presentations Chair:

James Curran, University of Sydney, Australia

Publications Chair:

Olivia Kwong, City University of Hong Kong, Hong Kong

Sponsorship Chairs:

Steven Krauwer, ELSNET / UiL OTS, The Netherlands
Dominique Estival, Appen Pty Limited, Australia

Exhibits Chair:

Menno van Zaanen, Macquarie University, Australia

Mentoring Service Chair:

Richard Power, The Open University, UK

Publicity Chair:

Timothy Baldwin, University of Melbourne, Australia

Student Research Workshop:

Rebecca Hwa, University of Pittsburgh, USA
Marine Carpuat, Hong Kong University of Science and Technology, Hong Kong
Kevin Duh, University of Washington, USA

Local Organization Chairs:

Robert Dale, Macquarie University, Australia
Cécile Paris, CSIRO ICT Centre, Australia

Local Organizing Advisory Committee:

John Debenham, University of Technology, Sydney, Australia
Jon Patrick, University of Sydney, Australia
Raymond Wong, University of New South Wales, Australia

Student Volunteers Coordinator:

Priscilla Rasmussen, Association for Computational Linguistics, USA

Conference Webmasters:

Andrew Lampert, CSIRO ICT Centre, Australia
Brett Powley, Macquarie University, Australia

Conference Secretariat Coordinator:

Judy Potter, Well Done Events, Australia

Graphic Design:

Kathie Mason, Macquarie University, Australia

Advisory Committee:

Sandra Carberry, University of Delaware, USA
Eva Hajicova, Charles University, Czech Republic
Aravind K. Joshi, University of Pennsylvania, USA
Martin Kay, Stanford University, USA
Kathleen McCoy, University of Delaware, USA
Martha Palmer, University of Colorado, USA
Priscilla Rasmussen, Association for Computational Linguistics, USA
Benjamin Tsou, City University of Hong Kong, Hong Kong
Jun'ichi Tsujii, University of Tokyo, Japan

International Committee on Computational Linguistics (ICCL):

Igor Boguslavsky, Russian Academy of Sciences, Russia
Christian Boitet, GETA, CLIPS, IMAG, France
Nicoletta Calzolari, Istituto di Linguistica Computazionale – CNR, Italy
Eva Hajicova, Charles University, Czech Republic
Kolbjorn Heggstad
Chu-Ren Huang, Academia Sinica, Taiwan
Pierre Isabelle, National Research Council of Canada, Canada

Aravind K. Joshi, University of Pennsylvania, USA
Martin Kay, Stanford University, USA
Winfried Lenders, IKS-Universitaet Bonn, Germany
Makoto Nagao, Kyoto University, Japan
Sergei Nirenburg, University of Maryland Baltimore County, USA
Helmut Schnelle
Donia Scott, The Open University, UK
Petr Sgall, Charles University, Czech Republic
Hozumi Tanaka, Tokyo Institute of Technology, Japan
Jun'ichi Tsujii, University of Tokyo, Japan
Hans Uszkoreit, DFKI Saarbruecken, Germany
Hiroshi Wada
Yorick Wilks, University of Sheffield, UK

ACL Executive Committee:

Jun'ichi Tsujii, University of Tokyo, Japan
Mark Steedman, University of Edinburgh, UK
Bonnie Dorr, University of Maryland, USA
Kathleen McCoy, University of Delaware, USA
Dragomir Radev, University of Michigan, USA
Martha Palmer, University of Colorado, USA
Sandra Carberry, University of Delaware, USA
Walter Daelemans, University of Antwerp, Belgium
Keh-Yih Su, Behavior Design Corporation, Taiwan
Claire Cardie, Cornell University, USA

Program Committee

Chairs:

Claire Cardie, Cornell University, USA
Pierre Isabelle, National Research Council of Canada, Canada

Area Chairs:

Johan Bos, Università di Roma “La Sapienza”, Italy
Jason Chang, National Tsing Hua University, Taiwan
David Chiang, USC Information Sciences Institute, USA
Eva Hajicova, Charles University, Czech Republic
Chu-Ren Huang, Academia Sinica, Taiwan
Martin Kay, Stanford University, USA
Emiel Krahmer, Tilburg University, The Netherlands
Roland Kuhn, National Research Council of Canada, Canada
Lillian Lee, Cornell University, USA
Yuji Matsumoto, Nara Institute of Technology, Japan
Dan Moldovan, University of Texas, USA
Mark-Jan Nederhof, University of Groningen, The Netherlands
Hwee Tou Ng, National University of Singapore, Singapore
John Prager, IBM Watson Research Center, USA
Anoop Sarkar, Simon Fraser University, Canada
Donia Scott, The Open University, UK
Simone Teufel, University of Cambridge, UK
Benjamin Tsou, City University of Hong Kong, Hong Kong
ChengXiang Zhai, University of Illinois, USA
Ming Zhou, Microsoft Beijing, China

Program Committee Members:

Anne Abeille, Eugene Agichtein, Eneko Agirre, David Ahn, Lars Ahrenberg, Miguel Alonso Pardo, Rie Ando, Elisabeth Andre, Galen Andrew, Shlomo Argamon, Masayuki Asahara, Tania Avgustinova, Necip Fazil Ayan

Srinivas Bangalore, Regina Barzilay, Roberto Basili, Tilman Becker, S M Bendre, Stefano Bertolo, Pushpak Bhattacharya, Steffen Bickel, Daniel Bikel, Mikhail Bilenko, Philippe Blache, William Black, Constantinos Boulis, Thorsten Brants, Eric Breck, Sabine Buchholz, Razvan Bunescu, John Burger, Donna Byron

Chris Callison-Burch, Jaime Carbonell, Xavier Carreras, Vitor Carvalho, Nuria Castell, Frantisek Cermak, Joyce Chai, Soumen Chakrabarti, Ciprian Chelba, Hsin-Hsi Chen, John Chen, Keh-Jiann Chen, Kuang-hua Chen, Lee-Feng Chien, Yejin Choi, Tat-Seng Chua, Ken Church, Stephen Clark, James Clarke, Michael Collins, Matteo Contolini, Koby Crammer, Mathias Creutz, Silviu Cucerzan, James Curran, Krzysztof Czuba

Walter Daelemans, Ido Dagan, Hal Daume III, Renato DeMori, Barbara Di Eugenio, Mona Diab, Christy Doran, Mark Dras, Amit Dubey, Kevin Duh

Noemie Elhadad, Katrin Erk, Andrea Esuli, Yair Even-Zohar

Marcello Federico, Christiane Fellbaum, Radu Florian, George Forman, George Foster, Anette Frank, Robert Frank, Alex Fraser, Dayne Freitag, Sadaoki Furui

Evgeniy Gabrilovich, Jianfeng Gao, Eric Gaussier, Mark Gawron, Ruifang Ge, Effi Georgala, Mazin Gilbert, Roxana Girju, Oren Glickman, Jade Goldstein, Yifan Gong, Silke Goronzy, Cyril Goutte, Gregory Grefenstette, Ralph Grishman

Nizar Habash, Udo Hahn, Thomas Hain, Jan Hajic, Keith Hall, Sanda Harabagiu, Mary Harper, James Henderson, John Henderson, Erhard Hinrichs, Graeme Hirst, Barbora Hladka, Julia Hockenmaier, Veronique Hoste, Shu-Kai Hsieh, Fei Huang, Liang Huang

Nancy Ide, Kentaro Inui, Hitoshi Isahara, Hideki Isozaki, Abe Ittycheriah

Paul Jacobs, Nathalie Japkowicz, Donghong Ji, Rong Jin, Hongyan Jing, Howard Johnson, Michael Johnston, Rosie Jones, Jean-Claude Junqua

Kyo Kageura, Laura Kallmeyer, Nanda Kambhatla, Noriko Kando, Boris Katz, Asanee Kawtrakul, Martin Kay, Frank Keller, Rodger Kibble, Adam Kilgarriff, Tracy Hooloway King, Alexandra Kinyon, Katrin Kirchhoff, Dan Klein, Kevin Knight, Alistair Knott, Philipp Koehn, Moshe Koppel, Kimmo Koskenniemi, Taku Kudo, Peter Kuehnlein, Jonas Kuhn, Roland Kuhn, Shankar Kumar, Oren Kurland, Sadao Kurohashi, K L Kwok, Olivia Kwong

Tom Lai, Irene Langkilde-Geary, Philippe Langlais, Mirella Lapata, Alex Lascarides, Alberto Lavelli, Alon Lavie, Victor Lavrenko, Guy Lebanon, Gary (Geunbae) Lee, Jong-Hyeok Lee, Young-Suk Lee, Oliver Lemon, Alessandro Lenci, Piroska Lendvai, David Lewis, Hang Li, Mu Li, Xin Li, Liz Liddy, Chin-Yew Lin, Jimmy Lin, Ken Litkowski, Bing Liu, Beth Logan, Marketa Lopatkova, Adam Lopez, Yajuan Lu, Xiaoqiang Luo

Qing Ma, Bente Maegaard, Milind Mahajan, Steve Maiorano, Rob Malouf, Daniel Marcu, Mitch Marcus, Katja Markert, Lluis Marquez, Erwin Marsi, Yuji Matsumoto, John Maxwell, Andrew McCallum, Diana McCarthy, Kathleen McCoy, Ryan McDonald, Dan Melamed, Helen Meng, Wolfgang Menzel, Paola Merlo, Detmar Meurers, Adam Meyers, Rada Mihalcea, Natasa Milic-Frayling, Eleni Miltsakaki, Ruslan Mitkov, Mandar Mitra, Vibhu Mittal, Yusuke Miyao, Dunja Mladenic, Dan Moldovan, Monica Monachini, Christof Monz, Johanna Moore, Alessandro Moschitti, Isabelle Moulinier, Dragos Munteanu, Masaki Murata

Masaaki Nagata, Sobha L Nair, Hiroshi Nakagawa, Satoshi Nakamura, Srini Narayanan, Wuritu Nashun, Roberto Navigli, Hermann Ney, Vincent Ng, Patrick Nguyen, Nicolas Nicolov, Jian-Yun Nie, Kamal Nigam, Takashi Ninomiya, Malvina Nissim, Cheng Niu, Zheng-Yu Niu, Joakim Nivre, Eric Nyberg

Jon Oberlander, Franz Och, Stephan Oepen, Kemal Oflazer, Miles Osborne

Petr Pajas, Karel Pala, Martha Palmer, Jarmila Panevova, Bo Pang, Patrick Pantel, Fuchun Peng, Gerald Penn, Vladimqr Petkevic, Fabio Pianesi, Paul Piwek, Massimo Poesio, Patrice Pognan, Ana-Maria Popescu, Andrei Popescu-Belis, Richard Power, Sameer Pradhan, John Prange, Rashmi Prasad, Detlef Prescher, Laurent Prevot, Katharina Probst, Gabor Proszeky, Josef Psutka

Yan Qu

Owen Rambow, Alan Ramsay, Philip Resnik, Stefan Riezler, German Rigau, Hae-Chang Rim, Brian Roark, Rick Rose, Alex Rudnicky, Pavel Rychly

Carl Sable, Louisa Sadler, Kenji Sagae, Antonio Sanfilippo, Rajeev Sangal, Sunita Sarawagi, Giorgio Satta, Michael Schiehlen, Frank Schilder, Helmut Schmid, Hinrich Schuetze, Tanja Schultz, Holger Schwenk, Donia Scott, Jiri Semecky, Petr Sgall, Fei Sha, Vijay Shanker, Dipti M Sharma, Libin Shen, Xiaodong Shi, Atsushi Shimojima, Luo Si, Advaith Siddharthan, Khalil Simaan, Man-Hung Siu, David Smith, Noah Smith, Harold Somers, Virach Sornlertlamvanich, Karen Sparck-Jones, Rohini Srihari, Manfred Stede, Mark Steedman, Amanda Stent, Suzanne Stevenson, Matthew Stone, Veselin Stoyanov, Carlo Strapparava, Tomek Strzalkowski, Jian Su, Keh-Yih Su, Maosong Sun, Mihai Surdeanu, Jun Suzuki, Marc Swerts

Hiroya Takamura, Marta Tatu, Thanaruk Theeramunkong, Mariet Theune, Christoph Tillmann, Takenobu Tokunaga, Andrew Tomkins, Kristina Toutanova, David Traum, Huihsin Tseng, Benjamin Tsou, Dan Tufis, Peter Turney

Kiyotaka Uchimoto, Nicola Ueffing, Takehito Utsuro

Antal van den Bosch, Josef van Genabith, Gertjan van Noord, Lucy Vanderwende, Hans vanHalteren, Ashish Venugopal, Peter Veprek, Stephan Vogel, Piek Vossen

Marilyn Walker, Shaojun Wang, Wei Wang, Xiaolong Wang, Ye-Yi Wang, Andy Way, Bonnie Webber, David Weir, Michael White, Ed Whittaker, Richard Wicentowski, Jan Wiebe, Yorick Wilks, Theresa Wilson, Shuly Wintner, Dekai Wu, Xiaoyuan Wu, Yunfang Wu

Fei Xia, Peng Xu

Z Zabokrtsky, Fabio Massimo Zanzotto, Daniel Zeman, Richard Zens, Hao Zhang, Tong Zhang, Yi Zhang, Ying Zhang, GuoDong Zhou, Jingbo Zhu, Imed Zitouni

Conference Program

Monday, 17 July 2006

09:00–09:30 Opening Session

Session 1A: Machine Translation I

09:30–10:00 *Combination of Arabic Preprocessing Schemes for Statistical Machine Translation*
Fatiha Sadat and Nizar Habash

10:00–10:30 *Going Beyond AER: An Extensive Analysis of Word Alignments and Their Impact on MT*
Necip Fazil Ayan and Bonnie J. Dorr

Session 1B: Topic Segmentation

09:30–10:00 *Unsupervised Topic Modelling for Multi-Party Spoken Discourse*
Matthew Purver, Konrad P. Körding, Thomas L. Griffiths and Joshua B. Tenenbaum

10:00–10:30 *Minimum Cut Model for Spoken Lecture Segmentation*
Igor Malioutov and Regina Barzilay

Session 1C: Coreference

09:30–10:00 *Bootstrapping Path-Based Pronoun Resolution*
Shane Bergsma and Dekang Lin

10:00–10:30 *Kernel-Based Pronoun Resolution with Structured Syntactic Knowledge*
Xiaofeng Yang, Jian Su and Chew Lim Tan

Session 1D: Grammars I

09:30–10:00 *A Finite-State Model of Human Sentence Processing*
Jihyun Park and Chris Brew

10:00–10:30 *Acceptability Prediction by Means of Grammaticality Quantification*
Philippe Blache, Barbara Hemforth and Stéphane Rauzy

10:30–11:00 Break

Monday, 17 July 2006 (continued)

Session 2A: Machine Translation II

11:00–11:30 *Discriminative Word Alignment with Conditional Random Fields*
Phil Blunsom and Trevor Cohn

11:30–12:00 *Named Entity Transliteration with Comparable Corpora*
Richard Sproat, Tao Tao and ChengXiang Zhai

12:00–12:30 *Extracting Parallel Sub-Sentential Fragments from Non-Parallel Corpora*
Dragos Stefan Munteanu and Daniel Marcu

Session 2B: Word Sense Disambiguation I

11:00–11:30 *Estimating Class Priors in Domain Adaptation for Word Sense Disambiguation*
Yee Seng Chan and Hwee Tou Ng

11:30–12:00 *Ensemble Methods for Unsupervised WSD*
Samuel Brody, Roberto Navigli and Mirella Lapata

12:00–12:30 *Meaningful Clustering of Senses Helps Boost Word Sense Disambiguation Performance*
Roberto Navigli

Session 2C: Information Extraction I

11:00–11:30 *Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations*
Patrick Pantel and Marco Pennacchiotti

11:30–12:00 *Modeling Commonality among Related Classes in Relation Extraction*
GuoDong Zhou, Jian Su and Min Zhang

12:00–12:30 *Relation Extraction Using Label Propagation Based Semi-Supervised Learning*
Jinxiu Chen, Donghong Ji, Chew Lim Tan and Zhengyu Niu

Session 2D: Grammars II

11:00–11:30 *Polarized Unification Grammars*
Sylvain Kahane

11:30–12:00 *Partially Specified Signatures: A Vehicle for Grammar Modularity*
Yael Cohen-Sygal and Shuly Wintner

12:00–12:30 *Morphology-Syntax Interface for Turkish LFG*
Özlem Çetinoğlu and Kemal Oflazer

12:30–14:00 Lunch

Monday, 17 July 2006 (continued)

Session 3A: Parsing I

14:00–14:30 *PCFGs with Syntactic and Prosodic Indicators of Speech Repairs*
John Hale, Izhak Shafran, Lisa Yung, Bonnie Dorr, Mary Harper, Anna Krasnyanskaya, Matthew Lease, Yang Liu, Brian Roark, Matthew Snover and Robin Stewart

14:30–15:00 *Dependency Parsing of Japanese Spoken Monologue Based on Clause Boundaries*
Tomohiro Ohno, Shigeki Matsubara, Hideki Kashioka, Takehiko Maruyama and Yasuyoshi Inagaki

15:00–15:30 *Trace Prediction and Recovery with Unlexicalized PCFGs and Slash Features*
Helmut Schmid

Session 3B: Dialogue I

14:00–14:30 *Learning More Effective Dialogue Strategies Using Limited Dialogue Move Features*
Matthew Frampton and Oliver Lemon

14:30–15:00 *Dependencies between Student State and Speech Recognition Problems in Spoken Tutoring Dialogues*
Mihai Rotaru and Diane J. Litman

15:00–15:30 *Learning the Structure of Task-Driven Human-Human Dialogs*
Srinivas Bangalore, Giuseppe Di Fabbrizio and Amanda Stent

Session 3C: Machine Learning Methods I

14:00–14:30 *Semi-Supervised Conditional Random Fields for Improved Sequence Segmentation and Labeling*
Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner and Dale Schuurmans

14:30–15:00 *Training Conditional Random Fields with Multivariate Evaluation Measures*
Jun Suzuki, Erik McDermott and Hideki Isozaki

15:00–15:30 *Approximation Lasso Methods for Language Modeling*
Jianfeng Gao, Hisami Suzuki and Bin Yu

Session 3D: Applications I

14:00–14:30 *Automated Japanese Essay Scoring System based on Articles Written by Experts*
Tsunenori Ishioka and Masayuki Kameda

14:30–15:00 *A Feedback-Augmented Method for Detecting Errors in the Writing of Learners of English*
Ryo Nagata, Atsuo Kawai, Koichiro Morihiro and Naoki Isu

15:00–15:30 *Correcting ESL Errors Using Phrasal SMT Techniques*
Chris Brockett, William B. Dolan and Michael Gamon

15:30–16:00 Break

Monday, 17 July 2006 (continued)

Session 4A: Parsing II

16:00–16:30 *Graph Transformations in Data-Driven Dependency Parsing*
Jens Nilsson, Joakim Nivre and Johan Hall

Session 4B: Dialogue II

16:00–16:30 *Learning to Generate Naturalistic Utterances Using Reviews in Spoken Dialogue Systems*
Ryuichiro Higashinaka, Rashmi Prasad and Marilyn A. Walker

Session 4C: Linguistic Kinships

16:00–16:30 *Measuring Language Divergence by Intra-Lexical Comparison*
T. Mark Ellison and Simon Kirby

Session 4D: Applications II

16:00–16:30 *Enhancing Electronic Dictionaries with an Index Based on Associations*
Olivier Ferret and Michael Zock

16:30–17:30 ACL Lifetime Achievement Award

17:30–19:30 Poster Sessions

Tuesday, 18 July 2006

09:00–10:00 Invited Talk by Daniel Marcu: $Argmax$ *Search in Natural Language Processing*

Session 5A: Parsing III

10:00–10:30 *Guiding a Constraint Dependency Parser with Supertags*
Kilian Foth, Tomas By and Wolfgang Menzel

Session 5B: Lexical Issues I

10:00–10:30 *Efficient Unsupervised Discovery of Word Categories Using Symmetric Patterns and High Frequency Words*
Dmitry Davidov and Ari Rappoport

Session 5C: Summarization I

10:00–10:30 *Bayesian Query-Focused Summarization*
Hal Daumé III and Daniel Marcu

Session 5D: Semantics I

10:00–10:30 *Expressing Implicit Semantic Relations without Supervision*
Peter D. Turney

10:30–11:00 Break

Tuesday, 18 July 2006 (continued)

Session 6A: Parsing IV

11:00–11:30 *Hybrid Parsing: Using Probabilistic Models as Predictors for a Symbolic Parser*
Kilian A. Foth and Wolfgang Menzel

11:30–12:00 *Error Mining in Parsing Results*
Benoît Sagot and Éric de La Clergerie

12:00–12:30 *Reranking and Self-Training for Parser Adaptation*
David McClosky, Eugene Charniak and Mark Johnson

Session 6B: Lexical Issues II

11:00–11:30 *Automatic Classification of Verbs in Biomedical Texts*
Anna Korhonen, Yuval Krymolowski and Nigel Collier

11:30–12:00 *Selection of Effective Contextual Information for Automatic Synonym Acquisition*
Masato Hagiwara, Yasuhiro Ogawa and Katsuhiko Toyama

12:00–12:30 *Scaling Distributional Similarity to Large Corpora*
James Gorman and James R. Curran

Session 6C: Summarization II

11:00–11:30 *Extractive Summarization using Inter- and Intra- Event Relevance*
Wenjie Li, Mingli Wu, Qin Lu, Wei Xu and Chunfa Yuan

11:30–12:00 *Models for Sentence Compression: A Comparison across Domains, Training Requirements and Evaluation Measures*
James Clarke and Mirella Lapata

12:00–12:30 *A Bottom-Up Approach to Sentence Ordering for Multi-Document Summarization*
Danushka Bollegala, Naoaki Okazaki and Mitsuru Ishizuka

Session 6D: Semantics II

11:00–11:30 *Learning Event Durations from Event Descriptions*
Feng Pan, Rutu Mulkar and Jerry R. Hobbs

11:30–12:00 *Automatic Learning of Textual Entailments with Cross-Pair Similarities*
Fabio Massimo Zanzotto and Alessandro Moschitti

12:00–12:30 *An Improved Redundancy Elimination Algorithm for Underspecified Representations*
Alexander Koller and Stefan Thater

12:30–14:00 Lunch

Tuesday, 18 July 2006 (continued)

Session 7A: Parsing V

14:00–14:30 *Integrating Syntactic Priming into an Incremental Probabilistic Parser, with an Application to Psycholinguistic Modeling*
Amit Dubey, Frank Keller and Patrick Sturt

14:30–15:00 *A Fast, Accurate Deterministic Parser for Chinese*
Mengqiu Wang, Kenji Sagae and Teruko Mitamura

15:00–15:30 *Learning Accurate, Compact, and Interpretable Tree Annotation*
Slav Petrov, Leon Barrett, Romain Thibaux and Dan Klein

Session 7B: Word Sense Disambiguation II

14:00–14:30 *Semi-Supervised Learning of Partial Cognates Using Bilingual Bootstrapping*
Oana Frunza and Diana Inkpen

14:30–15:00 *Direct Word Sense Matching for Lexical Substitution*
Ido Dagan, Oren Glickman, Alfio Gliozzo, Efrat Marmorshtein and Carlo Strapparava

15:00–15:30 *An Equivalent Pseudoword Solution to Chinese Word Sense Disambiguation*
Zhimao Lu, Haifeng Wang, Jianmin Yao, Ting Liu and Sheng Li

Session 7C: Information Extraction II

14:00–14:30 *Improving the Scalability of Semi-Markov Conditional Random Fields for Named Entity Recognition*
Daisuke Okanohara, Yusuke Miyao, Yoshimasa Tsuruoka and Jun’ichi Tsujii

14:30–15:00 *Factorizing Complex Models: A Case Study in Mention Detection*
Radu Florian, Hongyan Jing, Nanda Kambhatla and Imed Zitouni

15:00–15:30 *Segment-Based Hidden Markov Models for Information Extraction*
Zhenmei Gu and Nick Cercone

Session 7D: Resources I

14:00–14:30 *A DOM Tree Alignment Model for Mining Parallel Data from the Web*
Lei Shi, Cheng Niu, Ming Zhou and Jianfeng Gao

14:30–15:00 *QuestionBank: Creating a Corpus of Parse-Annotated Questions*
John Judge, Aoife Cahill and Josef van Genabith

15:00–15:30 *Creating a CCGbank and a Wide-Coverage CCG Lexicon for German*
Julia Hockenmaier

15:30–16:00 Break

Tuesday, 18 July 2006 (continued)

Session 8A: Machine Translation III

16:00–16:30 *Improved Discriminative Bilingual Word Alignment*
Robert C. Moore, Wen-tau Yih and Andreas Bode

16:30–17:00 *Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation*
Deyi Xiong, Qun Liu and Shouxun Lin

17:00–17:30 *Distortion Models for Statistical Machine Translation*
Yaser Al-Onaizan and Kishore Papineni

Session 8B: Text Classification I

16:00–16:30 *A Study on Automatically Extracted Keywords in Text Categorization*
Anette Hulth and Beáta B. Megyesi

16:30–17:00 *A Comparison and Semi-Quantitative Analysis of Words and Character-Bigrams as Features in Chinese Text Categorization*
Jingyang Li, Maosong Sun and Xian Zhang

17:00–17:30 *Exploiting Comparable Corpora and Bilingual Dictionaries for Cross-Language Text Categorization*
Alfio Gliozzo and Carlo Strapparava

Session 8C: Machine Learning Methods II

16:00–16:30 *A Progressive Feature Selection Algorithm for Ultra Large Feature Spaces*
Qi Zhang, Fuliang Weng and Zhe Feng

16:30–17:00 *Annealing Structural Bias in Multilingual Weighted Grammar Induction*
Noah A. Smith and Jason Eisner

17:00–17:30 *Maximum Entropy Based Restoration of Arabic Diacritics*
Imed Zitouni, Jeffrey S. Sorensen and Ruhi Sarikaya

Session 8D: Information Retrieval I

16:00–16:30 *An Iterative Implicit Feedback Approach to Personalized Search*
Yuanhua Lv, Le Sun, Junlin Zhang, Jian-Yun Nie, Wan Chen and Wei Zhang

16:30–17:00 *The Effect of Translation Quality in MT-Based Cross-Language Information Retrieval*
Jiang Zhu and Haifeng Wang

17:00–17:30 *A Comparison of Document, Sentence, and Term Event Spaces*
Catherine Blake

17:30–19:30 Poster Sessions

Thursday, 20 July 2006

Session 9A: Best Asian Language Paper Nominee

09:00–09:30 *Tree-to-String Alignment Template for Statistical Machine Translation*
Yang Liu, Qun Liu and Shouxun Lin

Session 9B: Best Asian Language Paper Nominee

09:00–09:30 *Incorporating Speech Recognition Confidence into Discriminative Named Entity Recognition of Speech Data*
Katsuhito Sudoh, Hajime Tsukada and Hideki Isozaki

Session 9C: Best Asian Language Paper Nominee

09:00–09:30 *Exploiting Syntactic Patterns as Clues in Zero-Anaphora Resolution*
Ryu Iida, Kentaro Inui and Yuji Matsumoto

Session 9D: Best Asian Language Paper Nominee

09:00–09:30 *Self-Organizing n-gram Model for Automatic Word Spacing*
Seong-Bae Park, Yoon-Shik Tae and Se-Young Park

09:30–10:30 Asian Language Special Event: *Challenges in NLP: Some New Perspectives from the East*

10:30–11:00 Break

Thursday, 20 July 2006 (continued)

Session 10A: Asian Language Processing

11:00–11:30 *Concept Unification of Terms in Different Languages for IR*
Qing Li, Sung-Hyon Myaeng, Yun Jin and Bo-yeong Kang

11:30–12:00 *Word Alignment in English-Hindi Parallel Corpus Using Recency-Vector Approach: Some Studies*
Niladri Chatterjee and Saumya Agrawal

12:00–12:30 *Extracting Loanwords from Mongolian Corpora and Producing a Japanese-Mongolian Bilingual Dictionary*
Badam-Osor Khaltar, Atsushi Fujii and Tetsuya Ishikawa

Session 10B: Morphology and Word Segmentation

11:00–11:30 *An Unsupervised Morpheme-Based HMM for Hebrew Morphological Disambiguation*
Meni Adler and Michael Elhadad

11:30–12:00 *Contextual Dependencies in Unsupervised Word Segmentation*
Sharon Goldwater, Thomas L. Griffiths and Mark Johnson

12:00–12:30 *MAGEAD: A Morphological Analyzer and Generator for the Arabic Dialects*
Nizar Habash and Owen Rambow

Session 10C: Tagging and Chunking

11:00–11:30 *Noun Phrase Chunking in Hebrew: Influence of Lexical and Morphological Features*
Yoav Goldberg, Meni Adler and Michael Elhadad

11:30–12:00 *Multi-Tagging for Lexicalized-Grammar Parsing*
James R. Curran, Stephen Clark and David Vadas

12:00–12:30 *Guessing Parts-of-Speech of Unknown Words Using Global Information*
Tetsuji Nakagawa and Yuji Matsumoto

12:30–13:30 Lunch

13:30–14:30 ACL Business Meeting

Thursday, 20 July 2006 (continued)

Session 11A: Machine Translation IV

14:30–15:00 *A Clustered Global Phrase Reordering Model for Statistical Machine Translation*
Masaaki Nagata, Kuniko Saito, Kazuhide Yamamoto and Kazuteru Ohashi

15:00–15:30 *A Discriminative Global Training Algorithm for Statistical MT*
Christoph Tillmann and Tong Zhang

Session 11B: Speech

14:30–15:00 *Phoneme-to-Text Transcription System with an Infinite Vocabulary*
Shinsuke Mori, Daisuke Takuma and Gakuto Kurata

15:00–15:30 *Automatic Generation of Domain Models for Call-Centers from Noisy Transcriptions*
Shourya Roy and L Venkata Subramaniam

Session 11C: Discourse

14:30–15:00 *Proximity in Context: An Empirically Grounded Computational Model of Proximity for Processing Topological Spatial Expressions*
John D. Kelleher, Geert-Jan M. Kruijff and Fintan J. Costello

15:00–15:30 *Machine Learning of Temporal Relations*
Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee and James Pustejovsky

15:30–16:00 Break

Thursday, 20 July 2006 (continued)

Session 12A: Machine Translation V

16:00–16:30 *An End-to-End Discriminative Approach to Machine Translation*
Percy Liang, Alexandre Bouchard-Côté, Dan Klein and Ben Taskar

16:30–17:00 *Semi-Supervised Training for Statistical Word Alignment*
Alexander Fraser and Daniel Marcu

17:00–17:30 *Left-to-Right Target Generation for Hierarchical Phrase-Based Translation*
Taro Watanabe, Hajime Tsukada and Hideki Isozaki

Session 12B: Lexical Issues III

16:00–16:30 *You Can't Beat Frequency (Unless You Use Linguistic Knowledge) – A Qualitative Evaluation of Association Measures for Collocation and Term Extraction*
Joachim Wermter and Udo Hahn

16:30–17:00 *Ontologizing Semantic Relations*
Marco Pennacchiotti and Patrick Pantel

17:00–17:30 *Semantic Taxonomy Induction from Heterogenous Evidence*
Rion Snow, Daniel Jurafsky and Andrew Y. Ng

Session 12C: Information Extraction III

16:00–16:30 *Names and Similarities on the Web: Fact Extraction in the Fast Lane*
Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits and Alpa Jain

16:30–17:00 *Weakly Supervised Named Entity Transliteration and Discovery from Multilingual Comparable Corpora*
Alexandre Klementiev and Dan Roth

17:00–17:30 *A Composite Kernel to Extract Relations between Entities with Both Flat and Structured Features*
Min Zhang, Jie Zhang, Jian Su and GuoDong Zhou

Friday, 21 July 2006

9:00–10:00 Invited Talk by Sally McConnell-Ginet: *Language, Gender and Sexuality: Do Bodies Always Matter?*

Session 13A: Parsing VI

10:00–10:30 *Japanese Dependency Parsing Using Co-Occurrence Information and a Combination of Case Elements*
Takeshi Abekawa and Manabu Okumura

Session 13B: Question Answering I

10:00–10:30 *Answer Extraction, Semantic Clustering, and Extractive Summarization for Clinical Question Answering*
Dina Demner-Fushman and Jimmy Lin

Session 13C: Semantics III

10:00–10:30 *Discovering Asymmetric Entailment Relations between Verbs Using Selectional Preferences*
Fabio Massimo Zanzotto, Marco Pennacchiotti and Maria Teresa Pazienza

Session 13D: Applications III

10:00–10:30 *Event Extraction in a Plot Advice Agent*
Harry Halpin and Johanna D. Moore

10:30–11:00 Break

Friday, 21 July 2006 (continued)

Session 14A: Parsing VII

11:00–11:30 *An All-Subtrees Approach to Unsupervised Parsing*
Rens Bod

11:30–12:00 *Advances in Discriminative Parsing*
Joseph Turian and I. Dan Melamed

12:00–12:30 *Prototype-Driven Grammar Induction*
Aria Haghighi and Dan Klein

Session 14B: Question Answering II

11:00–11:30 *Exploring Correlation of Dependency Relation Paths for Answer Extraction*
Dan Shen and Dietrich Klakow

11:30–12:00 *Question Answering with Lexical Chains Propagating Verb Arguments*
Adrian Novischi and Dan Moldovan

12:00–12:30 *Methods for Using Textual Entailment in Open-Domain Question Answering*
Sanda Harabagiu and Andrew Hickl

Session 14C: Semantics IV

11:00–11:30 *Using String-Kernels for Learning Semantic Parsers*
Rohit J. Kate and Raymond J. Mooney

11:30–12:00 *A Bootstrapping Approach to Unsupervised Detection of Cue Phrase Variants*
Rashid M. Abdalla and Simone Teufel

12:00–12:30 *Semantic Role Labeling via FrameNet, VerbNet and PropBank*
Ana-Maria Giuglea and Alessandro Moschitti

Session 14D: Resources II

11:00–11:30 *Multilingual Legal Terminology on the Jibiki Platform: The LexALP Project*
Gilles Sérasset, Francis Brunet-Manquat and Elena Chiocchetti

11:30–12:00 *Leveraging Reusability: Cost-Effective Lexical Acquisition for Large-Scale Ontology Translation*
G. Craig Murray, Bonnie J. Dorr, Jimmy Lin, Jan Hajič and Pavel Pecina

12:00–12:30 *Accurate Collocation Extraction Using a Multilingual Parser*
Violeta Seretan and Eric Wehrli

12:30–14:00 Lunch

Friday, 21 July 2006 (continued)

Session 15A: Machine Translation VI

14:00–14:30 *Scalable Inference and Training of Context-Rich Syntactic Translation Models*
Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang and Ignacio Thayer

14:30–15:00 *Modelling Lexical Redundancy for Machine Translation*
David Talbot and Miles Osborne

15:00–15:30 *Empirical Lower Bounds on the Complexity of Translational Equivalence*
Benjamin Wellington, Sonjia Waxmonsky and I. Dan Melamed

Session 15B: Language Modeling

14:00–14:30 *A Hierarchical Bayesian Language Model Based On Pitman-Yor Processes*
Yee Whye Teh

14:30–15:00 *A Phonetic-Based Approach to Chinese Chat Text Normalization*
Yunqing Xia, Kam-Fai Wong and Wenjie Li

15:00–15:30 *Discriminative Pruning of Language Models for Chinese Word Segmentation*
Jianfeng Li, Haifeng Wang, Dengjun Ren and Guohua Li

Session 15C: Information Retrieval II

14:00–14:30 *Novel Association Measures Using Web Search with Double Checking*
Hsin-Hsi Chen, Ming-Shun Lin and Yu-Chuan Wei

14:30–15:00 *Semantic Retrieval for the Accurate Identification of Relational Concepts in Massive Textbases*
Yusuke Miyao, Tomoko Ohta, Katsuya Masuda, Yoshimasa Tsuruoka, Kazuhiro Yoshida, Takashi Ninomiya and Jun'ichi Tsujii

15:00–15:30 *Exploring Distributional Similarity Based Models for Query Spelling Correction*
Mu Li, Muhua Zhu, Yang Zhang and Ming Zhou

Session 15D: Generation I

14:00–14:30 *Robust PCFG-Based Generation Using Automatically Acquired LFG Approximations*
Aoife Cahill and Josef van Genabith

14:30–15:00 *Incremental Generation of Spatial Referring Expressions in Situated Dialog*
John D. Kelleher and Geert-Jan M. Kruijff

15:00–15:30 *Learning to Predict Case Markers in Japanese*
Hisami Suzuki and Kristina Toutanova

15:30–16:00 Break

Friday, 21 July 2006 (continued)

Session 16A: Text Classification II

16:00–16:30 *Are These Documents Written from Different Perspectives? A Test of Different Perspectives Based on Statistical Distribution Divergence*
Wei-Hao Lin and Alexander Hauptmann

16:30–17:00 *Word Sense and Subjectivity*
Janyce Wiebe and Rada Mihalcea

Session 16B: Question Answering III

16:00–16:30 *Improving QA Accuracy by Question Inversion*
John Prager, Pablo Duboue and Jennifer Chu-Carroll

16:30–17:00 *Reranking Answers for Definitional QA Using Language Modeling*
Yi Chen, Ming Zhou and Shilong Wang

Session 16C: Grammars III

16:00–16:30 *Highly Constrained Unification Grammars*
Daniel Feinstein and Shuly Wintner

16:30–17:00 *A Polynomial Parsing Algorithm for the Topological Model: Synchronizing Constituent and Dependency Grammars, Illustrated by German Word Order Phenomena*
Kim Gerdes and Sylvain Kahane

Session 16D: Generation II

16:00–16:30 *Stochastic Language Generation Using WIDL-Expressions and its Application in Machine Translation and Summarization*
Radu Soricut and Daniel Marcu

16:30–17:00 *Learning to Say It Well: Reranking Realizations by Predicted Synthesis Quality*
Crystal Nakatsu and Michael White

17:00–17:30 Closing Session

Combination of Arabic Preprocessing Schemes for Statistical Machine Translation

Fatiha Sadat
Institute for Information Technology
National Research Council of Canada
fatiha.sadat@cnrc-nrc.gc.ca

Nizar Habash
Center for Computational Learning Systems
Columbia University
habash@cs.columbia.edu

Abstract

Statistical machine translation is quite robust when it comes to the choice of input representation. It only requires consistency between training and testing. As a result, there is a wide range of possible preprocessing choices for data used in statistical machine translation. This is even more so for morphologically rich languages such as Arabic. In this paper, we study the effect of different word-level preprocessing schemes for Arabic on the quality of phrase-based statistical machine translation. We also present and evaluate different methods for combining preprocessing schemes resulting in improved translation quality.

1 Introduction

Statistical machine translation (SMT) is quite robust when it comes to the choice of input representation. It only requires consistency between training and testing. As a result, there is a wide range of possible preprocessing choices for data used in SMT. This is even more so for morphologically rich languages such as Arabic. We use the term "preprocessing" to describe various input modifications applied to raw training and testing texts for SMT. Preprocessing includes different kinds of tokenization, stemming, part-of-speech (POS) tagging and lemmatization. The ultimate goal of preprocessing is to improve the quality of the SMT output by addressing issues such as sparsity in training data. We refer to a specific kind of preprocessing as a "scheme" and differentiate it from the "technique" used to obtain it. In a previous publication, we presented results describing six preprocessing schemes for Arabic (Habash and Sadat, 2006). These schemes were evaluated against three different techniques that vary in linguistic complexity; and across a learning curve of training sizes. Additionally, we reported on the effect of scheme/technique combination on genre variation between training and testing.

In this paper, we shift our attention to exploring and contrasting additional preprocessing schemes for Arabic and describing and evaluating different methods for combining them. We use a single technique throughout the experiments reported here. We show an improved MT performance when combining different schemes.

Similarly to Habash and Sadat (2006), the set of schemes we explore are all word-level. As such, we do not utilize any syntactic information. We define the word to be limited to written Modern Standard Arabic (MSA) strings separated by white space, punctuation and numbers.

Section 2 presents previous relevant research. Section 3 presents some relevant background on Arabic linguistics to motivate the schemes discussed in Section 4. Section 5 presents the tools and data sets used, along with the results of basic scheme experiments. Section 6 presents combination techniques and their results.

2 Previous Work

The anecdotal intuition in the field is that reduction of word sparsity often improves translation quality. This reduction can be achieved by increasing training data or via morphologically driven preprocessing (Goldwater and McClosky, 2005). Recent publications on the effect of morphology on SMT quality focused on morphologically rich languages such as German (Nießen and Ney, 2004); Spanish, Catalan, and Serbian (Popović

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 1–8,
Sydney, July 2006.

and Ney, 2004); and Czech (Goldwater and McClosky, 2005). They all studied the effects of various kinds of tokenization, lemmatization and POS tagging and show a positive effect on SMT quality.

Specifically considering Arabic, Lee (2004) investigated the use of automatic alignment of POS tagged English and affix-stem segmented Arabic to determine appropriate tokenizations. Her results show that morphological preprocessing helps, but only for the smaller corpora. As size increases, the benefits diminish. Our results are comparable to hers in terms of BLEU score and consistent in terms of conclusions. Other research on preprocessing Arabic suggests that minimal preprocessing, such as splitting off the conjunction +و *w*+ 'and', produces best results with very large training data (Och, 2005).

System combination for MT has also been investigated by different researchers. Approaches to combination generally either select one of the hypotheses produced by the different systems combined (Nomoto, 2004; Paul et al., 2005; Lee, 2005) or combine lattices/n-best lists from the different systems with different degrees of synthesis or mixing (Frederking and Nirenburg, 1994; Bangalore et al., 2001; Jayaraman and Lavie, 2005; Matusov et al., 2006). These different approaches use various translation and language models in addition to other models such as word matching, sentence and document alignment, system translation confidence, phrase translation lexicons, etc.

We extend on previous work by experimenting with a wider range of preprocessing schemes for Arabic and exploring their combination to produce better results.

3 Arabic Linguistic Issues

Arabic is a morphologically complex language with a large set of morphological features[1]. These features are realized using both concatenative morphology (affixes and stems) and templatic morphology (root and patterns). There is a variety of morphological and phonological adjustments that appear in word orthography and interact with orthographic variations. Next we discuss a subset of these issues that are necessary background for the later sections. We do not address derivational morphology (such as using roots as tokens) in this paper.

• **Orthographic Ambiguity**: The form of certain letters in Arabic script allows suboptimal orthographic variants of the same word to coexist in the same text. For example, variants of Hamzated Alif, أ > or إ < are often written without their Hamza (ء): ا *A*. These variant spellings increase the ambiguity of words. The Arabic script employs diacritics for representing short vowels and doubled consonants. These diacritics are almost always absent in running text, which increases word ambiguity. We assume all of the text we are using is undiacritized.

• **Clitics**: Arabic has a set of attachable clitics to be distinguished from inflectional features such as gender, number, person, voice, aspect, etc. These clitics are written attached to the word and thus increase the ambiguity of alternative readings. We can classify three degrees of cliticization that are applicable to a word base in a strict order:

```
[CONJ+ [PART+ [Al+ BASE +PRON]]]
```

At the deepest level, the BASE can have a definite article (+ال *Al*+ 'the') or a member of the class of pronominal enclitics, +PRON, (e.g. هم+ +*hm* 'their/them'). Pronominal enclitics can attach to nouns (as possessives) or verbs and prepositions (as objects). The definite article doesn't apply to verbs or prepositions. +PRON and *Al*+ cannot co-exist on nouns. Next comes the class of particle proclitics (PART+): +ل *l*+ 'to/for', +ب *b*+ 'by/with', +ك *k*+ 'as/such' and +س *s*+ 'will/future'. *b*+ and *k*+ are only nominal; *s*+ is only verbal and *l*+ applies to both nouns and verbs. At the shallowest level of attachment we find the conjunctions (CONJ+) +و *w*+ 'and' and +ف *f*+ 'so'. They can attach to everything.

• **Adjustment Rules**: Morphological features that are realized concatenatively (as opposed to templatically) are not always simply concatenated to a word base. Additional morphological, phonological and orthographic rules are applied to the word. An example of a morphological rule is the feminine morpheme, ة *+p (ta marbuta)*, which can only be word final. In medial position, it is turned into ت *t*. For example, مكتبة+هم *mktbp+hm* appears as مكتبتهم *mktbthm* 'their library'. An example of an orthographic rule is the deletion of the Alif (ا) of the definite article +ال *Al*+ in nouns when preceded by the preposition +ل *l*+ 'to/for' but not with any other prepositional proclitic.

[1] Arabic words have fourteen morphological features: POS, person, number, gender, voice, aspect, determiner proclitic, conjunctive proclitic, particle proclitic, pronominal enclitic, nominal case, nunation, idafa (possessed), and mood.

- **Templatic Inflections**: Some of the inflectional features in Arabic words are realized templatically by applying a different pattern to the Arabic root. As a result, extracting the lexeme (or lemma) of an Arabic word is not always an easy task and often requires the use of a morphological analyzer. One common example in Arabic nouns is *Broken Plurals*. For example, one of the plural forms of the Arabic word كاتب *kAtb* 'writer' is كتبة *ktbp* 'writers'. An alternative non-broken plural (concatenatively derived) is كاتبون *kAtbwn* 'writers'.

These phenomena highlight two issues related to the task at hand (preprocessing): First, ambiguity in Arabic words is an important issue to address. To determine whether a clitic or feature should be split off or abstracted off requires that we determine that said feature is indeed present in the word we are considering in context – not just that it is possible given an analyzer. Secondly, once a specific analysis is determined, the process of splitting off or abstracting off a feature must be clear on what the form of the resulting word should be. In principle, we would like to have whatever adjustments now made irrelevant (because of the missing feature) to be removed. This ensures reduced sparsity and reduced unnecessary ambiguity. For example, the word كتبتهم *ktbthm* has two possible readings (among others) as 'their writers' or 'I wrote them'. Splitting off the pronominal enclitic +هم *+hm* without normalizing the ت *t* to ة *p* in the nominal reading leads the coexistence of two forms of the noun كتبة *ktbp* and كتبت *ktbt*. This increased sparsity is only worsened by the fact that the second form is also the verbal form (thus increased ambiguity).

4 Arabic Preprocessing Schemes

Given Arabic morphological complexity, the number of possible preprocessing schemes is very large since any subset of morphological and orthographic features can be separated, deleted or normalized in various ways. To implement any preprocessing scheme, a preprocessing technique must be able to disambiguate amongst the possible analyses of a word, identify the features addressed by the scheme in the chosen analysis and process them as specified by the scheme. In this section we describe eleven different schemes.

4.1 Preprocessing Technique

We use the Buckwalter Arabic Morphological Analyzer (BAMA) (Buckwalter, 2002) to obtain possible word analyses. To select among these analyses, we use the Morphological Analysis and Disambiguation for Arabic (MADA) tool,[2] an off-the-shelf resource for Arabic disambiguation (Habash and Rambow, 2005). Being a disambiguation system of morphology, not word sense, MADA sometimes produces ties for analyses with the same inflectional features but different lexemes (resolving such ties require word-sense disambiguation). We resolve these ties in a consistent arbitrary manner: first in a sorted list of analyses.

Producing a preprocessing scheme involves removing features from the word analysis and regenerating the word without the split-off features. The regeneration ensures that the generated form is appropriately normalized by addressing various morphotactics described in Section 3. The generation is completed using the off-the-shelf Arabic morphological generation system Aragen (Habash, 2004).

This preprocessing technique we use here is the best performer amongst other explored techniques presented in Habash and Sadat (2006).

4.2 Preprocessing Schemes

Table 1 exemplifies the effect of different schemes on the same sentence.

- **ST**: Simple Tokenization is the baseline preprocessing scheme. It is limited to splitting off punctuations and numbers from words. For example the last non-white-space string in the example sentence in Table 1, "trkyA." is split into two tokens: "trkyA" and ".". An example of splitting numbers from words is the case of the conjunction +و *w+* 'and' which can prefix numerals such as when a list of numbers is described: و15 *w15* 'and 15'. This scheme requires no disambiguation. Any diacritics that appear in the input are removed in this scheme. This scheme is used as input to produce the other schemes.

- **ON**: Orthographic Normalization addresses the issue of sub-optimal spelling in Arabic. We use the Buckwalter answer undiacritized as the orthographically normalized form. An example of **ON** is the spelling of the last letter in the first and

[2] The version of MADA used in this paper was trained on the Penn Arabic Treebank (PATB) part 1 (Maamouri et al., 2004).

Table 1: Various Preprocessing Schemes

Input	wsynhY	Alr}ys	jwlth	bzyArp	AlY	trkyA.	
Gloss	and will finish	the president	tour his	with visit	to	Turkey	.
English	The president will finish his tour with a visit to Turkey.						
Scheme	**Baseline**						
ST	wsynhY	Alr}ys	jwlth	bzyArp	AlY	trkyA	.
ON	wsynhy	Alr}ys	jwlth	bzyArp	<lY	trkyA	.
D1	w+ synhy	Alr}ys	jwlth	bzyArp	<lY	trkyA	.
D2	w+ s+ ynhy	Alr}ys	jwlth	b+ zyArp	<lY	trkyA	.
D3	w+ s+ ynhy	Al+ r}ys	jwlp +P$_{3MS}$	b+ zyArp	<lY	trkyA	.
WA	w+ synhy	Alr}ys	jwlth	bzyArp	<lY	trkyA	.
TB	w+ synhy	Alr}ys	jwlp +P$_{3MS}$	b+ zyArp	<lY	trkyA	.
MR	w+ s+ y+ nhy	Al+ r}ys	jwl +p +h	b+ zyAr +p	<lY	trkyA	.
L1	>nhY$_{V}$	r}ys$_{N}$	jwlp$_{N}$	zyArp$_{N}$	<lY$_{P}$	trkyA$_{PN}$	.
L2	>nhY$_{VBP}$	r}ys$_{NN}$	jwlp$_{NN}$	zyArp$_{NN}$	<lY$_{IN}$	trkyA$_{NNP}$	.
EN	w+ s+ >nhY$_{VBP}$ +S$_{3MS}$	Al+ r}ys$_{NN}$	jwlp$_{NN}$ +P$_{3MS}$	b+ zyArp$_{NN}$	<lY$_{IN}$	trkyA$_{NNP}$	.

fifth words in the example in Table 1 (wsynhY and AlY, respectively). Since orthographic normalization is tied to the use of MADA and BAMA, all of the schemes we use here are normalized.

- **D1, D2, and D3**: Decliticization (degree 1, 2 and 3) are schemes that split off clitics in the order described in Section 3. **D1** splits off the class of conjunction clitics (*w+* and *f+*). **D2** is the same as **D1** plus splitting off the class of particles (*l+*, *k+*, *b+* and *s+*). Finally **D3** splits off what **D2** does in addition to the definite article *Al+* and all pronominal enclitics. A pronominal clitic is represented as its feature representation to preserve its uniqueness. (See the third word in the example in Table 1.) This allows distinguishing between the possessive pronoun and object pronoun which often look similar.

- **WA**: Decliticizing the conjunction *w+*. This is the simplest tokenization used beyond ON. It is similar to D1, but without including *f+*. This is included to compare to evidence in its support as best preprocessing scheme for very large data (Och, 2005).

- **TB**: Arabic Treebank Tokenization. This is the same tokenization scheme used in the Arabic Treebank (Maamouri et al., 2004). This is similar to **D3** but without the splitting off of the definite article *Al+* or the future particle *s+*.

- **MR**: Morphemes. This scheme breaks up words into stem and affixival morphemes. It is identical to the initial tokenization used by Lee (2004).

- **L1 and L2**: Lexeme and POS. These reduce a word to its lexeme and a POS. **L1** and **L2** differ in the set of POS tags they use. **L1** uses the simple POS tags advocated by Habash and Rambow (2005) (15 tags); while **L2** uses the reduced tag set used by Diab et al. (2004) (24 tags). The latter is modeled after the English Penn POS tag set. For example, Arabic nouns are differentiated for being singular (NN) or Plural/Dual (NNS), but adjectives are not even though, in Arabic, they inflect exactly the same way nouns do.

- **EN**: English-like. This scheme is intended to minimize differences between Arabic and English. It decliticizes similarly to **D3**, but uses Lexeme and POS tags instead of the regenerated word. The POS tag set used is the reduced Arabic Treebank tag set (24 tags) (Maamouri et al., 2004; Diab et al., 2004). Additionally, the subject inflection is indicated explicitly as a separate token. We do not use any additional information to remove specific features using alignments or syntax (unlike, e.g. removing all but one *Al+* in noun phrases (Lee, 2004)).

4.3 Comparing Various Schemes

Table 2 compares the different schemes in terms of the number of tokens, number of out-of-vocabulary (OOV) tokens, and perplexity. These statistics are computed over the MT04 set, which we use in this paper to report SMT results (Section 5). Perplexity is measured against a language model constructed from the Arabic side of the parallel corpus used in the MT experiments (Section 5).

Obviously the more verbose a scheme is, the bigger the number of tokens in the text. The **ST**, **ON**, **L1**, and **L2** share the same number of tokens because they all modify the word without splitting off any of its morphemes or features. The increase in the number of tokens is in inverse correlation

Table 2: Scheme Statistics

Scheme	Tokens	OOVs	Perplexity
ST	36000	1345	1164
ON	36000	1212	944
D1	38817	1016	582
D2	40934	835	422
D3	52085	575	137
WA	38635	1044	596
TB	42880	662	338
MR	62410	409	69
L1	36000	392	401
L2	36000	432	460
EN	55525	432	103

with the number of OOVs and perplexity. The only exceptions are **L1** and **L2**, whose low OOV rate is the result of the reductionist nature of the scheme, which does not preserve morphological information.

5 Basic Scheme Experiments

We now describe the system and the data sets we used to conduct our experiments.

5.1 Portage

We use an off-the-shelf phrase-based SMT system, Portage (Sadat et al., 2005). For training, Portage uses IBM word alignment models (models 1 and 2) trained in both directions to extract phrase tables in a manner resembling (Koehn, 2004a). Trigram language models are implemented using the SRILM toolkit (Stolcke, 2002). Decoding weights are optimized using Och's algorithm (Och, 2003) to set weights for the four components of the log-linear model: language model, phrase translation model, distortion model, and word-length feature. The weights are optimized over the BLEU metric (Papineni et al., 2001). The Portage decoder, Canoe, is a dynamic-programming beam search algorithm resembling the algorithm described in (Koehn, 2004a).

5.2 Experimental data

All of the training data we use is available from the Linguistic Data Consortium (LDC). We use an Arabic-English parallel corpus of about 5 million words for translation model training data.[3] We created the English language model from the English side of the parallel corpus together with 116 million words the English Gigaword Corpus (LDC2005T12) and 128 million words from the English side of the UN Parallel corpus (LDC2004E13).[4]

English preprocessing simply included lower-casing, separating punctuation from words and splitting off "'s". The same preprocessing was used on the English data for all experiments. Only Arabic preprocessing was varied. Decoding weight optimization was done using a set of 200 sentences from the 2003 NIST MT evaluation test set (MT03). We report results on the 2004 NIST MT evaluation test set (MT04) The experiment design and choices of schemes and techniques were done independently of the test set. The data sets, MT03 and MT04, include one Arabic source and four English reference translations. We use the evaluation metric BLEU-4 (Papineni et al., 2001) although we are aware of its caveats (Callison-Burch et al., 2006).

5.3 Experimental Results

We conducted experiments with all schemes discussed in Section 4 with different training corpus sizes: 1%, 10%, 50% and 100%. The results of the experiments are summarized in Table 3. These results are not English case sensitive. All reported scores must have over 1.1% BLEU-4 difference to be significant at the 95% confidence level for 1% training. For all other training sizes, the difference must be over 1.7% BLEU-4. Error intervals were computed using bootstrap resampling (Koehn, 2004b).

Across different schemes, **EN** performs the best under scarce-resource condition; and **D2** performs as best under large resource conditions. The results from the learning curve are consistent with previous published work on using morphological preprocessing for SMT: deeper morph analysis helps for small data sets, but the effect is diminished with more data. One interesting observation is that for our best performing system (**D2**), the BLEU score at 50% training (35.91) was higher than the baseline **ST** at 100% training data (34.59). This relationship is not consistent across the rest of the experiments. **ON** improves over the baseline

[3] The parallel text includes Arabic News (LDC2004T17), eTIRR (LDC2004E72), English translation of Arabic Treebank (LDC2005E46), and Ummah (LDC2004T18).

[4] The SRILM toolkit has a limit on the size of the training corpus. We selected portions of additional corpora using a heuristic that picks documents containing the word "Arab" only. The Language model created using this heuristic had a bigger improvement in BLEU score (more than 1% BLEU-4) than a randomly selected portion of equal size.

Table 3: Scheme Experiment Results (BLEU-4)

	Training Data			
Scheme	1%	10%	50%	100%
ST	9.42	22.92	31.09	34.59
ON	10.71	24.3	32.52	35.91
D1	13.11	26.88	33.38	36.06
D2	14.19	27.72	35.91	37.10
D3	16.51	28.69	34.04	34.33
WA	13.12	26.29	34.24	35.97
TB	14.13	28.71	35.83	36.76
MR	11.61	27.49	32.99	34.43
L1	14.63	24.72	31.04	32.23
L2	14.87	26.72	31.28	33.00
EN	17.45	28.41	33.28	34.51

but only statistically significantly at the 1% level. The results for **WA** are generally similar to **D1**. This makes sense since *w*+ is by far the most common of the two conjunctions **D1** splits off. The **TB** scheme behaves similarly to **D2**, the best scheme we have. It outperformed **D2** in few instances, but the difference were not statistically significant. **L1** and **L2** behaved similar to **EN** across the different training size. However, both were always worse than **EN**. Neither variant was consistently better than the other.

6 System Combination

The complementary variation in the behavior of different schemes under different resource size conditions motivated us to investigate system combination. The intuition is that even under large resource conditions, some words will occur very infrequently that the only way to model them is to use a technique that behaves well under poor resource conditions.

We conducted an oracle study into system combination. An oracle combination output was created by selecting for each input sentence the output with the highest sentence-level BLEU score. We recognize that since the brevity penalty in BLEU is applied globally, this score may not be the highest possible combination score. The oracle combination has a 24% improvement in BLEU score (from 37.1 in best system to 46.0) when combining all eleven schemes described in this paper. This shows that combining of output from all schemes has a large *potential* of improvement over all of the different systems and that the different schemes are complementary in some way.

In the rest of this section we describe two successful methods for system combination of different schemes: rescoring-only combination (ROC) and decoding-plus-rescoring combination (DRC). All of the experiments use the same training data, test data (MT04) and preprocessing schemes described in the previous section.

6.1 Rescoring-only Combination

This "shallow" approach rescores all the one-best outputs generated from separate scheme-specific systems and returns the top choice. Each scheme-specific system uses its own scheme-specific preprocessing, phrase-tables, and decoding weights. For rescoring, we use the following features:

- The four basic features used by the decoder: trigram language model, phrase translation model, distortion model, and word-length feature.
- IBM model 1 and IBM model 2 probabilities in both directions.
  We call the union of these two sets of features *standard*.
- The perplexity of the preprocessed source sentence (PPL) against a source language model as described in Section 4.3.
- The number of out-of-vocabulary words in the preprocessed source sentence (OOV).
- Length of the preprocessed source sentence (SL).
- An encoding of the specific scheme used (SC). We use a one-hot coding approach with 11 separate binary features, each corresponding to a specific scheme.

Optimization of the weights on the rescoring features is carried out using the same max-BLEU algorithm and the same development corpus described in Section 5.

Results of different sets of features with the ROC approach are presented in Table 4. Using *standard* features with all eleven schemes, we obtain a BLEU score of 34.87 – a significant drop from the best scheme system (D2, 37.10). Using different subsets of features or limiting the number of systems to the best four systems (D2, TB, D1 and WA), we get some improvements. The best results are obtained using all schemes with *standard* features plus perplexity and scheme coding. The improvements are small; however they are statistically significant (see Section 6.3).

Table 4: ROC Approach Results

Combination	All Schemes	4 Best Schemes
standard	34.87	37.12
+PPL+SC	**37.58**	**37.45**
+PPL+SC+OOV	37.40	
+PPL+SC+OOV+SL	37.39	
+PPL+SC+SL	37.15	

6.2 Decoding-plus-Rescoring Combination

This “deep” approach allows the decoder to consult several different phrase tables, each generated using a different preprocessing scheme; just as with ROC, there is a subsequent rescoring stage. A problem with DRC is that the decoder we use can only cope with one format for the source sentence at a time. Thus, we are forced to designate a particular scheme as *privileged* when the system is carrying out decoding. The privileged preprocessing scheme will be the one applied to the source sentence. Obviously, words and phrases in the preprocessed source sentence will more frequently match the phrases in the privileged phrase table than in the non-privileged ones. Nevertheless, the decoder may still benefit from having access to all the tables. For each choice of a privileged scheme, optimization of log-linear weights is carried out (with the version of the development set preprocessed in the same privileged scheme).

The middle column of Table 5 shows the results for 1-best output from the decoder under different choices of the privileged scheme. The best-performing system in this column has as its privileged preprocessing scheme TB. The decoder for this system uses TB to preprocess the source sentence, but has access to a log-linear combination of information from all 11 preprocessing schemes.

The final column of Table 5 shows the results of rescoring the concatenation of the 1-best outputs from each of the combined systems. The rescoring features used are the same as those used for the ROC experiments. For rescoring, a privileged preprocessing scheme is chosen and applied to the development corpus. We chose TB for this (since it yielded the best result when chosen to be privileged at the decoding stage). Applied to 11 schemes, this yields the best result so far: 38.67 BLEU. Combining the 4 best pre-processing schemes (D2, TB, D1, WA) yielded a lower BLEU score (37.73). These results show that combining phrase tables from different schemes have a positive effect on MT performance.

Table 5: DRC Approach Results

Combination	Decoding		Rescoring
	Scheme	1-best	Standard+PPL
All schemes	D2	37.16	**38.67**
	TB	38.24	
	D1	37.89	
	WA	36.91	
	ON	36.42	
	ST	34.27	
	EN	30.78	
	MR	34.65	
	D3	34.73	
	L2	32.25	
	L1	30.47	
4 best schemes	D2	37.39	**37.73**
	TB	37.53	
	D1	36.05	
	WA	37.53	

Table 6: Statistical Significance using Bootstrap Resampling

DRC	ROC	D2	TB	D1	WA	ON
100	0	0	0	0	0	0
	97.7	2.2	0.1	0	0	0
		92.1	7.9	0	0	0
			98.8	0.7	0.3	0.2
				53.8	24.1	22.1
					59.3	40.7

6.3 Significance Test

We use bootstrap resampling to compute MT statistical significance as described in (Koehn, 2004a). The results are presented in Table 6. Comparing the 11 individual systems and the two combinations DRC and ROC shows that DRC is significantly better than the other systems – DRC got a max BLEU score in 100% of samples. When excluding DRC from the comparison set, ROC got max BLEU score in 97.7% of samples, while D2 and TB got max BLEU score in 2.2% and 0.1% of samples, respectively. The difference between ROC and D2 and ATB is statistically significant.

7 Conclusions and Future Work

We motivated, described and evaluated several preprocessing schemes for Arabic. The choice of a preprocessing scheme is related to the size of available training data. We also presented two techniques for scheme combination. Although the results we got are not as high as the oracle scores, they are statistically significant.

In the future, we plan to study additional scheme variants that our current results support as potentially helpful. We plan to include more syntactic knowledge. We also plan to continue investigating combination techniques at the sentence and sub-sentence levels. We are especially interested in the relationship between alignment and decoding and the effect of preprocessing scheme on both.

Acknowledgments

This paper is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA. We thank Roland Kuhn and George Forster for helpful discussions and support.

References

S. Bangalore, G. Bordel, and G. Riccardi. 2001. Computing Consensus Translation from Multiple Machine Translation Systems. In *Proc. of IEEE Automatic Speech Recognition and Understanding Workshop, Italy.*

T. Buckwalter. 2002. Buckwalter Arabic Morphological Analyzer Version 1.0. Linguistic Data Consortium, University of Pennsylvania. Catalog: LDC2002L49.

C. Callison-Burch, M. Osborne, and P. Koehn. 2006. Re-evaluating the Role of Bleu in Machine Translation Research. In *Proc. of the European Chapter of the Association for Computational Linguistics (EACL)*, Trento, Italy.

M. Diab, K. Hacioglu, and D. Jurafsky. 2004. Automatic Tagging of Arabic Text: From Raw Text to Base Phrase Chunks. In *Proc. of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Boston, MA.

R. Frederking and S. Nirenburg. 2005. Three Heads are Better Than One. In *Proc. of Applied Natural Language Processing, Stuttgart, Germany.*

S. Goldwater and D. McClosky. 2005. Improving Statistical MT through Morphological Analysis. In *Proc. of Empirical Methods in Natural Language Processing (EMNLP)*, Vancouver, Canada.

N. Habash and O. Rambow. 2005. Tokenization, Morphological Analysis, and Part-of-Speech Tagging for Arabic in One Fell Swoop. In *Proc. of Association for Computational Linguistics (ACL)*, Ann Arbor, Michigan.

N. Habash and F. Sadat. 2006. Arabic Preprocessing Schemes for Statistical Machine Translation. In *Proc. of NAACL*, Brooklyn, New York.

N. Habash. 2004. Large Scale Lexeme-based Arabic Morphological Generation. In *Proc. of Traitement Automatique du Langage Naturel (TALN).* Fez, Morocco.

S. Jayaraman and A. Lavie. 2005. Multi-Engine Machine Translation Guided by Explicit Word Matching. In *Proc. of the Association of Computational Linguistics (ACL)*, Ann Arbor, MI.

P. Koehn. 2004a. Pharaoh: a Beam Search Decoder for Phrase-based Statistical Machine Translation Models. In *Proc. of the Association for Machine Translation in the Americas (AMTA).*

P. Koehn. 2004b. Statistical Significance Tests for Machine Translation Evaluation. In *Proc. of the EMNLP*, Barcelona, Spain.

Y. Lee. 2004. Morphological Analysis for Statistical Machine Translation. In *Proc. of NAACL*, Boston, MA.

Y. Lee. 2005. IBM Statistical Machine Translation for Spoken Languages. In *Proc. of International Workshop on Spoken Language Translation (IWSLT).*

M. Maamouri, A. Bies, and T. Buckwalter. 2004. The Penn Arabic Treebank: Building a Large-scale Annotated Arabic Corpus. In *Proc. of NEMLAR Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

E. Matusov, N. Ueffing, H. Ney 2006. Computing Consensus Translation from Multiple Machine Translation Systems Using Enhanced Hypotheses Alignment. In *Proc. of EACL, Trento, Italy*.

S. Nießen and H. Ney. 2004. Statistical Machine Translation with Scarce Resources Using Morpho-syntactic Information. *Computational Linguistics*, 30(2).

T. Nomoto. 2004. Multi-Engine Machine Translation with Voted Language Model. In *Proc. of ACL*, Barcelona, Spain.

F. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of the ACL*, Sapporo, Japan.

F. Och. 2005. Google System Description for the 2005 Nist MT Evaluation. In *MT Eval Workshop (unpublished talk)*.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2001. Bleu: a Method for Automatic Evaluation of Machine Translation. Technical Report RC22176(W0109-022), IBM Research Division, Yorktown Heights, NY.

M. Paul, T. Doi, Y. Hwang, K. Imamura, H. Okuma, and E. Sumita. 2005. Nobody is Perfect: ATR's Hybrid Approach to Spoken Language Translation. In *Proc. of IWSLT*.

M. Popović and H. Ney. 2004. Towards the Use of Word Stems and Suffixes for Statistical Machine Translation. In *Proc. of Language Resources and Evaluation (LREC)*, Lisbon, Portugal.

F. Sadat, H. Johnson, A. Agbago, G. Foster, R. Kuhn, J. Martin, and A. Tikuisis. 2005. Portage: A Phrase-based Machine Translation System. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, Ann Arbor, Michigan.

A. Stolcke. 2002. Srilm - An Extensible Language Modeling Toolkit. In *Proc. of International Conference on Spoken Language Processing*.

Going Beyond AER: An Extensive Analysis of Word Alignments and Their Impact on MT

Necip Fazil Ayan and **Bonnie J. Dorr**
Institute of Advanced Computer Studies (UMIACS)
University of Maryland
College Park, MD 20742
{nfa,bonnie}@umiacs.umd.edu

Abstract

This paper presents an extensive evaluation of five different alignments and investigates their impact on the corresponding MT system output. We introduce new measures for intrinsic evaluations and examine the distribution of phrases and untranslated words during decoding to identify which characteristics of different alignments affect translation. We show that precision-oriented alignments yield better MT output (translating more words and using longer phrases) than recall-oriented alignments.

1 Introduction

Word alignments are a by-product of statistical machine translation (MT) and play a crucial role in MT performance. In recent years, researchers have proposed several algorithms to generate word alignments. However, evaluating word alignments is difficult because even humans have difficulty performing this task.

The state-of-the art evaluation metric—alignment error rate (AER)—attempts to balance the precision and recall scores at the level of alignment links (Och and Ney, 2000). Other metrics assess the impact of alignments externally, e.g., different alignments are tested by comparing the corresponding MT outputs using automated evaluation metrics (e.g., BLEU (Papineni et al., 2002) or METEOR (Banerjee and Lavie, 2005)). However, these studies showed that AER and BLEU do not correlate well (Callison-Burch et al., 2004; Goutte et al., 2004; Ittycheriah and Roukos, 2005). Despite significant AER improvements achieved by several researchers, the improvements in BLEU scores are insignificant or, at best, small.

This paper demonstrates the difficulty in assessing whether alignment quality makes a difference in MT performance. We describe the impact of certain alignment characteristics on MT performance but also identify several alignment-related factors that impact MT performance regardless of the quality of the initial alignments. In so doing, we begin to answer long-standing questions about the value of alignment in the context of MT.

We first evaluate 5 different word alignments intrinsically, using: (1) community-standard metrics—precision, recall and AER; and (2) a new measure called *consistent phrase error rate* (CPER). Next, we observe the impact of different alignments on MT performance. We present BLEU scores on a phrase-based MT system, Pharaoh (Koehn, 2004), using five different alignments to extract phrases. We investigate the impact of different settings for phrase extraction, lexical weighting, maximum phrase length and training data. Finally, we present a quantitative analysis of which phrases are chosen during the actual decoding process and show how the distribution of the phrases differ from one alignment into another.

Our experiments show that precision-oriented alignments yield better phrases for MT than recall-oriented alignments. Specifically, they cover a higher percentage of our test sets and result in fewer untranslated words and selection of longer phrases during decoding.

The next section describes work related to our alignment evaluation approach. Following this we outline different intrinsic evaluation measures of alignment and we propose a new measure to evaluate word alignments within phrase-based MT framework. We then present several experiments to measure the impact of different word alignments on a phrase-based MT system, and investigate how different alignments change the phrase

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 9–16,
Sydney, July 2006.

selection in the same MT system.

2 Related Work

Starting with the IBM models (Brown et al., 1993), researchers have developed various statistical word alignment systems based on different models, such as hidden Markov models (HMM) (Vogel et al., 1996), log-linear models (Och and Ney, 2003), and similarity-based heuristic methods (Melamed, 2000). These methods are unsupervised, i.e., the only input is large parallel corpora. In recent years, researchers have shown that even using a limited amount of manually aligned data improves word alignment significantly (Callison-Burch et al., 2004). Supervised learning techniques, such as perceptron learning, maximum entropy modeling or maximum weighted bipartite matching, have been shown to provide further improvements on word alignments (Ayan et al., 2005; Moore, 2005; Ittycheriah and Roukos, 2005; Taskar et al., 2005).

The standard technique for evaluating word alignments is to represent alignments as a set of links (i.e., pairs of words) and to compare the generated alignment against manual alignment of the same data at the level of links. Manual alignments are represented by two sets: Probable (P) alignments and Sure (S) alignments, where $S \subseteq P$. Given A, P and S, the most commonly used metrics—precision (Pr), recall (Rc) and alignment error rate (AER)—are defined as follows:

$$Pr = \frac{|A \cap P|}{|A|} \quad Rc = \frac{|A \cap S|}{|S|}$$

$$AER = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$$

Another approach to evaluating alignments is to measure their impact on an external application, e.g., statistical MT. In recent years, phrase-based systems (Koehn, 2004; Chiang, 2005) have been shown to outperform word-based MT systems; therefore, in this paper, we use a publicly-available phrase-based MT system, Pharaoh (Koehn, 2004), to investigate the impact of different alignments.

Although it is possible to estimate phrases directly from a training corpus (Marcu and Wong, 2002), most phrase-based MT systems (Koehn, 2004; Chiang, 2005) start with a word alignment and extract phrases that are consistent with the given alignment. Once the consistent phrases are extracted, they are assigned multiple scores (such as translation probabilities and lexical weights), and the decoder's job is to choose the correct phrases based on those scores using a log-linear model.

Test

Lang Pair	**# of Sent's**	**# Words (en/fl)**	**Source**
en-ch	491	14K/12K	NIST MTEval'2002
en-ar	450	13K/11K	NIST MTEval'2003

Training

en-ch	107K	4.1M/3.3M	FBIS
en-ar	44K	1.4M/1.1M	News + Treebank

Table 1: Test and Training Data Used for Experiments

3 Intrinsic Evaluation of Alignments

Our goal is to compare different alignments and to investigate how their characteristics affect the MT systems. We evaluate alignments in terms of precision, recall, alignment error rate (AER), and a new measure called consistent phrase error rate (CPER).

We focus on 5 different alignments obtained by combining two uni-directional alignments. Each uni-directional alignment is the result of running GIZA++ (Och, 2000b) in one of two directions (source-to-target and vice versa) with default configurations. The combined alignments that are used in this paper are as follows:

1. Union of both directions ($\mathrm{S_U}$),
2. Intersection of both directions ($\mathrm{S_I}$),
3. A heuristic based combination technique called *grow-diag-final* ($\mathrm{S_G}$), which is the default alignment combination heuristic employed in Pharaoh (Koehn, 2004),

4-5. Two supervised alignment combination techniques ($\mathrm{S_A}$ and $\mathrm{S_B}$) using 2 and 4 input alignments as described in (Ayan et al., 2005).

This paper examines the impact of alignments according to their orientation toward precision or recall. Among the five alignments above, $\mathrm{S_U}$ and $\mathrm{S_G}$ are recall-oriented while the other three are precision-oriented. $\mathrm{S_B}$ is an improved version of $\mathrm{S_A}$ which attempts to increase recall without a significant sacrifice in precision.

Manually aligned data from two language pairs are used in our intrinsic evaluations using the five combinations above. A summary of the training and test data is presented in Table 1.

Our gold standard for each language pair is a manually aligned corpus. English-Chinese annotations distinguish between sure and probable alignment links, but English-Arabic annotations do not. The details of how the annotations are done can be found in (Ayan et al., 2005) and (Ittycheriah and Roukos, 2005).

3.1 Precision, Recall and AER

Table 2 presents the precision, recall, and AER for 5 different alignments on 2 language pairs. For each of these metrics, a different system achieves the best score – respectively, these are $\mathrm{S_I}$, $\mathrm{S_U}$, and $\mathrm{S_B}$. $\mathrm{S_U}$ and $\mathrm{S_G}$ yield low precision, high recall alignments. In contrast, $\mathrm{S_I}$ yields very high precision but very low recall. $\mathrm{S_A}$ and $\mathrm{S_B}$ attempt to balance these two measures but their precision is still higher than their recall. Both systems have nearly the same precision but $\mathrm{S_B}$ yields significantly higher recall than $\mathrm{S_A}$.

Align. Sys.	en-ch			en-ar		
	Pr	Rc	AER	Pr	Rc	AER
$\mathrm{S_U}$	58.3	**84.5**	31.6	56.0	**84.1**	32.8
$\mathrm{S_G}$	61.9	82.6	29.7	60.2	83.0	30.2
$\mathrm{S_I}$	**94.8**	53.6	31.2	**96.1**	57.1	28.4
$\mathrm{S_A}$	87.0	74.6	19.5	88.6	71.1	21.1
$\mathrm{S_B}$	87.8	80.5	**15.9**	90.1	76.1	**17.5**

Table 2: Comparison of 5 Different Alignments using AER (on English-Chinese and English-Arabic)

3.2 Consistent Phrase Error Rate

In this section, we present a new method, called *consistent phrase error rate* (CPER), for evaluating word alignments in the context of phrase-based MT. The idea is to compare phrases consistent with a given alignment against phrases that would be consistent with human alignments.

CPER is similar to AER but operates at the phrase level instead of at the word level. To compute CPER, we define a link in terms of the position of its start and end words in the phrases. For instance, the phrase link (i_1, i_2, j_1, j_2) indicates that the English phrase $e_{i_1}, \ldots, e_{i_2}$ and the FL phrase $f_{j_1}, \ldots, f_{j_2}$ are consistent with the given alignment. Once we generate the set of phrases P_A and P_G that are consistent with a given alignment A and a manual alignment G, respectively, we compute precision (*Pr*), recall (*Rc*), and CPER as follows:[1]

$$Pr = \frac{|P_A \cap P_G|}{|P_A|} \quad Rc = \frac{|P_A \cap P_G|}{|P_G|}$$

$$CPER = 1 - \frac{2 \times Pr \times Rc}{Pr + Rc}$$

	Chinese		Arabic	
Align.	CPER-3	CPER-7	CPER-3	CPER-7
$\mathrm{S_U}$	63.2	73.3	55.6	67.1
$\mathrm{S_G}$	59.5	69.4	52.0	62.6
$\mathrm{S_I}$	50.8	69.8	50.7	67.6
$\mathrm{S_A}$	40.8	51.6	42.0	54.1
$\mathrm{S_B}$	36.8	45.1	36.1	46.6

Table 3: Consistent Phrase Error Rates with Maximum Phrase Lengths of 3 and 7

CPER penalizes incorrect or missing alignment links more severely than AER. While computing AER, an incorrect alignment link reduces the number of correct alignment links by 1, affecting precision and recall slightly. Similarly, if there is a missing link, only the recall is reduced slightly. However, when computing CPER, an incorrect or missing alignment link might result in more than one phrase pair being eliminated from or added to the set of phrases. Thus, the impact is more severe on both precision and recall.

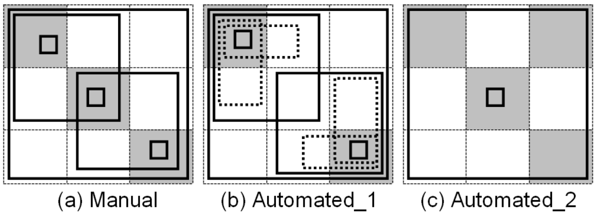

(a) Manual (b) Automated_1 (c) Automated_2

Figure 1: Sample phrases that are generated from a human alignment and an automated alignment: Gray cells show the alignment links, and rectangles show the possible phrases.

In Figure 1, the first box represents a manual alignment and the other two represent automated alignments A. In the case of a missing alignment link (Figure 1b), P_A includes 9 valid phrases. For this alignment, $AER = 1 - (2 \times 2/2 \times 2/3)/(2/2 + 2/3) = 0.2$ and $CPER = 1 - (2 \times 5/9 \times 5/6)/(5/9 + 5/6) = 0.33$. In the case of an incorrect alignment link (Figure 1c), P_A includes only 2 valid phrases, which results in a higher CPER $(1 - (2 \times 2/2 \times 2/6)/(2/2 + 2/6) = 0.49)$ but a lower AER $(1 - (2 \times 3/4 \times 3/3)/(3/4 + 3/3) = 0.14)$.

Table 3 presents the CPER values on two different language pairs, using 2 different maximum phrase lengths. For both maximum phrase lengths, $\mathrm{S_A}$ and $\mathrm{S_B}$ yield the lowest CPER. For all 5 alignments—in both languages—CPER increases as the length of the phrase increases. For all alignments except $\mathrm{S_I}$, this amount of increase is nearly the same on both languages. Since $\mathrm{S_I}$ contains very few alignment points, the number of generated phrases dramatically increases, yielding

[1] Note that CPER is equal to 1 - F-score.

poor precision and CPER as the maximum phrase length increases.

4 Evaluating Alignments within MT

We now move from intrinsic measurement to extrinsic measurement using an off-the-shelf phrase-based MT system Pharaoh (Koehn, 2004). Our goal is to identify the characteristics of alignments that change MT behavior and the types of changes induced by these characteristics.

All MT system components were kept the same in our experiments except for the component that generates a phrase table from a given alignment. We used the corpora presented in Table 1 to train the MT system. The phrases were scored using translation probabilities and lexical weights in two directions and a phrase penalty score. We also use a language model, a distortion model and a word penalty feature for MT.

We measure the impact of different alignments on Pharaoh using three different settings:

1. Different maximum phrase length,
2. Different sizes of training data, and
3. Different lexical weighting.

For maximum phrase length, we used 3 (based on what was suggested by (Koehn et al., 2003) and 7 (the default maximum phrase length in Pharaoh).

For lexical weighting, we used the original weighting scheme employed in Pharaoh and a modified version. We realized that the publicly-available implementation of Pharaoh computes the lexical weights only for non-NULL alignment links. As a consequence, loose phrases containing NULL-aligned words along their edges receive the same lexical weighting as tight phrases without NULL-aligned words along the edges. We therefore adopted a modified weighting scheme following (Koehn et al., 2003), which incorporates NULL alignments.

MT output was evaluated using the standard evaluation metric BLEU (Papineni et al., 2002).[2] The parameters of the MT System were optimized for BLEU metric on NIST MTEval'2002 test sets using minimum error rate training (Och, 2003), and the systems were tested on NIST MTEval'2003 test sets for both languages.

The SRI Language Modeling Toolkit was used to train a trigram model with modified Kneser-Ney smoothing on 155M words of English newswire text, mostly from the Xinhua portion of the Gigaword corpus. During decoding, the number of English phrases per FL phrase was limited to 100 and phrase distortion was limited to 4.

4.1 BLEU Score Comparison

Table 4 presents the BLEU scores for Pharaoh runs on Chinese with five different alignments using different settings for maximum phrase length (3 vs. 7), size of training data (107K vs. 241K), and lexical weighting (original vs. modified).[3]

The modified lexical weighting yields huge improvements when the alignment leaves several words unaligned: the BLEU score for $\mathrm{S_A}$ goes from 24.26 to 25.31 and the BLEU score for $\mathrm{S_B}$ goes from 23.91 to 25.38. In contrast, when the alignments contain a high number of alignment links (e.g., $\mathrm{S_U}$ and $\mathrm{S_G}$), modifying lexical weighting does not bring significant improvements because the number of phrases containing unaligned words is relatively low. Increasing the phrase length increases the BLEU scores for all systems by nearly 0.7 points and increasing the size of the training data increases the BLEU scores by 1.5-2 points for all systems. For all settings, $\mathrm{S_U}$ yields the lowest BLEU scores while $\mathrm{S_B}$ clearly outperforms the others.

Table 5 presents BLEU scores for Pharaoh runs on 5 different alignments on English-Arabic, using different settings for lexical weighting and maximum phrase lengths.[4] Using the original lexical weighting, $\mathrm{S_A}$ and $\mathrm{S_B}$ perform better than the others while $\mathrm{S_U}$ and $\mathrm{S_I}$ yield the worst results. Modifying the lexical weighting leads to slight reductions in BLEU scores for $\mathrm{S_U}$ and $\mathrm{S_G}$, but improves the scores for the other 3 alignments significantly. Finally, increasing the maximum phrase length to 7 leads to additional improvements in BLEU scores, where $\mathrm{S_G}$ and $\mathrm{S_U}$ benefit nearly 2 BLEU points. As in English-Chinese, the worst BLEU scores are obtained by $\mathrm{S_U}$ while the best scores are produced by $\mathrm{S_B}$.

As we see from the tables, the relation between intrinsic alignment measures (AER and CPER)

[2] We used the NIST script (version 11a) for BLEU with its default settings: case-insensitive matching of n-grams up to $n = 4$, and the shortest reference sentence for the brevity penalty. The words that were not translated during decoding were deleted from the MT output before running the BLEU script.

[3] We could not run $\mathrm{S_B}$ on the larger corpus because of the lack of required inputs.

[4] Due to lack of additional training data, we could not do experiments using different sizes of training data on English-Arabic.

Alignment	Original Max Phr Len = 3 |Corpus| = 107K	Modified Max Phr Len=3 |Corpus| = 107K	Modified Max Phr Len=7 |Corpus| = 107K	Modified Max Phr Len=3 |Corpus| = 241K
S_U	22.56	22.66	23.30	24.40
S_G	23.65	23.79	24.48	25.54
S_I	23.60	23.97	24.76	26.06
S_A	24.26	25.31	25.99	26.92
S_B	23.91	25.38	26.14	N/A

Table 4: BLEU Scores on English-Chinese with Different Lexical Weightings, Maximum Phrase Lengths and Training Data

Alignment	LW=Org
	MPL=3
		LW=Mod
		MPL=3
			LW=Mod
			MPL=7

S_U	41.97	41.72	43.50
S_G	44.06	43.82	45.78
S_I	42.29	42.76	43.88
S_A	44.49	45.23	46.06
S_B	44.92	45.39	46.66

Table 5: BLEU Scores on English-Arabic with Different Lexical Weightings and Maximum Phrase Lengths

and the corresponding BLEU scores varies, depending on the language, lexical weighting, maximum phrase length, and training data size. For example, using a modified lexical weighting, the systems are ranked according to their BLEU scores as follows: S_B, S_A, S_G, S_I, S_U—an ordering that differs from that of AER but is identical to that of CPER (with a phrase length of 3) for Chinese. On the other hand, in Arabic, both AER and CPER provide a slightly different ranking from that of BLEU, with S_G and S_I swapping places.

4.2 Tight vs. Loose Phrases

To demonstrate how alignment-related components of the MT system might change the translation quality significantly, we did an additional experiment to compare different techniques for extracting phrases from a given alignment. Specifically, we are comparing two techniques for phrase extraction:

1. Loose phrases (the original ‘consistent phrase extraction’ method)
2. Tight phrases (the set of phrases where the first/last words on each side are forced to align to some word in the phrase pair)

Using tight phrases penalizes alignments with many unaligned words, whereas using loose phrases rewards them. Our goal is to compare the performance of precision-oriented vs. recall-oriented alignments when we allow only tight phrases in the phrase extraction step. To simplify things, we used only 2 alignments: S_G, the best recall-oriented alignment, and S_B, the best precision-oriented alignment. For this experiment, we used modified lexical weighting and a maximum phrase length of 7.

	Chinese		Arabic	
Alignment	Loose	Tight	Loose	Tight
S_G	24.48	23.19	45.78	43.67
S_B	26.14	22.68	46.66	40.10

Table 6: BLEU Scores with Loose vs. Tight Phrases

Table 6 presents the BLEU scores for S_G and S_B using two different phrase extraction techniques on English-Chinese and English-Arabic. In both languages, S_B outperforms S_G significantly when loose phrases are used. However, when we use only tight phrases, the performance of S_B gets significantly worse (3.5 to 6.5 BLEU-score reduction in comparison to loose phrases). The performance of S_G also gets worse but the degree of BLEU-score reduction is less than that of S_B. Overall S_G performs better than S_B with tight phrases; for English-Arabic, the difference between the two systems is more than 3 BLEU points. Note that, as before, the relation between the alignment measures and the BLEU scores varies, this time depending on whether loose phrases or tight phrases are used: both CPER and AER track the BLEU rankings for loose (but not for tight) phrases.

This suggests that changing alignment-related components of the system (i.e., phrase extraction and phrase scoring) influences the overall translation quality significantly for a particular alignment. Therefore, when comparing two alignments in the context of a MT system, it is important to take the alignment characteristics into account. For instance, alignments with many unaligned words are severely penalized when using tight phrases.

4.3 Untranslated Words

We analyzed the percentage of words left untranslated during decoding. Figure 2 shows the percentage of untranslated words in the FL using the Chinese and Arabic NIST MTEval’2003 test sets.

On English-Chinese data (using all four settings given in Table 4) S_U and S_G yield the highest percentage of untranslated words while S_I produces the lowest percentage of untranslated words. S_A and S_B leave about 2% of the FL words phrases

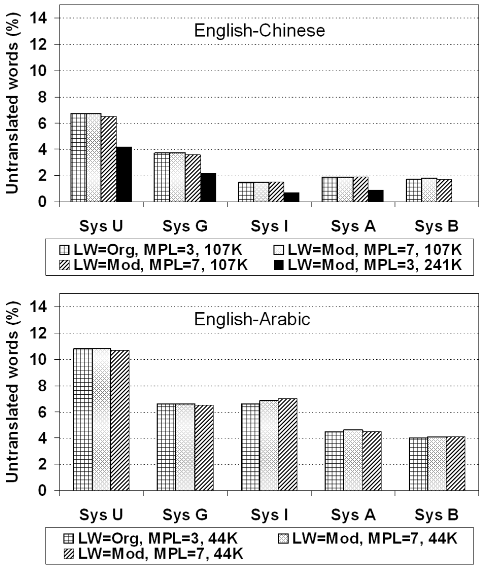


Figure 2: Percentage of untranslated words out of the total number of FL words

without translating them. Increasing the training data size reduces the percentage of untranslated words by nearly half with all five alignments. No significant impact on untranslated words is observed from modifying the lexical weights and changing the phrase length.

On English-Arabic data, all alignments result in higher percentages of untranslated words than English-Chinese, most likely due to data sparsity. As in Chinese-to-English translation, S_U is the worst and S_B is the best. S_I behaves quite differently, leaving nearly 7% of the words untranslated—an indicator of why it produces a higher BLEU score on Chinese but a lower score on Arabic compared to other alignments.

4.4 Analysis of Phrase Tables

This section presents several experiments to analyze how different alignments affect the size of the generated phrase tables, the distribution of the phrases that are used in decoding, and the coverage of the test set with the generated phrase tables.

	Chinese		Arabic	
Alignment	**MPL=3**	**MPL=7**	**MPL=3**	**MPL=7**
S_U	106	122	32	38
S_G	161	181	48	55
S_I	1331	3498	377	984
S_A	954	1856	297	594
S_B	876	1624	262	486

Table 7: Number of Phrases in the Phrase Table Filtered for MTEval'2003 Test Sets (in thousands)

Size of Phrase Tables The major impact of using different alignments in a phrase-based MT system is that each one results in a different phrase table. Table 7 presents the number of phrases that are extracted from five alignments using two different maximum phrase lengths (3 vs. 7) in two languages, after filtering the phrase table for MTEval'2003 test set. The size of the phrase table increases dramatically as the number of links in the initial alignment gets smaller. As a result, for both languages, S_U and S_G yield a much smaller phrase table than the other three alignments. As the maximum phrase length increases, the size of the phrase table gets bigger for all alignments; however, the growth of the table is more significant for precision-oriented alignments due to the high number of unaligned words.

Distribution of Phrases To investigate how the decoder chooses phrases of different lengths, we analyzed the distribution of the phrases in the filtered phrase table and the phrases that were used to decode Chinese MTEval'2003 test set.[5] For the remaining experiments in the paper, we use modified lexical weighting, a maximum phrase length of 7, and 107K sentence pairs for training.

The top row in Figure 3 shows the distribution of the phrases generated by the five alignments (using a maximum phrase length of 7) according to their length. The "j-i" designators correspond to the phrase pairs with j FL words and i English words. For S_U and S_G, the majority of the phrases contain only one FL word, and the percentage of the phrases with more than 2 FL words is less than 18%. For the other three alignments, however, the distribution of the phrases is almost inverted. For S_I, nearly 62% of the phrases contain more than 3 words on either FL or English side; for S_A and S_B, this percentage is around 45-50%.

Given the completely different phrase distribution, the most obvious question is whether the longer phrases generated by S_I, S_A and S_B are actually used in decoding. In order to investigate this, we did an analysis of the phrases used to decode the same test set.

The bottom row of Figure 3 shows the percentage of phrases used to decode the Chinese MTEval'2003 test set. The distribution of the actual phrases used in decoding is completely the reverse of the distribution of the phrases in the entire filtered table. For all five alignments, the majority of the used phrases is one-to-one (between

[5] Due to lack of space, we will present results on Chinese-English only in the rest of this paper but the Arabic-English results show the same trends.

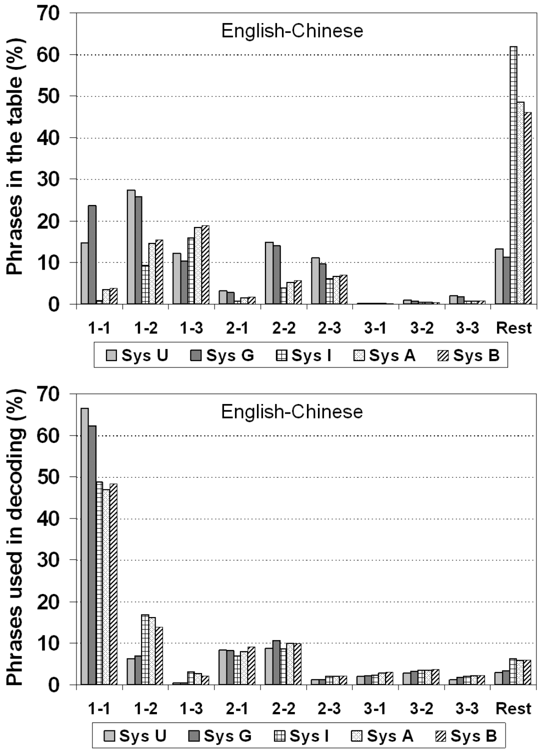


Figure 3: Distribution of the phrases in the phrase table filtered for Chinese MTEval'2003 test set (top row) and the phrases used in decoding the same test set (bottom row) according to their lengths

50-65% of the total number of phrases used in decoding). $\mathrm{S_I}$, $\mathrm{S_A}$ and $\mathrm{S_B}$ use the other phrase pairs (particularly 1-to-2 phrases) more than $\mathrm{S_U}$ and $\mathrm{S_G}$.

Note that $\mathrm{S_I}$, $\mathrm{S_A}$ and $\mathrm{S_B}$ use only a small portion of the phrases with more than 3 words although the majority of the phrase table contains phrases with more than 3 words on one side. It is surprising that the inclusion of phrase pairs with more than 3 words in the search space increases the BLEU score although the majority of the phrases used in decoding is mostly one-to-one.

Length of the Phrases used in Decoding We also investigated the number and length of phrases that are used to decode the given test set for different alignments. Table 8 presents the average number of English and FL words in the phrases used in decoding Chinese MTEval'2003 test set. The decoder uses fewer phrases with $\mathrm{S_I}$, $\mathrm{S_A}$ and $\mathrm{S_B}$ than for the other two, thus yielding a higher number of FL words per phrase. The number of English words per phrase is also higher for these three systems than the other two.

Alignment	|Eng|	|FL|
$\mathrm{S_U}$	1.39	1.28
$\mathrm{S_G}$	1.45	1.33
$\mathrm{S_I}$	1.51	1.55
$\mathrm{S_A}$	1.54	1.55
$\mathrm{S_B}$	1.56	1.52

Table 8: The average length of the phrases that are used in decoding Chinese MTEval'2003 test set

Coverage of the Test Set Finally, we examine the coverage of a test set using phrases of a specific length in the phrase table. Table 9 presents the coverage of the Chinese MTEval'2003 test set (source side) using only phrases of a particular length (from 1 to 7). For this experiment, we assume that a word in the test set is covered if it is part of a phrase pair that exists in the phrase table (if a word is part of multiple phrases, it is counted only once). Not surprisingly, using only phrases with one FL word, more than 90% of the test set can be covered for all 5 alignments. As the length of the phrases increases, the coverage of the test set decreases. For instance, using phrases with 5 FL words results in less than 5% coverage of the test set.

	Phrase Length (FL)						
A	1	2	3	4	5	6	7
$\mathrm{S_U}$	92.2	59.5	21.4	6.7	1.3	0.4	0.1
$\mathrm{S_G}$	95.5	64.4	24.9	7.4	1.6	0.5	0.3
$\mathrm{S_I}$	97.8	75.8	38.0	13.8	4.6	1.9	1.2
$\mathrm{S_A}$	97.3	75.3	36.1	12.5	3.8	1.5	0.8
$\mathrm{S_B}$	97.5	74.8	35.7	12.4	4.2	1.8	0.9

Table 9: Coverage of Chinese MTEval'2003 Test Set Using Phrases with a Specific Length on FL side (in percentages)

Table 9 reveals that the coverage of the test set is higher for precision-oriented alignments than recall-oriented alignments for all different lengths of the phrases. For instance, $\mathrm{S_I}$, $\mathrm{S_A}$, and $\mathrm{S_B}$ cover nearly 75% of the corpus using only phrases with 2 FL words, and nearly 36% of the corpus using phrases with 3 FL words. This suggests that recall-oriented alignments fail to catch a significant number of phrases that would be useful to decode this test set, and precision-oriented alignments would yield potentially more useful phrases.

Since precision-oriented alignments make a higher number of longer phrases available to the decoder (based on the coverage of phrases presented in Table 9), they are used more during decoding. Consequently, the major difference between the alignments is the coverage of the phrases extracted from different alignments. The more the phrase table covers the test set, the more the longer phrases are used during decoding, and precision-oriented alignments are better at generating high-coverage phrases than recall-oriented alignments.

5 Conclusions and Future Work

This paper investigated how different alignments change the behavior of phrase-based MT. We showed that AER is a poor indicator of MT performance because it penalizes incorrect links less than is reflected in the corresponding phrase-based MT. During phrase-based MT, an incorrect alignment link might prevent extraction of several phrases, but the number of phrases affected by that link depends on the context.

We designed CPER, a new phrase-oriented metric that is more informative than AER when the alignments are used in a phrase-based MT system because it is an indicator of how the set of phrases differ from one alignment to the next according to a pre-specified maximum phrase length.

Even with refined evaluation metrics (including CPER), we found it difficult to assess the impact of alignment on MT performance because word alignment is not the only factor that affects the choice of the correct words (or phrases) during decoding. We empirically showed that different phrase extraction techniques result in better MT output for certain alignments but the MT performance gets worse for other alignments. Similarly, adjusting the scores assigned to the phrases makes a significant difference for certain alignments while it has no impact on some others. Consequently, when comparing two BLEU scores, it is difficult to determine whether the alignments are bad to start with or the set of extracted phrases is bad or the phrases extracted from the alignments are assigned bad scores. This suggests that finding a direct correlation between AER (or even CPER) and the automated MT metrics is infeasible.

We demonstrated that recall-oriented alignment methods yield smaller phrase tables and a higher number of untranslated words when compared to precision-oriented methods. We also showed that the phrases extracted from recall-oriented alignments cover a smaller portion of a given test set when compared to precision-oriented alignments. Finally, we showed that the decoder with recall-oriented alignments uses shorter phrases more frequently as a result of unavailability of longer phrases that are extracted.

Future work will involve an investigation into how the phrase extraction and scoring should be adjusted to take the nature of the alignment into account and how the phrase-table size might be reduced without sacrificing the MT output quality.

Acknowledgments This work has been supported, in part, under ONR MURI Contract FCPO.810548265 and the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-2-0001. We also thank Adam Lopez for his very helpful comments on earlier drafts of this paper.

References

Necip F. Ayan, Bonnie J. Dorr, and Christof Monz. 2005. Neuralign: Combining word alignments using neural networks. In *Proceedings of EMNLP'2005*, pages 65–72.

Stanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at ACL-2005*.

Peter F. Brown, Stephan A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Chris Callison-Burch, David Talbot, and Miles Osborne. 2004. Statistical machine translation with word- and sentence-aligned parallel corpora. In *Proceedings of ACL'2004*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL'2005*.

Cyril Goutte, Kenji Yamada, and Eric Gaussier. 2004. Aligning words using matrix factorisation. In *Proceedings of ACL'2004*, pages 502–509.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for arabic-english machine translation. In *Proceedings of EMNLP'2005*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL'2003*.

Philipp Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation. In *Proceedings of AMTA'2004*.

Daniel Marcu and William Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of EMNLP'2002*.

I. Dan Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.

Robert C. Moore. 2005. A discriminative framework for bilingual word alignment. In *Proceedings of EMNLP'2005*.

Franz J. Och and Hermann Ney. 2000. A comparison of alignment models for statistical machine translation. In *Proceedings of COLING'2000*.

Franz J. Och. 2000b. GIZA++: Training of statistical translation models. Technical report, RWTH Aachen, University of Technology.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):9–51, March.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL'2003*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL'2002*.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 2005. A discriminative matching approach to word alignment. In *Proceedings of EMNLP'2005*.

Stefan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of COLING'1996*, pages 836–841.

Unsupervised Topic Modelling for Multi-Party Spoken Discourse

Matthew Purver
CSLI
Stanford University
Stanford, CA 94305, USA
mpurver@stanford.edu

Konrad P. Körding
Dept. of Brain & Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
kording@mit.edu

Thomas L. Griffiths
Dept. of Cognitive & Linguistic Sciences
Brown University
Providence, RI 02912, USA
tom_griffiths@brown.edu

Joshua B. Tenenbaum
Dept. of Brain & Cognitive Sciences
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
jbt@mit.edu

Abstract

We present a method for unsupervised topic modelling which adapts methods used in document classification (Blei et al., 2003; Griffiths and Steyvers, 2004) to unsegmented multi-party discourse transcripts. We show how Bayesian inference in this generative model can be used to simultaneously address the problems of topic *segmentation* and topic *identification*: automatically segmenting multi-party meetings into topically coherent segments with performance which compares well with previous unsupervised segmentation-only methods (Galley et al., 2003) while simultaneously extracting topics which rate highly when assessed for coherence by human judges. We also show that this method appears robust in the face of off-topic dialogue and speech recognition errors.

1 Introduction

Topic *segmentation* – division of a text or discourse into topically coherent segments – and topic *identification* – classification of those segments by subject matter – are joint problems. Both are necessary steps in automatic indexing, retrieval and summarization from large datasets, whether spoken or written. Both have received significant attention in the past (see Section 2), but most approaches have been targeted at either text or monologue, and most address only one of the two issues (usually for the very good reason that the dataset itself provides the other, for example by the explicit separation of individual documents or news stories in a collection). Spoken multi-party meetings pose a difficult problem: firstly, neither the segmentation nor the discussed topics can be taken as given; secondly, the discourse is by nature less tidily structured and less restricted in domain; and thirdly, speech recognition results have unavoidably high levels of error due to the noisy multi-speaker environment.

In this paper we present a method for unsupervised topic modelling which allows us to approach both problems simultaneously, inferring a set of topics while providing a segmentation into topically coherent segments. We show that this model can address these problems over multi-party discourse transcripts, providing good segmentation performance on a corpus of meetings (comparable to the best previous unsupervised method that we are aware of (Galley et al., 2003)), while also inferring a set of topics rated as semantically coherent by human judges. We then show that its segmentation performance appears relatively robust to speech recognition errors, giving us confidence that it can be successfully applied in a real speech-processing system.

The plan of the paper is as follows. Section 2 below briefly discusses previous approaches to the identification and segmentation problems. Section 3 then describes the model we use here. Section 4 then details our experiments and results, and conclusions are drawn in Section 5.

2 Background and Related Work

In this paper we are interested in spoken discourse, and in particular multi-party human-human meetings. Our overall aim is to produce information which can be used to summarize, browse and/or retrieve the information contained in meetings. User studies (Lisowska et al., 2004; Banerjee et al., 2005) have shown that topic information is important here: people are likely to want to know

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 17–24,
Sydney, July 2006. ©2006 Association for Computational Linguistics

which topics were discussed in a particular meeting, as well as have access to the discussion on particular topics in which they are interested. Of course, this requires both identification of the topics discussed, and segmentation into the periods of topically related discussion.

Work on automatic topic segmentation of *text* and *monologue* has been prolific, with a variety of approaches used. (Hearst, 1994) uses a measure of lexical cohesion between adjoining paragraphs in text; (Reynar, 1999) and (Beeferman et al., 1999) combine a variety of features such as statistical language modelling, cue phrases, discourse information and the presence of pronouns or named entities to segment broadcast news; (Maskey and Hirschberg, 2003) use entirely non-lexical features. Recent advances have used generative models, allowing lexical models of the topics themselves to be built while segmenting (Imai et al., 1997; Barzilay and Lee, 2004), and we take a similar approach here, although with some important differences detailed below.

Turning to *multi-party* discourse and *meetings*, however, most previous work on automatic segmentation (Reiter and Rigoll, 2004; Dielmann and Renals, 2004; Banerjee and Rudnicky, 2004), treats segments as representing meeting phases or events which characterize the *type* or *style* of discourse taking place (presentation, briefing, discussion etc.), rather than the topic or subject matter. While we expect some correlation between these two types of segmentation, they are clearly different problems. However, one comparable study is described in (Galley et al., 2003). Here, a lexical cohesion approach was used to develop an essentially unsupervised segmentation tool (*LCSeg*) which was applied to both text and meeting transcripts, giving performance better than that achieved by applying text/monologue-based techniques (see Section 4 below), and we take this as our benchmark for the segmentation problem. Note that they improved their accuracy by combining the unsupervised output with discourse features in a supervised classifier – while we do not attempt a similar comparison here, we expect a similar technique would yield similar segmentation improvements.

In contrast, we take a generative approach, modelling the text as being generated by a sequence of mixtures of underlying topics. The approach is unsupervised, allowing both segmentation and topic extraction from unlabelled data.

3 Learning topics and segments

We specify our model to address the problem of topic segmentation: attempting to break the discourse into discrete segments in which a particular set of topics are discussed. Assume we have a corpus of U utterances, ordered in sequence. The uth utterance consists of N_u words, chosen from a vocabulary of size W. The set of words associated with the uth utterance are denoted $\mathbf{w}_u$, and indexed as $w_{u,i}$. The entire corpus is represented by $\mathbf{w}$.

Following previous work on probabilistic topic models (Hofmann, 1999; Blei et al., 2003; Griffiths and Steyvers, 2004), we model each utterance as being generated from a particular distribution over topics, where each topic is a probability distribution over words. The utterances are ordered sequentially, and we assume a Markov structure on the distribution over topics: with high probability, the distribution for utterance u is the same as for utterance $u-1$; otherwise, we sample a new distribution over topics. This pattern of dependency is produced by associating a binary switching variable with each utterance, indicating whether its topic is the same as that of the previous utterance. The joint states of all the switching variables define segments that should be semantically coherent, because their words are generated by the same topic vector. We will first describe this generative model in more detail, and then discuss inference in this model.

3.1 A hierarchical Bayesian model

We are interested in where changes occur in the set of topics discussed in these utterances. To this end, let c_u indicate whether a change in the distribution over topics occurs at the uth utterance and let $P(c_u = 1) = \pi$ (where π thus defines the expected number of segments). The distribution over topics associated with the uth utterance will be denoted $\theta^{(u)}$, and is a multinomial distribution over T topics, with the probability of topic t being $\theta_t^{(u)}$. If $c_u = 0$, then $\theta^{(u)} = \theta^{(u-1)}$. Otherwise, $\theta^{(u)}$ is drawn from a symmetric Dirichlet distribution with parameter α. The distribution is thus:

$$P(\theta^{(u)}|c_u, \theta^{(u-1)}) = \begin{cases} \delta(\theta^{(u)}, \theta^{(u-1)}) & c_u = 0 \\ \frac{\Gamma(T\alpha)}{\Gamma(\alpha)^T} \prod_{t=1}^{T} (\theta_t^{(u)})^{\alpha-1} & c_u = 1 \end{cases}$$

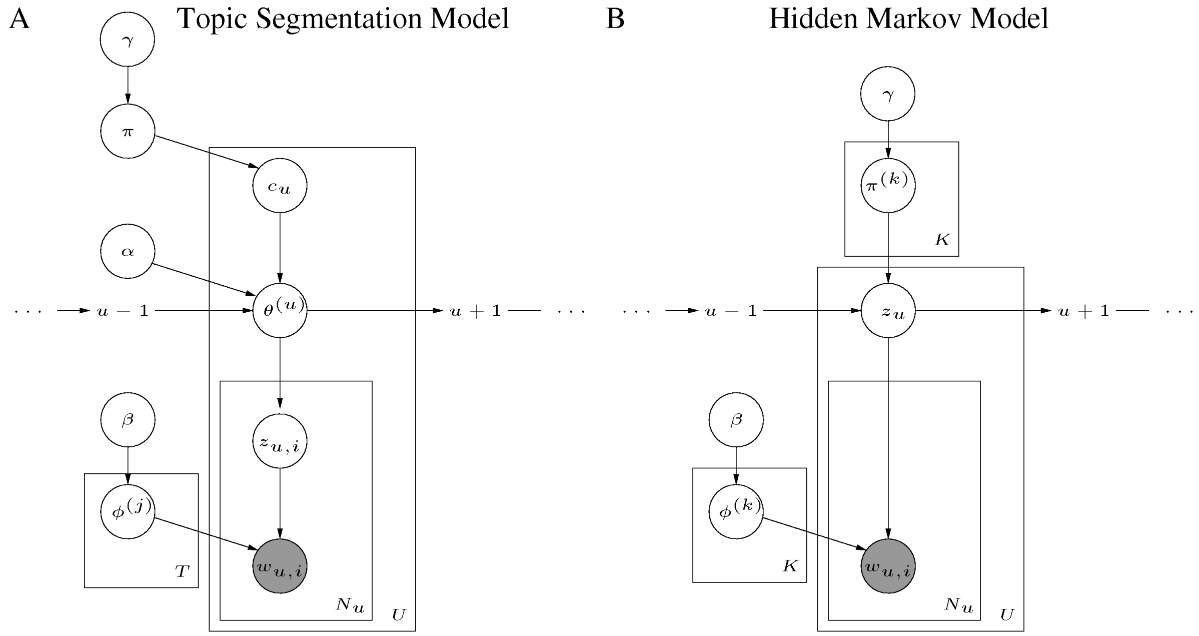


Figure 1: Graphical models indicating the dependencies among variables in (a) the topic segmentation model and (b) the hidden Markov model used as a comparison.

where $\delta(\cdot,\cdot)$ is the Dirac delta function, and $\Gamma(\cdot)$ is the generalized factorial function. This distribution is not well-defined when $u = 1$, so we set $c_1 = 1$ and draw $\theta^{(1)}$ from a symmetric Dirichlet(α) distribution accordingly.

As in (Hofmann, 1999; Blei et al., 2003; Griffiths and Steyvers, 2004), each topic T_j is a multinomial distribution $\phi^{(j)}$ over words, and the probability of the word w under that topic is $\phi^{(j)}_w$. The uth utterance is generated by sampling a topic assignment $z_{u,i}$ for each word i in that utterance with $P(z_{u,i} = t|\theta^{(u)}) = \theta^{(u)}_t$, and then sampling a word $w_{u,i}$ from $\phi^{(j)}$, with $P(w_{u,i} = w|z_{u,i} = j, \phi^{(j)}) = \phi^{(j)}_w$. If we assume that π is generated from a symmetric Beta(γ) distribution, and each $\phi^{(j)}$ is generated from a symmetric Dirichlet(β) distribution, we obtain a joint distribution over all of these variables with the dependency structure shown in Figure 1A.

3.2 Inference

Assessing the posterior probability distribution over topic changes $\mathbf{c}$ given a corpus $\mathbf{w}$ can be simplified by integrating out the parameters θ, ϕ, and π. According to Bayes rule we have:

$$P(\mathbf{z}, \mathbf{c}|\mathbf{w}) = \frac{P(\mathbf{w}|\mathbf{z})P(\mathbf{z}|\mathbf{c})P(\mathbf{c})}{\sum_{\mathbf{z},\mathbf{c}} P(\mathbf{w}|\mathbf{z})P(\mathbf{z}|\mathbf{c})P(\mathbf{c})} \tag{1}$$

Evaluating $P(\mathbf{c})$ requires integrating over π. Specifically, we have:

$$\begin{aligned} P(\mathbf{c}) &= \textstyle\int_0^1 P(\mathbf{c}|\pi)P(\pi)\, d\pi \\ &= \tfrac{\Gamma(2\gamma)}{\Gamma(\gamma)^2} \tfrac{\Gamma(n_1+\gamma)\Gamma(n_0+\gamma)}{\Gamma(N+2\gamma)} \end{aligned} \tag{2}$$

where n_1 is the number of utterances for which $c_u = 1$, and n_0 is the number of utterances for which $c_u = 0$. Computing $P(\mathbf{w}|\mathbf{z})$ proceeds along similar lines:

$$\begin{aligned} P(\mathbf{w}|\mathbf{z}) &= \textstyle\int_{\Delta_W^T} P(\mathbf{w}|\mathbf{z}, \phi)P(\phi)\, d\phi \\ &= \left(\tfrac{\Gamma(W\beta)}{\Gamma(\beta)^W}\right)^T \textstyle\prod_{t=1}^T \tfrac{\prod_{w=1}^W \Gamma(n_w^{(t)}+\beta)}{\Gamma(n_\cdot^{(t)}+W\beta)} \end{aligned} \tag{3}$$

where Δ_W^T is the T-dimensional cross-product of the multinomial simplex on W points, $n_w^{(t)}$ is the number of times word w is assigned to topic t in $\mathbf{z}$, and $n_\cdot^{(t)}$ is the total number of words assigned to topic t in $\mathbf{z}$. To evaluate $P(\mathbf{z}|\mathbf{c})$ we have:

$$P(\mathbf{z}|\mathbf{c}) = \int_{\Delta_T^U} P(\mathbf{z}|\theta)P(\theta|\mathbf{c})\, d\theta \tag{4}$$

The fact that the c_u variables effectively divide the sequence of utterances into segments that use the same distribution over topics simplifies solving the integral and we obtain:

$$P(\mathbf{z}|\mathbf{c}) = \left(\frac{\Gamma(T\alpha)}{\Gamma(\alpha)^T}\right)^{n_1} \prod_{u \in \mathcal{U}_1} \frac{\prod_{t=1}^T \Gamma(n_t^{(S_u)} + \alpha)}{\Gamma(n_\cdot^{(S_u)} + T\alpha)}. \tag{5}$$

$$P(c_u|\mathbf{c}_{-u},\mathbf{z},\mathbf{w}) \propto \begin{cases} \frac{\prod_{t=1}^{T}\Gamma(n_t^{(\mathcal{S}_u^0)}+\alpha)}{\Gamma(n_\cdot^{(\mathcal{S}_u^0)}+T\alpha)} & \frac{n_0+\gamma}{N+2\gamma} \quad c_u = 0 \\ \frac{\Gamma(T\alpha)}{\Gamma(\alpha)^T} \frac{\prod_{t=1}^{T}\Gamma(n_t^{(\mathcal{S}_{u-1}^1)}+\alpha)}{\Gamma(n_\cdot^{(\mathcal{S}_{u-1}^1)}+T\alpha)} \frac{\prod_{t=1}^{T}\Gamma(n_t^{(\mathcal{S}_u^1)}+\alpha)}{\Gamma(n_\cdot^{(\mathcal{S}_u^1)}+T\alpha)} & \frac{n_1+\gamma}{N+2\gamma} \quad c_u = 1 \end{cases} \tag{7}$$

where $\mathcal{U}_1 = \{u|c_u = 1\}$, $\mathcal{U}_0 = \{u|c_u = 0\}$, $\mathcal{S}_u$ denotes the set of utterances that share the same topic distribution (i.e. belong to the same segment) as u, and $n_t^{(\mathcal{S}_u)}$ is the number of times topic t appears in the segment $\mathcal{S}_u$ (i.e. in the values of $z_{u'}$ corresponding for $u' \in \mathcal{S}_u$).

Equations 2, 3, and 5 allow us to evaluate the numerator of the expression in Equation 1. However, computing the denominator is intractable. Consequently, we sample from the posterior distribution $P(\mathbf{z}, \mathbf{c}|\mathbf{w})$ using Markov chain Monte Carlo (MCMC) (Gilks et al., 1996). We use Gibbs sampling, drawing the topic assignment for each word, $z_{u,i}$, conditioned on all other topic assignments, $\mathbf{z}_{-(u,i)}$, all topic change indicators, $\mathbf{c}$, and all words, $\mathbf{w}$; and then drawing the topic change indicator for each utterance, c_u, conditioned on all other topic change indicators, $\mathbf{c}_{-u}$, all topic assignments $\mathbf{z}$, and all words $\mathbf{w}$.

The conditional probabilities we need can be derived directly from Equations 2, 3, and 5. The conditional probability of $z_{u,i}$ indicates the probability that $w_{u,i}$ should be assigned to a particular topic, given other assignments, the current segmentation, and the words in the utterances. Cancelling constant terms, we obtain:

$$P(z_{u,i}|\mathbf{z}_{-(u,i)},\mathbf{c},\mathbf{w}) = \frac{n_{w_{u,i}}^{(t)} + \beta}{n_\cdot^{(t)} + W\beta} \frac{n_{z_{u,i}}^{(\mathcal{S}_u)} + \alpha}{n_\cdot^{(\mathcal{S}_u)} + T\alpha}. \tag{6}$$

where all counts (i.e. the n terms) exclude $z_{u,i}$. The conditional probability of c_u indicates the probability that a new segment should start at u. In sampling c_u from this distribution, we are splitting or merging segments. Similarly we obtain the expression in (7), where $\mathcal{S}_u^1$ is $\mathcal{S}_u$ for the segmentation when $c_u = 1$, $\mathcal{S}_u^0$ is $\mathcal{S}_u$ for the segmentation when $c_u = 0$, and all counts (e.g. n_1) exclude c_u. For this paper, we fixed α, β and γ at 0.01.

Our algorithm is related to (Barzilay and Lee, 2004)'s approach to text segmentation, which uses a hidden Markov model (HMM) to model segmentation and topic inference for text using a bigram representation in restricted domains. Due to the adaptive combination of different topics our algorithm can be expected to generalize well to larger domains. It also relates to earlier work by (Blei and Moreno, 2001) that uses a topic representation but also does not allow adaptively combining different topics. However, while HMM approaches allow a segmentation of the data by topic, they do not allow adaptively combining different topics into segments: while a new segment can be modelled as being identical to a topic that has already been observed, it can not be modelled as a combination of the previously observed topics.[1] Note that while (Imai et al., 1997)'s HMM approach allows topic mixtures, it requires supervision with hand-labelled topics.

In our experiments we therefore compared our results with those obtained by a similar but simpler 10 state HMM, using a similar Gibbs sampling algorithm. The key difference between the two models is shown in Figure 1. In the HMM, all variation in the content of utterances is modelled at a single level, with each segment having a distribution over words corresponding to a single state. The hierarchical structure of our topic segmentation model allows variation in content to be expressed at two levels, with each segment being produced from a linear combination of the distributions associated with each topic. Consequently, our model can often capture the content of a sequence of words by postulating a single segment with a novel distribution over topics, while the HMM has to frequently switch between states.

4 Experiments

4.1 Experiment 0: Simulated data

To analyze the properties of this algorithm we first applied it to a simulated dataset: a sequence of 10,000 words chosen from a vocabulary of 25. Each segment of 100 successive words had a con-

[1] Say that a particular corpus leads us to infer topics corresponding to "speech recognition" and "discourse understanding". A single discussion concerning *speech recognition for discourse understanding* could be modelled by our algorithm as a single segment with a suitable weighted mixture of the two topics; a HMM approach would tend to split it into multiple segments (or require a specific topic for this segment).

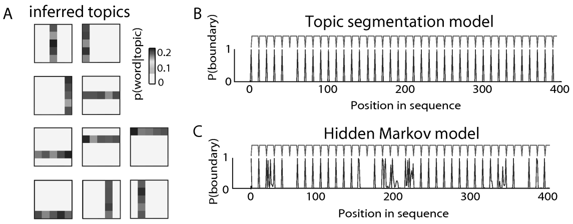


Figure 2: Simulated data: A) inferred topics; B) segmentation probabilities; C) HMM version.

stant topic distribution (with distributions for different segments drawn from a Dirichlet distribution with $\beta = 0.1$), and each subsequence of 10 words was taken to be one utterance. The topic-word assignments were chosen such that when the vocabulary is aligned in a 5×5 grid the topics were binary bars. The inference algorithm was then run for 200,000 iterations, with samples collected after every 1,000 iterations to minimize autocorrelation. Figure 2 shows the inferred topic-word distributions and segment boundaries, which correspond well with those used to generate the data.

4.2 Experiment 1: The ICSI corpus

We applied the algorithm to the ICSI meeting corpus transcripts (Janin et al., 2003), consisting of manual transcriptions of 75 meetings. For evaluation, we use (Galley et al., 2003)'s set of human-annotated segmentations, which covers a sub-portion of 25 meetings and takes a relatively coarse-grained approach to topic with an average of 5-6 topic segments per meeting. Note that these segmentations were not used in training the model: topic inference and segmentation was unsupervised, with the human annotations used only to provide some knowledge of the overall segmentation density and to evaluate performance.

The transcripts from all 75 meetings were linearized by utterance start time and merged into a single dataset that contained 607,263 word tokens. We sampled for 200,000 iterations of MCMC, taking samples every 1,000 iterations, and then averaged the sampled c_u variables over the last 100 samples to derive an estimate for the posterior probability of a segmentation boundary at each utterance start. This probability was then thresholded to derive a final segmentation which was compared to the manual annotations. More precisely, we apply a small amount of smoothing (Gaussian kernel convolution) and take the mid-points of any areas above a set threshold to be the segment boundaries. Varying this threshold allows us to segment the discourse in a more or less fine-grained way (and we anticipate that this could be user-settable in a meeting browsing application). If the correct number of segments is known for a meeting, this can be used directly to determine the optimum threshold, increasing performance; if not, we must set it at a level which corresponds to the desired general level of granularity. For each set of annotations, we therefore performed two sets of segmentations: one in which the threshold was set for each meeting to give the known gold-standard number of segments, and one in which the threshold was set on a separate development set to give the overall corpus-wide average number of segments, and held constant for all test meetings.[2] This also allows us to compare our results with those of (Galley et al., 2003), who apply a similar threshold to their lexical cohesion function and give corresponding results produced with known/unknown numbers of segments.

Segmentation We assessed segmentation performance using the P_k and *WindowDiff* (W_D) error measures proposed by (Beeferman et al., 1999) and (Pevzner and Hearst, 2002) respectively; both intuitively provide a measure of the probability that two points drawn from the meeting will be *in*correctly separated by a hypothesized segment boundary – thus, lower P_k and W_D figures indicate better agreement with the human-annotated results.[3] For the numbers of segments we are dealing with, a baseline of segmenting the discourse into equal-length segments gives both P_k and W_D about 50%. In order to investigate the effect of the number of underlying topics T, we tested models using 2, 5, 10 and 20 topics. We then compared performance with (Galley et al., 2003)'s *LCSeg* tool, and with a 10-state HMM model as described above. Results are shown in Table 1, averaged over the 25 test meetings.

Results show that our model significantly outperforms the HMM equivalent – because the HMM cannot combine different topics, it places a lot of segmentation boundaries, resulting in inferior performance. Using stemming and a bigram

[2] The development set was formed from the other meetings in the same ICSI subject areas as the annotated test meetings.

[3] W_D takes into account the likely number of incorrectly separating hypothesized boundaries; P_k only a binary correct/incorrect classification.

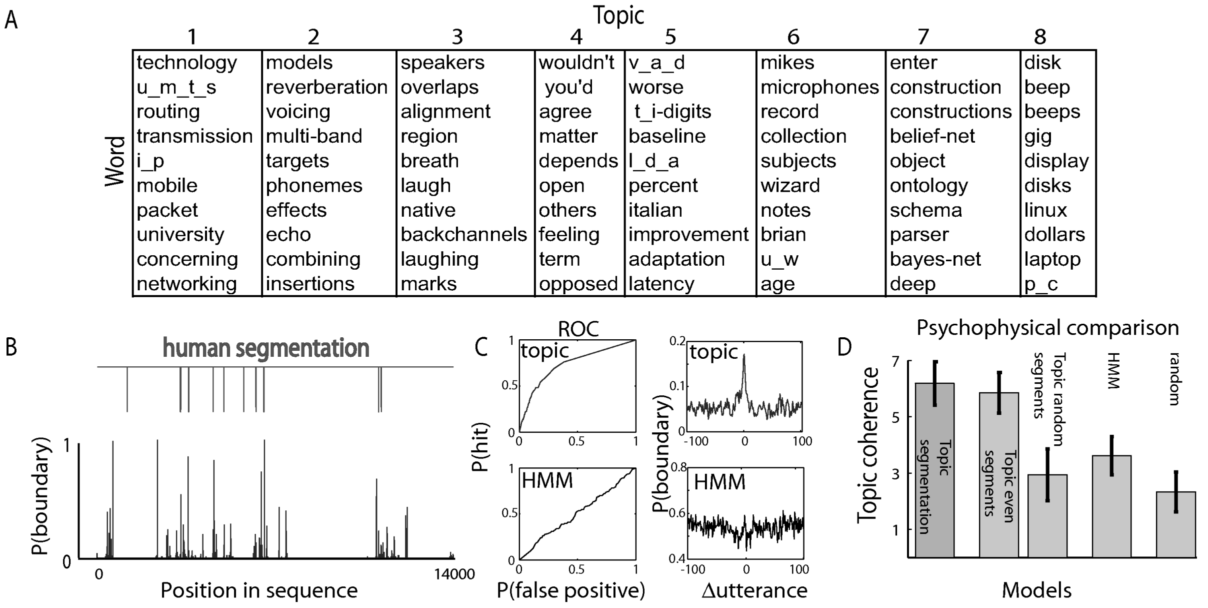


Figure 3: Results from the ICSI corpus: A) the words most indicative for each topic; B) Probability of a segment boundary, compared with human segmentation, for an arbitrary subset of the data; C) Receiver-operator characteristic (ROC) curves for predicting human segmentation, and conditional probabilities of placing a boundary at an offset from a human boundary; D) subjective topic coherence ratings.

Model	Number of topics T 2	5	10	20	HMM	*LCSeg*
P_k	.284	.297	.329	.290	.375	.319

	known		unknown	
Model	P_k	W_D	P_k	W_D
$T = 10$	.289	.329	.329	.353
LCSeg	.264	.294	.319	.359

Table 1: Results on the ICSI meeting corpus.

representation, however, might improve its performance (Barzilay and Lee, 2004), although similar benefits might equally apply to our model. It also performs comparably to (Galley et al., 2003)'s unsupervised performance (exceeding it for some settings of T). It does not perform as well as their hybrid supervised system, which combined *LCSeg* with supervised learning over discourse features ($P_k = .23$); but we expect that a similar approach would be possible here, combining our segmentation probabilities with other discourse-based features in a supervised way for improved performance. Interestingly, segmentation quality, at least at this relatively coarse-grained level, seems hardly affected by the overall number of topics T.

Figure 3B shows an example for one meeting of how the inferred topic segmentation probabilities at each utterance compare with the gold-standard segment boundaries. Figure 3C illustrates the performance difference between our model and the HMM equivalent at an example segment boundary: for this example, the HMM model gives almost no discrimination.

Identification Figure 3A shows the most indicative words for a subset of the topics inferred at the last iteration. Encouragingly, most topics seem intuitively to reflect the subjects we know were discussed in the ICSI meetings – the majority of them (67 meetings) are taken from the weekly meetings of 3 distinct research groups, where discussions centered around speech recognition techniques (topics 2, 5), meeting recording, annotation and hardware setup (topics 6, 3, 1, 8), robust language processing (topic 7). Others reflect general classes of words which are independent of subject matter (topic 4).

To compare the quality of these inferred topics we performed an experiment in which 7 human observers rated (on a scale of 1 to 9) the semantic coherence of 50 lists of 10 words each. Of these lists, 40 contained the most indicative words for each of the 10 topics from different models: the topic segmentation model; a topic model that had the same number of segments but with fixed evenly spread segmentation boundaries; an equivalent with randomly placed segmentation boundaries; and the HMM. The other 10 lists contained random samples of 10 words from the other 40 lists. Results are shown in Figure 3D, with the topic segmentation model producing the most coherent topics and the HMM model and random words scoring less well. Interestingly, using an even distribution of boundaries but allowing the topic model to infer topics performs similarly well with even segmentation, but badly with random segmentation – topic quality is thus not very susceptible to the precise segmentation of the text, but does require some reasonable approximation (on ICSI data, an even segmentation gives a P_k of about 50%, while random segmentations can do much worse). However, note that the full topic segmentation model is able to identify meaningful segmentation boundaries *at the same time* as inferring topics.

4.3 Experiment 2: Dialogue robustness

Meetings often include off-topic dialogue, in particular at the beginning and end, where informal chat and meta-dialogue are common. Galley et al. (2003) annotated these sections explicitly, together with the ICSI "digit-task" sections (participants read sequences of digits to provide data for speech recognition experiments), and removed them from their data, as did we in Experiment 1 above. While this seems reasonable for the purposes of investigating ideal algorithm performance, in real situations we will be faced with such off-topic dialogue, and would obviously prefer segmentation performance not to be badly affected (and ideally, enabling segmentation of the off-topic sections from the meeting proper). One might suspect that an unsupervised generative model such as ours might not be robust in the presence of numerous off-topic words, as spurious topics might be inferred and used in the mixture model throughout. In order to investigate this, we therefore also tested on the full dataset without removing these sections (806,026 word tokens in total), and added the section boundaries as further desired gold-standard segmentation boundaries. Table 2 shows the results: performance is not significantly affected, and again is very similar for both our model and *LCSeg*.

4.4 Experiment 3: Speech recognition

The experiments so far have all used manual word transcriptions. Of course, in real meeting processing systems, we will have to deal with speech recognition (ASR) errors. We therefore also tested on 1-best ASR output provided by ICSI, and results are shown in Table 2. The "off-topic" and "digits" sections were removed in this test, so results are comparable with Experiment 1. Segmentation accuracy seems extremely robust; interestingly, *LCSeg*'s results are less robust (the drop in performance is higher), especially when the number of segments in a meeting is unknown.

		known		unknown	
Experiment	Model	P_k	W_D	P_k	W_D
2	$T = 10$	.296	.342	.325	.366
(off-topic data)	*LCSeg*	.307	.338	.322	.386
3	$T = 10$	.266	.306	.291	.331
(ASR data)	*LCSeg*	.289	.339	.378	.472

Table 2: Results for Experiments 2 & 3: robustness to off-topic and ASR data.

It is surprising to notice that the segmentation accuracy in this experiment was actually slightly higher than achieved in Experiment 1 (especially given that ASR word error rates were generally above 20%). This may simply be a smoothing effect: differences in vocabulary and its distribution can effectively change the prior towards sparsity instantiated in the Dirichlet distributions.

5 Summary and Future Work

We have presented an unsupervised generative model which allows topic segmentation and identification from unlabelled data. Performance on the ICSI corpus of multi-party meetings is comparable with the previous unsupervised segmentation results, and the extracted topics are rated well by human judges. Segmentation accuracy is robust in the face of noise, both in the form of off-topic discussion and speech recognition hypotheses.

Future Work Spoken discourse exhibits several features not derived from the words themselves but which seem intuitively useful for segmentation, e.g. speaker changes, speaker identities and roles, silences, overlaps, prosody and so on. As shown by (Galley et al., 2003), some of these features can be combined with lexical information to improve segmentation performance (although in a supervised manner), and (Maskey and Hirschberg, 2003) show some success in broadcast news segmentation using *only* these kinds of non-lexical features. We are currently investigating the addition of non-lexical features as observed outputs in our unsupervised generative model.

We are also investigating improvements into the lexical model as presented here, firstly via simple techniques such as word stemming and replacement of named entities by generic class tokens (Barzilay and Lee, 2004); but also via the use of multiple ASR hypotheses by incorporating word confusion networks into our model. We expect that this will allow improved segmentation and identification performance with ASR data.

Acknowledgements

This work was supported by the CALO project (DARPA grant NBCH-D-03-0010). We thank Elizabeth Shriberg and Andreas Stolcke for providing automatic speech recognition data for the ICSI corpus and for their helpful advice; John Niekrasz and Alex Gruenstein for help with the NOMOS corpus annotation tool; and Michel Galley for discussion of his approach and results.

References

Satanjeev Banerjee and Alex Rudnicky. 2004. Using simple speech-based features to detect the state of a meeting and the roles of the meeting participants. In *Proceedings of the 8th International Conference on Spoken Language Processing*.

Satanjeev Banerjee, Carolyn Rosé, and Alex Rudnicky. 2005. The necessity of a meeting recording and playback system, and the benefit of topic-level annotations to meeting browsing. In *Proceedings of the 10th International Conference on Human-Computer Interaction*.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL 2004: Proceedings of the Main Conference*, pages 113–120.

Doug Beeferman, Adam Berger, and John D. Lafferty. 1999. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177–210.

David Blei and Pedro Moreno. 2001. Topic segmentation with an aspect hidden Markov model. In *Proceedings of the 24th Annual International Conference on Research and Development in Information Retrieval*, pages 343–348.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Alfred Dielmann and Steve Renals. 2004. Dynamic Bayesian Networks for meeting structuring. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

Michel Galley, Kathleen McKeown, Eric Fosler-Lussier, and Hongyan Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 562–569.

W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors. 1996. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, Suffolk.

Thomas Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Science*, 101:5228–5235.

Marti A. Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proc. 32nd Meeting of the Association for Computational Linguistics*, Los Cruces, NM, June.

Thomas Hofmann. 1999. Probablistic latent semantic indexing. In *Proceedings of the 22nd Annual SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57.

Toru Imai, Richard Schwartz, Francis Kubala, and Long Nguyen. 1997. Improved topic discrimination of broadcast news using a model of multiple simultaneous topics. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 727–730.

Adam Janin, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Nelson Morgan, Barbara Peskin, Thilo Pfau, Elizabeth Shriberg, Andreas Stolcke, and Chuck Wooters. 2003. The ICSI Meeting Corpus. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 364–367.

Agnes Lisowska, Andrei Popescu-Belis, and Susan Armstrong. 2004. User query analysis for the specification and evaluation of a dialogue processing and retrieval system. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*.

Sameer R. Maskey and Julia Hirschberg. 2003. Automatic summarization of broadcast news using structural features. In *Eurospeech 2003*, Geneva, Switzerland.

Lev Pevzner and Marti Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

Stehpan Reiter and Gerhard Rigoll. 2004. Segmentation and classification of meeting events using multiple classifier fusion and dynamic programming. In *Proceedings of the International Conference on Pattern Recognition*.

Jeffrey Reynar. 1999. Statistical models for topic segmentation. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 357–364.

Minimum Cut Model for Spoken Lecture Segmentation

Igor Malioutov and **Regina Barzilay**
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
{igorm,regina}@csail.mit.edu

Abstract

We consider the task of unsupervised lecture segmentation. We formalize segmentation as a graph-partitioning task that optimizes the normalized cut criterion. Our approach moves beyond localized comparisons and takes into account long-range cohesion dependencies. Our results demonstrate that global analysis improves the segmentation accuracy and is robust in the presence of speech recognition errors.

1 Introduction

The development of computational models of text structure is a central concern in natural language processing. Text segmentation is an important instance of such work. The task is to partition a text into a linear sequence of topically coherent segments and thereby induce a content structure of the text. The applications of the derived representation are broad, encompassing information retrieval, question-answering and summarization.

Not surprisingly, text segmentation has been extensively investigated over the last decade. Following the first unsupervised segmentation approach by Hearst (1994), most algorithms assume that variations in lexical distribution indicate topic changes. When documents exhibit sharp variations in lexical distribution, these algorithms are likely to detect segment boundaries accurately. For example, most algorithms achieve high performance on synthetic collections, generated by concatenation of random text blocks (Choi, 2000). The difficulty arises, however, when transitions between topics are smooth and distributional variations are subtle. This is evident in the performance of existing unsupervised algorithms on less structured datasets, such as spoken meeting transcripts (Galley et al., 2003). Therefore, a more refined analysis of lexical distribution is needed.

Our work addresses this challenge by casting text segmentation in a graph-theoretic framework. We abstract a text into a weighted undirected graph, where the nodes of the graph correspond to sentences and edge weights represent the pairwise sentence similarity. In this framework, text segmentation corresponds to a graph partitioning that optimizes the *normalized-cut criterion* (Shi and Malik, 2000). This criterion measures both the similarity within each partition and the dissimilarity across different partitions. Thus, our approach moves beyond localized comparisons and takes into account long-range changes in lexical distribution. Our key hypothesis is that global analysis yields more accurate segmentation results than local models.

We tested our algorithm on a corpus of spoken lectures. Segmentation in this domain is challenging in several respects. Being less structured than written text, lecture material exhibits digressions, disfluencies, and other artifacts of spontaneous communication. In addition, the output of speech recognizers is fraught with high word error rates due to specialized technical vocabulary and lack of in-domain spoken data for training. Finally, pedagogical considerations call for fluent transitions between different topics in a lecture, further complicating the segmentation task.

Our experimental results confirm our hypothesis: considering long-distance lexical dependencies yields substantial gains in segmentation performance. Our graph-theoretic approach compares favorably to state-of-the-art segmentation algorithms and attains results close to the range of human agreement scores. Another attractive prop-

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 25–32,
Sydney, July 2006.

erty of the algorithm is its robustness to noise: the accuracy of our algorithm does not deteriorate significantly when applied to speech recognition output.

2 Previous Work

Most unsupervised algorithms assume that fragments of text with homogeneous lexical distribution correspond to topically coherent segments. Previous research has analyzed various facets of lexical distribution, including lexical weighting, similarity computation, and smoothing (Hearst, 1994; Utiyama and Isahara, 2001; Choi, 2000; Reynar, 1998; Kehagias et al., 2003; Ji and Zha, 2003).

The focus of our work, however, is on an orthogonal yet fundamental aspect of this analysis — the impact of long-range cohesion dependencies on segmentation performance. In contrast to previous approaches, the homogeneity of a segment is determined not only by the similarity of its words, but also by their relation to words in other segments of the text. We show that optimizing our global objective enables us to detect subtle topical changes.

Graph-Theoretic Approaches in Vision Segmentation Our work is inspired by minimum-cut-based segmentation algorithms developed for image analysis. Shi and Malik (2000) introduced the normalized-cut criterion and demonstrated its practical benefits for segmenting static images.

Our method, however, is not a simple application of the existing approach to a new task. First, in order to make it work in the new linguistic framework, we had to redefine the underlying representation and introduce a variety of smoothing and lexical weighting techniques. Second, the computational techniques for finding the optimal partitioning are also quite different. Since the minimization of the normalized cut is NP-complete in the general case, researchers in vision have to approximate this computation. Fortunately, we can find an exact solution due to the linearity constraint on text segmentation.

3 Minimum Cut Framework

Linguistic research has shown that word repetition in a particular section of a text is a device for creating thematic cohesion (Halliday and Hasan, 1976), and that changes in the lexical distributions usually signal topic transitions.

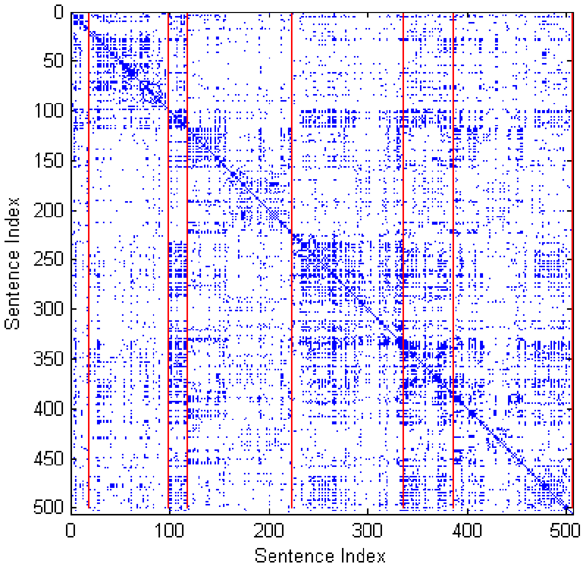


Figure 1: Sentence similarity plot for a Physics lecture, with vertical lines indicating true segment boundaries.

Figure 1 illustrates these properties in a lecture transcript from an undergraduate Physics class. We use the text Dotplotting representation by (Church, 1993) and plot the cosine similarity scores between every pair of sentences in the text. The intensity of a point (i, j) on the plot indicates the degree to which the i-th sentence in the text is similar to the j-th sentence. The true segment boundaries are denoted by vertical lines. This similarity plot reveals a block structure where true boundaries delimit blocks of text with high inter-sentential similarity. Sentences found in different blocks, on the other hand, tend to exhibit low similarity.

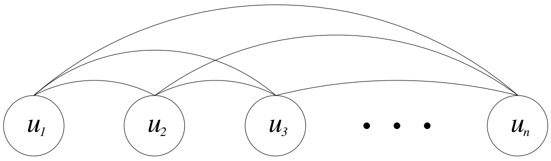


Figure 2: Graph-based Representation of Text

Formalizing the Objective Whereas previous unsupervised approaches to segmentation rested on intuitive notions of similarity density, we formalize the objective of text segmentation through cuts on graphs. We aim to jointly maximize the intra-segmental similarity and minimize the similarity between different segments. In other words, we want to find the segmentation with a maximally homogeneous set of segments that are also maximally different from each other.

Let $G = \{V, E\}$ be an undirected, weighted graph, where V is the set of nodes corresponding to sentences in the text and E is the set of weighted edges (See Figure 2). The edge weights, $w(u, v)$, define a measure of similarity between pairs of nodes u and v, where higher scores indicate higher similarity. Section 4 provides more details on graph construction.

We consider the problem of partitioning the graph into two disjoint sets of nodes A and B. We aim to minimize the cut, which is defined to be the sum of the crossing edges between the two sets of nodes. In other words, we want to split the sentences into two maximally dissimilar classes by choosing A and B to minimize:

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

However, we need to ensure that the two partitions are not only maximally different from each other, but also that they are themselves homogeneous by accounting for intra-partition node similarity. We formulate this requirement in the framework of normalized cuts (Shi and Malik, 2000), where the cut value is normalized by the volume of the corresponding partitions. The volume of the partition is the sum of its edges to the whole graph:

$$vol(A) = \sum_{u \in A, v \in V} w(u, v)$$

The normalized cut criterion ($Ncut$) is then defined as follows:

$$Ncut(A, B) = \frac{cut(A, B)}{vol(A)} + \frac{cut(A, B)}{vol(B)}$$

By minimizing this objective we simultaneously minimize the similarity across partitions and maximize the similarity within partitions. This formulation also allows us to decompose the objective into a sum of individual terms, and formulate a dynamic programming solution to the multiway cut problem.

This criterion is naturally extended to a k-way normalized cut:

$$Ncut_k(V) = \frac{cut(A_1, V - A_1)}{vol(A_1)} + \ldots + \frac{cut(A_k, V - A_k)}{vol(A_k)}$$

where $A_1 \ldots A_k$ form a partition of the graph, and $V - A_k$ is the set difference between the entire graph and partition k.

Decoding Papadimitriou proved that the problem of minimizing normalized cuts on graphs is NP-complete (Shi and Malik, 2000). However, in our case, the multi-way cut is constrained to preserve the linearity of the segmentation. By segmentation linearity, we mean that all of the nodes between the leftmost and the rightmost nodes of a particular partition have to belong to that partition. With this constraint, we formulate a dynamic programming algorithm for exactly finding the minimum normalized multiway cut in polynomial time:

$$C\,[i, k] = \min_{j<k} \left[C\,[i-1, j] + \frac{cut\,[A_{j,k}, V - A_{j,k}]}{vol\,[A_{j,k}]} \right] \quad (1)$$

$$B\,[i, k] = \operatorname*{argmin}_{j<k} \left[C\,[i-1, j] + \frac{cut\,[A_{j,k}, V - A_{j,k}]}{vol\,[A_{j,k}]} \right] \quad (2)$$

$$\text{s.t. } C\,[0, 1] = 0, C\,[0, k] = \infty,\ 1 < k \leq N \quad (3)$$

$$B\,[0, k] = 1,\ 1 \leq k \leq N \quad (4)$$

$C\,[i, k]$ is the normalized cut value of the optimal segmentation of the first k sentences into i segments. The i-th segment, $A_{j,k}$, begins at node u_j and ends at node u_k. $B\,[i, k]$ is the back-pointer table from which we recover the optimal sequence of segment boundaries. Equations 3 and 4 capture respectively the condition that the normalized cut value of the trivial segmentation of an empty text into one segment is zero and the constraint that the first segment starts with the first node.

The time complexity of the dynamic programming algorithm is $O(KN^2)$, where K is the number of partitions and N is the number of nodes in the graph or sentences in the transcript.

4 Building the Graph

Clearly, the performance of our model depends on the underlying representation, the definition of the pairwise similarity function, and various other model parameters. In this section we provide further details on the graph construction process.

Preprocessing Before building the graph, we apply standard text preprocessing techniques to the text. We stem words with the Porter stemmer (Porter, 1980) to alleviate the sparsity of word counts through stem equivalence classes. We also remove words matching a prespecified list of stop words.

Graph Topology As we noted in the previous section, the normalized cut criterion considers long-term similarity relationships between nodes. This effect is achieved by constructing a fully-connected graph. However, considering all pairwise relations in a long text may be detrimental to segmentation accuracy. Therefore, we discard edges between sentences exceeding a certain threshold distance. This reduction in the graph size also provides us with computational savings.

Similarity Computation In computing pairwise sentence similarities, sentences are represented as vectors of word counts. Cosine similarity is commonly used in text segmentation (Hearst, 1994). To avoid numerical precision issues when summing a series of very small scores, we compute exponentiated cosine similarity scores between pairs of sentence vectors:

$$w(s_i, s_j) = e^{\frac{s_i \cdot s_j}{||s_i|| \times ||s_j||}}$$

We further refine our analysis by smoothing the similarity metric. When comparing two sentences, we also take into account similarity between their immediate neighborhoods. The smoothing is achieved by adding counts of words that occur in adjoining sentences to the current sentence feature vector. These counts are weighted in accordance to their distance from the current sentence:

$$\tilde{s}_i = \sum_{j=i}^{i+k} e^{-\alpha(j-i)} s_j,$$

where s_i are vectors of word counts, and α is a parameter that controls the degree of smoothing.

In the formulation above we use sentences as our nodes. However, we can also represent graph nodes with non-overlapping blocks of words of fixed length. This is desirable, since the lecture transcripts lack sentence boundary markers, and short utterances can skew the cosine similarity scores. The optimal length of the block is tuned on a heldout development set.

Lexical Weighting Previous research has shown that weighting schemes play an important role in segmentation performance (Ji and Zha, 2003; Choi et al., 2001). Of particular concern are words that may not be common in general English discourse but that occur throughout the text for a particular lecture or subject. For example, in a lecture about support vector machines, the occurrence of the term “SVM” is not going to convey a lot of information about the distribution of sub-topics, even though it is a fairly rare term in general English and bears much semantic content. The same words can convey varying degrees of information across different lectures, and term weighting specific to individual lectures becomes important in the similarity computation.

Corpus	Lectures	Segments per Lecture	Total Word Tokens	ASR WER Accuracy
Physics	33	5.9	232K	19.4%
AI	22	12.3	182K	×

Table 1: Lecture Corpus Statistics

In order to address this issue, we introduce a variation on the *tf-idf* scoring scheme used in the information-retrieval literature (Salton and Buckley, 1988). A transcript is split uniformly into N chunks; each chunk serves as the equivalent of documents in the *tf-idf* computation. The weights are computed separately for each transcript, since topic and word distributions vary across lectures.

5 Evaluation Set-Up

In this section we present the different corpora used to evaluate our model and provide a brief overview of the evaluation metrics. Next, we describe our human segmentation study on the corpus of spoken lecture data.

5.1 Parameter Estimation

A heldout development set of three lectures is used for estimating the optimal word block length for representing nodes, the threshold distances for discarding node edges, the number of uniform chunks for estimating *tf-idf* lexical weights, the alpha parameter for smoothing, and the length of the smoothing window. We use a simple greedy search procedure for optimizing the parameters.

5.2 Corpora

We evaluate our segmentation algorithm on three sets of data. Two of the datasets we use are new segmentation collections that we have compiled for this study,[1] and the remaining set includes a standard collection previously used for evaluation of segmentation algorithms. Various corpus statistics for the new datasets are presented in Table 1. Below we briefly describe each corpus.

Physics Lectures Our first corpus consists of spoken lecture transcripts from an undergraduate

[1] Our materials are publicly available at `http://www.csail.mit.edu/~igorm/acl06.html`

Physics class. In contrast to other segmentation datasets, our corpus contains much longer texts. A typical lecture of 90 minutes has 500 to 700 sentences with 8500 words, which corresponds to about 15 pages of raw text. We have access both to manual transcriptions of these lectures and also output from an automatic speech recognition system. The word error rate for the latter is 19.4%,[2] which is representative of state-of-the-art performance on lecture material (Leeuwis et al., 2003).

The Physics lecture transcript segmentations were produced by the teaching staff of the introductory Physics course at the Massachusetts Institute of Technology. Their objective was to facilitate access to lecture recordings available on the class website. This segmentation conveys the high-level topical structure of the lectures. On average, a lecture was annotated with six segments, and a typical segment corresponds to two pages of a transcript.

Artificial Intelligence Lectures Our second lecture corpus differs in subject matter, lecturing style, and segmentation granularity. The graduate Artificial Intelligence class has, on average, twelve segments per lecture, and a typical segment is about half of a page. One segment roughly corresponds to the content of a slide. This time the segmentation was obtained from the lecturer herself. The lecturer went through the transcripts of lecture recordings and segmented the lectures with the objective of making the segments correspond to presentation slides for the lectures.

Due to the low recording quality, we were unable to obtain the ASR transcripts for this class. Therefore, we only use manual transcriptions of these lectures.

Synthetic Corpus Also as part of our analysis, we used the synthetic corpus created by Choi (2000) which is commonly used in the evaluation of segmentation algorithms. This corpus consists of a set of concatenated segments randomly sampled from the Brown corpus. The length of the segments in this corpus ranges from three to eleven sentences. It is important to note that the lexical transitions in these concatenated texts are very sharp, since the segments come from texts written in widely varying language styles on completely different topics.

5.3 Evaluation Metric

We use the P_k and WindowDiff measures to evaluate our system (Beeferman et al., 1999; Pevzner and Hearst, 2002). The P_k measure estimates the probability that a randomly chosen pair of words within a window of length k words is inconsistently classified. The WindowDiff metric is a variant of the P_k measure, which penalizes false positives on an equal basis with near misses.

Both of these metrics are defined with respect to the average segment length of texts and exhibit high variability on real data. We follow Choi (2000) and compute the mean segment length used in determining the parameter k on each reference text separately.

We also plot the Receiver Operating Characteristic (ROC) curve to gauge performance at a finer level of discrimination (Swets, 1988). The ROC plot is the plot of the true positive rate against the false positive rate for various settings of a decision criterion. In our case, the true positive rate is the fraction of boundaries correctly classified, and the false positive rate is the fraction of non-boundary positions incorrectly classified as boundaries. In computing the true and false positive rates, we vary the threshold distance to the true boundary within which a hypothesized boundary is considered correct. Larger areas under the ROC curve of a classifier indicate better discriminative performance.

5.4 Human Segmentation Study

Spoken lectures are very different in style from other corpora used in human segmentation studies (Hearst, 1994; Galley et al., 2003). We are interested in analyzing human performance on a corpus of lecture transcripts with much longer texts and a less clear-cut concept of a sub-topic. We define a segment to be a sub-topic that signals a prominent shift in subject matter. Disregarding this sub-topic change would impair the high-level understanding of the structure and the content of the lecture.

As part of our human segmentation analysis, we asked three annotators to segment the Physics lecture corpus. These annotators had taken the class in the past and were familiar with the subject matter under consideration. We wrote a detailed instruction manual for the task, with annotation guidelines for the most part following the model used by Gruenstein et al. (2005). The annotators were instructed to segment at a level of granularity

[2] A speaker-dependent model of the lecturer was trained on 38 hours of lectures from other courses using the SUMMIT segment-based Speech Recognizer (Glass, 2003).

	O	A	B	C
Mean Seg. Count	6.6	8.9	18.4	13.8
Mean Seg. Length	69.4	51.5	24.9	33.2
Seg. Length Dev.	39.6	37.4	34.5	39.4

Table 2: Annotator Segmentation Statistics for the first ten Physics lectures.

Ref/Hyp	O	A	B	C
O	0	**0.243**	0.418	0.312
A	0.219	0	0.400	0.355
B	0.314	0.337	0	0.332
C	0.260	0.296	0.370	0

Table 3: P_k annotation agreement between different pairs of annotators.

	Edge Cutoff					
	10	25	50	100	200	None
Physics (Manual)						
PK	0.394	0.373	0.341	**0.295**	0.311	0.330
WD	0.404	0.383	0.352	**0.308**	0.329	0.350
Physics (ASR)						
PK	0.440	0.371	0.343	0.330	**0.322**	0.359
WD	0.456	0.383	0.356	0.343	**0.342**	0.398
AI						
PK	0.480	0.422	0.408	0.416	**0.393**	0.397
WD	0.493	0.435	0.420	0.440	**0.424**	0.432
Choi						
PK	0.222	**0.202**	0.213	0.216	0.208	0.208
WD	0.234	**0.222**	0.233	0.238	0.230	0.230

Table 4: Edges between nodes separated beyond a certain threshold distance are removed.

that would identify most of the prominent topical transitions necessary for a summary of the lecture.

The annotators used the NOMOS annotation software toolkit, developed for meeting segmentation (Gruenstein et al., 2005). They were provided with recorded audio of the lectures and the corresponding text transcriptions. We intentionally did not provide the subjects with the target number of boundaries, since we wanted to see if the annotators would converge on a common segmentation granularity.

Table 2 presents the annotator segmentation statistics. We see two classes of segmentation granularities. The original reference (O) and annotator A segmented at a coarse level with an average of 6.6 and 8.9 segments per lecture, respectively. Annotators B and C operated at much finer levels of discrimination with 18.4 and 13.8 segments per lecture on average. We conclude that multiple levels of granularity are acceptable in spoken lecture segmentation. This is expected given the length of the lectures and varying human judgments in selecting relevant topical content.

Following previous studies, we quantify the level of annotator agreement with the P_k measure (Gruenstein et al., 2005).[3] Table 3 shows the annotator agreement scores between different pairs of annotators. P_k measures ranged from 0.24 and 0.42. We observe greater consistency at similar levels of granularity, and less so across the two classes. Note that annotator A operated at a level of granularity consistent with the original reference segmentation. Hence, the 0.24 P_k measure score serves as the benchmark with which we can compare the results attained by segmentation algorithms on the Physics lecture data.

As an additional point of reference we note that the uniform and random baseline segmentations attain 0.469 and 0.493 P_k measure, respectively, on the Physics lecture set.

6 Experimental Results

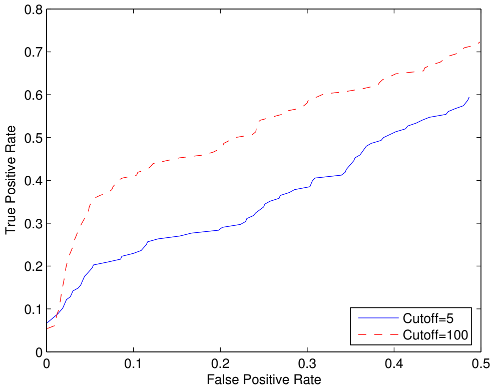


Figure 3: ROC plot for the Minimum Cut Segmenter on thirty Physics Lectures, with edge cutoffs set at five and hundred sentences.

Benefits of global analysis We first determine the impact of long-range pairwise similarity dependencies on segmentation performance. Our

[3] Kappa measure would not be the appropriate measure in this case, because it is not sensitive to near misses, and we cannot make the required independence assumption on the placement of boundaries.

	CHOI	UI	MINCUT
PHYSICS (MANUAL)			
PK	0.372	0.310	**0.298**
WD	0.385	0.323	**0.311**
PHYSICS (ASR TRANSCRIPTS)			
PK	0.361	0.352	**0.322**
WD	0.376	0.364	**0.340**
AI			
PK	0.445	**0.374**	0.383
WD	0.478	0.420	**0.417**
CHOI			
PK	0.110	**0.105**	0.212
WD	0.121	**0.116**	0.234

Table 5: Performance analysis of different algorithms using the P_k and WindowDiff measures, with three lectures heldout for development.

key hypothesis is that considering long-distance lexical relations contributes to the effectiveness of the algorithm. To test this hypothesis, we discard edges between nodes that are more than a certain number of sentences apart. We test the system on a range of data sets, including the Physics and AI lectures and the synthetic corpus created by Choi (2000). We also include segmentation results on Physics ASR transcripts.

The results in Table 4 confirm our hypothesis — taking into account non-local lexical dependencies helps across different domains. On manually transcribed Physics lecture data, for example, the algorithm yields 0.394 P_k measure when taking into account edges separated by up to ten sentences. When dependencies up to a hundred sentences are considered, the algorithm yields a 25% reduction in P_k measure. Figure 3 shows the ROC plot for the segmentation of the Physics lecture data with different cutoff parameters, again demonstrating clear gains attained by employing long-range dependencies. As Table 4 shows, the improvement is consistent across all lecture datasets. We note, however, that after some point increasing the threshold degrades performance, because it introduces too many spurious dependencies (see the last column of Table 4). The speaker will occasionally return to a topic described at the beginning of the lecture, and this will bias the algorithm to put the segment boundary closer to the end of the lecture.

Long-range dependencies do not improve the performance on the synthetic dataset. This is expected since the segments in the synthetic dataset are randomly selected from widely-varying documents in the Brown corpus, even spanning different genres of written language. So, effectively, there are no genuine long-range dependencies that can be exploited by the algorithm.

Comparison with local dependency models We compare our system with the state-of-the-art similarity-based segmentation system developed by Choi (2000). We use the publicly available implementation of the system and optimize the system on a range of mask-sizes and different parameter settings described in (Choi, 2000) on a heldout development set of three lectures. To control for segmentation granularity, we specify the number of segments in the reference ("O") segmentation for both our system and the baseline. Table 5 shows that the Minimum Cut algorithm consistently outperforms the similarity-based baseline on all the lecture datasets. We attribute this gain to the presence of more attenuated topic transitions in spoken language. Since spoken language is more spontaneous and less structured than written language, the speaker needs to keep the listener abreast of the changes in topic content by introducing subtle cues and references to prior topics in the course of topical transitions. Non-local dependencies help to elucidate shifts in focus, because the strength of a particular transition is measured with respect to other local and long-distance contextual discourse relationships.

Our system does not outperform Choi's algorithm on the synthetic data. This again can be attributed to the discrepancy in distributional properties of the synthetic corpus which lacks coherence in its thematic shifts and the lecture corpus of spontaneous speech with smooth distributional variations. We also note that we did not try to adjust our model to optimize its performance on the synthetic data. The smoothing method developed for lecture segmentation may not be appropriate for short segments ranging from three to eleven sentences that constitute the synthetic set.

We also compared our method with another state-of-the-art algorithm which does not explicitly rely on pairwise similarity analysis. This algorithm (Utiyama and Isahara, 2001) (UI) computes the optimal segmentation by estimating changes in the language model predictions over different partitions. We used the publicly available implementation of the system that does not require parameter tuning on a heldout development set.

Again, our method achieves favorable performance on a range of lecture data sets (See Table 5), and both algorithms attain results close to the range of human agreement scores. The attractive feature of our algorithm, however, is robustness to recognition errors — testing it on the ASR transcripts caused only 7.8% relative increase in P_k measure (from 0.298 to 0.322), compared to a 13.5% relative increase for the UI system. We attribute this feature to the fact that the model is less dependent on individual recognition errors, which have a detrimental effect on the local segment language modeling probability estimates for the UI system. The block-level similarity function is not as sensitive to individual word errors, because the partition volume normalization factor dampens their overall effect on the derived models.

7 Conclusions

In this paper we studied the impact of long-range dependencies on the accuracy of text segmentation. We modeled text segmentation as a graph-partitioning task aiming to simultaneously optimize the total similarity within each segment and dissimilarity across various segments. We showed that global analysis of lexical distribution improves the segmentation accuracy and is robust in the presence of recognition errors. Combining global analysis with advanced methods for smoothing (Ji and Zha, 2003) and weighting could further boost the segmentation performance.

Our current implementation does not automatically determine the granularity of a resulting segmentation. This issue has been explored in the past (Ji and Zha, 2003; Utiyama and Isahara, 2001), and we will explore the existing strategies in our framework. We believe that the algorithm has to produce segmentations for various levels of granularity, depending on the needs of the application that employs it.

Our ultimate goal is to automatically generate tables of content for lectures. We plan to investigate strategies for generating titles that will succinctly describe the content of each segment. We will explore how the interaction between the generation and segmentation components can improve the performance of such a system as a whole.

8 Acknowledgements

The authors acknowledge the support of the National Science Foundation (CAREER grant IIS-0448168, grant IIS-0415865, and the NSF Graduate Fellowship). Any opinions, findings, conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We would like to thank Masao Utiyama for providing us with an implementation of his segmentation system and Alex Gruenstein for assisting us with the NOMOS toolkit. We are grateful to David Karger for an illuminating discussion on the Minimum Cut algorithm. We also would like to acknowledge the MIT NLP and Speech Groups, the three annotators, and the three anonymous reviewers for valuable comments, suggestions, and help.

References

D. Beeferman, A. Berger, J. D. Lafferty. 1999. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177–210.

F. Choi, P. Wiemer-Hastings, J. Moore. 2001. Latent semantic analysis for text segmentation. In *Proceedings of EMNLP*, 109–117.

F. Y. Y. Choi. 2000. Advances in domain independent linear text segmentation. In *Proceedings of the NAACL*, 26–33.

K. W. Church. 1993. Char_align: A program for aligning parallel texts at the character level. In *Proceedings of the ACL*, 1–8.

M. Galley, K. McKeown, E. Fosler-Lussier, H. Jing. 2003. Discourse segmentation of multi-party conversation. In *Proceedings of the ACL*, 562–569.

J. R. Glass. 2003. A probabilistic framework for segment-based speech recognition. *Computer Speech and Language*, 17(2–3):137–152.

A. Gruenstein, J. Niekrasz, M. Purver. 2005. Meeting structure annotation: Data and tools. In *Proceedings of the SIGdial Workshop on Discourse and Dialogue*, 117–127.

M. A. K. Halliday, R. Hasan. 1976. *Cohesion in English*. Longman, London.

M. Hearst. 1994. Multi-paragraph segmentation of expository text. In *Proceedings of the ACL*, 9–16.

X. Ji, H. Zha. 2003. Domain-independent text segmentation using anisotropic diffusion and dynamic programming. In *Proceedings of SIGIR*, 322–329.

A. Kehagias, P. Fragkou, V. Petridis. 2003. Linear text segmentation using a dynamic programming algorithm. In *Proceedings of the EACL*, 171–178.

E. Leeuwis, M. Federico, M. Cettolo. 2003. Language modeling and transcription of the ted corpus lectures. In *Proceedings of ICASSP*, 232–235.

L. Pevzner, M. Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):pp. 19–36.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

J. Reynar. 1998. *Topic segmentation: Algorithms and applications*. Ph.D. thesis, University of Pennsylvania.

G. Salton, C. Buckley. 1988. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523.

J. Shi, J. Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

J. Swets. 1988. Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1293.

M. Utiyama, H. Isahara. 2001. A statistical model for domain-independent text segmentation. In *Proceedings of the ACL*, 499–506.

Bootstrapping Path-Based Pronoun Resolution

Shane Bergsma
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2E8
bergsma@cs.ualberta.ca

Dekang Lin
Google, Inc.
1600 Amphitheatre Parkway,
Mountain View, California, 94301
lindek@google.com

Abstract

We present an approach to pronoun resolution based on syntactic paths. Through a simple bootstrapping procedure, we learn the likelihood of coreference between a pronoun and a candidate noun based on the path in the parse tree between the two entities. This path information enables us to handle previously challenging resolution instances, and also robustly addresses traditional syntactic coreference constraints. Highly coreferent paths also allow mining of precise probabilistic gender/number information. We combine statistical knowledge with well known features in a Support Vector Machine pronoun resolution classifier. Significant gains in performance are observed on several datasets.

1 Introduction

Pronoun resolution is a difficult but vital part of the overall coreference resolution task. In each of the following sentences, a pronoun resolution system must determine what the pronoun *his* refers to:

(1) John needs *his* friend.

(2) John needs *his* support.

In (1), *John* and *his* corefer. In (2), *his* refers to some other, perhaps previously evoked entity. Traditional pronoun resolution systems are not designed to distinguish between these cases. They lack the specific world knowledge required in the second instance – the knowledge that a person does not usually explicitly need his own support.

We collect statistical path-coreference information from a large, automatically-parsed corpus to address this limitation. A *dependency path* is defined as the sequence of dependency links between two potentially coreferent entities in a parse tree. A path does not include the terminal entities; for example, "John needs his support" and "He needs their support" have the same syntactic path. Our algorithm determines that the dependency path linking the *Noun* and *pronoun* is very likely to connect coreferent entities for the path "*Noun* needs *pronoun*'s friend," while it is rarely coreferent for the path "*Noun* needs *pronoun*'s support."

This likelihood can be learned by simply counting how often we see a given path in text with an initial *Noun* and a final *pronoun* that are from the same/different gender/number classes. Cases such as "John needs her support" or "They need his support" are much more frequent in text than cases where the subject noun and pronoun terminals agree in gender/number. When there is agreement, the terminal nouns are likely to be coreferent. When they disagree, they refer to different entities. After a sufficient number of occurrences of agreement or disagreement, there is a strong statistical indication of whether the path is *coreferent* (terminal nouns tend to refer to the same entity) or *non-coreferent* (nouns refer to different entities).

We show that including path coreference information enables significant performance gains on three third-person pronoun resolution experiments. We also show that coreferent paths can provide the seed information for bootstrapping other, even more important information, such as the gender/number of noun phrases.

2 Related Work

Coreference resolution is generally conducted as a pairwise classification task, using various constraints and preferences to determine whether two

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 33–40,
Sydney, July 2006. ©2006 Association for Computational Linguistics

expressions corefer. Coreference is typically only allowed between nouns matching in gender and number, and not violating any intrasentential syntactic principles. Constraints can be applied as a preprocessing step to scoring candidates based on distance, grammatical role, etc., with scores developed either manually (Lappin and Leass, 1994), or through a machine-learning algorithm (Kehler et al., 2004). Constraints and preferences have also been applied together as decision nodes on a decision tree (Aone and Bennett, 1995).

When previous resolution systems handle cases like (1) and (2), where no disagreement or syntactic violation occurs, coreference is therefore determined by the weighting of features or learned decisions of the resolution classifier. Without path coreference knowledge, a resolution process would resolve the pronouns in (1) and (2) the same way. Indeed, coreference resolution research has focused on the importance of the *strategy* for combining well known constraints and preferences (Mitkov, 1997; Ng and Cardie, 2002), devoting little attention to the development of new features for these difficult cases. The application of *world knowledge* to pronoun resolution has been limited to the semantic compatibility between a candidate noun and the pronoun's context (Yang et al., 2005). We show semantic compatibility can be effectively combined with path coreference information in our experiments below.

Our method for determining path coreference is similar to an algorithm for discovering paraphrases in text (Lin and Pantel, 2001). In that work, the beginning and end nodes in the paths are collected, and two paths are said to be similar (and thus likely paraphrases of each other) if they have similar terminals (i.e. the paths occur with a similar *distribution*). Our work does not need to store the terminals themselves, only whether they are from the same pronoun group. Different paths are not compared in any way; each path is individually assigned a coreference likelihood.

3 Path Coreference

We define a *dependency path* as the sequence of nodes and dependency labels between two potentially coreferent entities in a dependency parse tree. We use the structure induced by the minimalist parser Minipar (Lin, 1998) on sentences from the news corpus described in Section 4. Figure 1 gives the parse tree of (2). As a short-form, we write the dependency path in this case as "*Noun* needs *pronoun*'s support." The path itself does not include the terminal nouns "John" and "his."

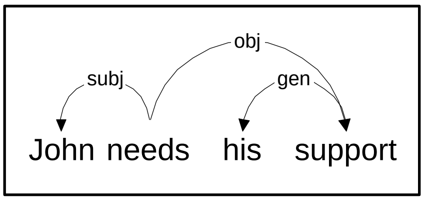


Figure 1: Example dependency tree.

Our algorithm finds the likelihood of coreference along dependency paths by counting the number of times they occur with terminals that are either *likely coreferent* or *non-coreferent*. In the simplest version, we count paths with terminals that are both pronouns. We partition pronouns into seven groups of matching gender, number, and person; for example, the first person singular group contains *I*, *me*, *my*, *mine*, and *myself*. If the two terminal pronouns are from the same group, coreference along the path is likely. If they are from different groups, like *I* and *his*, then they are non-coreferent. Let $N_S(p)$ be the number of times the two terminal pronouns of a path, p, are from the same pronoun group, and let $N_D(p)$ be the number of times they are from different groups. We define the *coreference* of p as:

$$C(p) = \frac{N_S(p)}{N_S(p) + N_D(p)}$$

Our statistics indicate the example path, "*Noun* needs *pronoun*'s support," has a low $C(p)$ value. We could use this fact to prevent us from resolving "his" to "John" when "John needs his support" is presented to a pronoun resolution system.

To mitigate data sparsity, we represent the path with the root form of the verbs and nouns. Also, we use Minipar's named-entity recognition to replace named-entity nouns by the semantic category of their named-entity, when available. All modifiers not on the direct path, such as adjectives, determiners and adverbs, are not considered. We limit the maximum path length to eight nodes.

Tables 1 and 2 give examples of coreferent and non-coreferent paths learned by our algorithm and identified in our test sets. *Coreferent* paths are defined as paths with a $C(p)$ value (and overall number of occurrences) above a certain threshold, indicating the terminal entities are highly likely

Table 1: Example coreferent paths: *Italicized* entities generally corefer.

	Pattern	Example
1.	*Noun* left ... to *pronoun*'s wife	*Buffett* will leave the stock to *his* wife.
2.	*Noun* says *pronoun* intends...	*The newspaper* says *it* intends to file a lawsuit.
3.	*Noun* was punished for *pronoun*'s crime.	*The criminal* was punished for *his* crime.
4.	... left *Noun* to fend for *pronoun-self*	They left *Jane* to fend for *herself*.
5.	*Noun* lost *pronoun*'s job.	*Dick* lost *his* job.
6.	... created *Noun* and populated *pronoun*.	Nzame created *the earth* and populated *it*
7.	*Noun* consolidated *pronoun*'s power.	*The revolutionaries* consolidated *their* power.
8.	*Noun* suffered ... in *pronoun*'s knee ligament.	*The leopard* suffered pain in *its* knee ligament.

to corefer. *Non-coreferent* paths have a $C(p)$ below a certain cutoff; the terminals are highly unlikely to corefer. Especially note the challenge of resolving most of the examples in Table 2 without path coreference information. Although these paths encompass some cases previously covered by Binding Theory (e.g. "Mary suspended her," *her* cannot refer to *Mary* by Principle B (Haegeman, 1994)), most have no syntactic justification for non-coreference *per se*. Likewise, although Binding Theory (Principle A) could identify the reflexive pronominal relationship of Example 4 in Table 1, most cases cannot be resolved through syntax alone. Our analysis shows that successfully handling cases that may have been handled with Binding Theory constitutes only a small portion of the total performance gain using path coreference.

In any case, Binding Theory remains a challenge with a noisy parser. Consider: "Alex gave her money." Minipar parses *her* as a possessive, when it is more likely an object, "Alex gave money *to her*." Without a correct parse, we cannot rule out the link between *her* and *Alex* through Binding Theory. Our algorithm, however, learns that the path "*Noun* gave *pronoun*'s money," is non-coreferent. In a sense, it corrects for parser errors by learning when coreference should be blocked, given *any* consistent parse of the sentence.

We obtain path coreference for millions of paths from our parsed news corpus (Section 4). While Tables 1 and 2 give test set examples, many other interesting paths are obtained. We learn coreference is unlikely between the nouns in "Bob married his mother," or "Sue wrote her obituary." The fact you don't marry your own mother or write your own obituary is perhaps obvious, but this is the first time this kind of knowledge has been made available computationally. Naturally, exceptions to the coreference or non-coreference of some of these paths can be found; our patterns represent general trends only. And, as mentioned above, reliable path coreference is somewhat dependent on consistent parsing.

Paths connecting pronouns to pronouns are different than paths connecting both nouns and pronouns to pronouns – the case we are ultimately interested in resolving. Consider "Company A gave its data on its website." The pronoun-pronoun path coreference algorithm described above would learn the terminals in "*Noun*'s data on *pronoun*'s website" are often coreferent. But if we see the phrase "Company A gave Company B's data on its website," then "its" is not likely to refer to "Company B," even though we identified this as a coreferent path! We address this problem with a two-stage extraction procedure. We first bootstrap gender/number information using the pronoun-pronoun paths as described in Section 4.1. We then use this gender/number information to count paths where an initial *noun* (with probabilistically-assigned gender/number) and following pronoun are connected by the dependency path, recording the agreement or disagreement of their gender/number category.[1] These superior paths are then used to re-bootstrap our final gender/number information used in the evaluation (Section 6).

We also bootstrap paths where the nodes in the path are replaced by their grammatical category. This allows us to learn general syntactic constraints not dependent on the surface forms of the words (including, but not limited to, the Binding Theory principles). A separate set of these non-coreferent paths is also used as a feature in our sys-

[1] As desired, this modification allows the first example to provide two instances of noun-pronoun paths with terminals from the same gender/number group, linking each "its" to the subject noun "Company A", rather than to each other.

Table 2: Example non-coreferent paths: *Italicized* entities do *not* generally corefer

	Pattern	Example
1.	*Noun* thanked ... for *pronoun*'s assistance	*John* thanked him for *his* assistance.
2.	*Noun* wanted *pronoun* to lie.	*The president* wanted *her* to lie.
3.	... *Noun* into *pronoun*'s pool	Max put the *floaties* into *their* pool.
4.	... use *Noun* to *pronoun*'s advantage	The company used *the delay* to *its* advantage.
5.	*Noun* suspended *pronoun*	*Mary* suspended *her*.
6.	*Noun* was *pronoun*'s relative.	*The Smiths* were *their* relatives.
7.	*Noun* met *pronoun*'s demands	*The players' association* met *its* demands.
8.	... put *Noun* at the top of *pronoun*'s list.	The government put *safety* at the top of *its* list.

tem. We also tried expanding our coverage by using paths *similar* to paths with known path coreference (based on distributionally similar words), but this did not generally increase performance.

4 Bootstrapping in Pronoun Resolution

Our determination of path coreference can be considered a bootstrapping procedure. Furthermore, the coreferent paths themselves can serve as the seed for bootstrapping additional coreference information. In this section, we sketch previous approaches to bootstrapping in coreference resolution and explain our new ideas.

Coreference bootstrapping works by assuming resolutions in unlabelled text, acquiring information from the putative resolutions, and then making inferences from the aggregate statistical data. For example, we assumed two pronouns from the same pronoun group were coreferent, and deduced path coreference from the accumulated counts.

The potential of the bootstrapping approach can best be appreciated by imagining millions of documents with coreference annotations. With such a set, we could extract fine-grained features, perhaps tied to individual words or paths. For example, we could estimate the likelihood each noun belongs to a particular gender/number class by the proportion of times this noun was labelled as the antecedent for a pronoun of this particular gender/number.

Since no such corpus exists, researchers have used coarser features learned from smaller sets through supervised learning (Soon et al., 2001; Ng and Cardie, 2002), manually-defined coreference patterns to mine specific kinds of data (Bean and Riloff, 2004; Bergsma, 2005), or accepted the noise inherent in unsupervised schemes (Ge et al., 1998; Cherry and Bergsma, 2005).

We address the drawbacks of these approaches by using coreferent paths as the assumed resolutions in the bootstrapping. Because we can vary the threshold for defining a coreferent path, we can trade-off coverage for precision. We now outline two potential uses of bootstrapping with coreferent paths: learning gender/number information (Section 4.1) and augmenting a semantic compatibility model (Section 4.2). We bootstrap this data on our automatically-parsed news corpus. The corpus comprises 85 GB of news articles taken from the world wide web over a 1-year period.

Table 3: Gender classification performance (%)

Classifier	F-Score
Bergsma (2005) Corpus-based	85.4
Bergsma (2005) Web-based	90.4
Bergsma (2005) Combined	92.2
Duplicated Corpus-based	88.0
Coreferent Path-based	90.3

4.1 Probabilistic Gender/Number

Bergsma (2005) learns noun gender (and number) from two principal sources: 1) mining it from manually-defined lexico-syntactic patterns in parsed corpora, and 2) acquiring it on the fly by counting the number of pages returned for various gender-indicating patterns by the Google search engine. The web-based approach outperformed the corpus-based approach, while a system that combined the two sets of information resulted in the highest performance (Table 3). The combined gender-classifying system is a machine-learned classifier with 20 features.

The time delay of using an Internet search engine within a large-scale anaphora resolution effort is currently impractical. Thus we attempted

Table 4: Example gender/number probability (%)

Word	*masc*	*fem*	*neut*	*plur*
company	0.6	0.1	98.1	1.2
condoleeza rice	4.0	92.7	0.0	3.2
pat	58.3	30.6	6.2	4.9
president	94.1	3.0	1.5	1.4
wife	9.9	83.3	0.8	6.1

to duplicate Bergsma's corpus-based extraction of gender and number, where the information can be stored in advance in a table, but using a much larger data set. Bergsma ran his extraction on roughly 6 GB of text; we used roughly 85 GB.

Using the test set from Bergsma (2005), we were only able to boost performance from an F-Score of 85.4% to one of 88.0% (Table 3). This result led us to re-examine the high performance of Bergsma's web-based approach. We realized that the corpus-based and web-based approaches are not exactly symmetric. The corpus-based approaches, for example, would not pick out gender from a pattern such as "John and his friends..." because "*Noun* and *pronoun*'s NP" is not one of the manually-defined gender extraction patterns. The web-based approach, however, would catch this instance with the "John * his/her/its/their" template, where "*" is the Google wild-card operator. Clearly, there are patterns useful for capturing gender and number information beyond the predefined set used in the corpus-based extraction.

We thus decided to capture gender/number information from coreferent paths. If a noun is connected to a pronoun of a particular gender along a coreferent path, we count this as an instance of that noun being that gender. In the end, the probability that the noun is a particular gender is the proportion of times it was connected to a pronoun of that gender along a coreferent path. Gender information becomes a single intuitive, accessible feature (i.e. the probability of the noun being that gender) rather than Bergsma's 20-dimensional feature vector requiring search-engine queries to instantiate.

We acquire gender and number data for over 3 million nouns. We use add-one smoothing for data sparsity. Some example gender/number probabilities are given in Table 4 (cf. (Ge et al., 1998; Cherry and Bergsma, 2005)). We get a performance of 90.3% (Table 3), again meeting our requirements of high performance and allowing for a fast, practical implementation. This is lower than Bergsma's top score of 92.2% (Figure 3), but again, Bergsma's top system relies on Google search queries for each new word, while ours are all pre-stored in a table for fast access.

We are pleased to be able to share our gender and number data with the NLP community.[2] In Section 6, we show the benefit of this data as a probabilistic feature in our pronoun resolution system. Probabilistic data is useful because it allows us to rapidly prototype resolution systems without incurring the overhead of large-scale lexical databases such as WordNet (Miller et al., 1990).

4.2 Semantic Compatibility

Researchers since Dagan and Itai (1990) have variously argued for and against the utility of collocation statistics between nouns and parents for improving the performance of pronoun resolution. For example, can the verb parent of a pronoun be used to select antecedents that satisfy the verb's selectional restrictions? If the verb phrase was *shatter it*, we would expect *it* to refer to some kind of brittle entity. Like path coreference, semantic compatibility can be considered a form of *world knowledge* needed for more challenging pronoun resolution instances.

We encode the semantic compatibility between a noun and its parse tree parent (and grammatical relationship with the parent) using mutual information (MI) (Church and Hanks, 1989). Suppose we are determining whether *ham* is a suitable antecedent for the pronoun *it* in *eat it*. We calculate the MI as:

$$\mathrm{MI}(\text{eat:obj}, \text{ham}) = \log \frac{\Pr(\text{eat:obj:ham})}{\Pr(\text{eat:obj})\Pr(\text{ham})}$$

Although semantic compatibility is usually only computed for possessive-noun, subject-verb, and verb-object relationships, we include 121 different kinds of syntactic relationships as parsed in our news corpus.[3] We collected 4.88 billion *parent:rel:node* triples, including over 327 million possessive-noun values, 1.29 billion subject-verb and 877 million verb-direct object. We use small probability values for unseen Pr(*parent:rel:node*), Pr(*parent:rel*), and Pr(*node*) cases, as well as a default MI when no relationship is parsed, roughly optimized for performance on the training set. We

[2] Available at http://www.cs.ualberta.ca/~bergsma/Gender/

[3] We convert prepositions to relationships to enhance our model's semantics, e.g. *Joan:of:Arc* rather than *Joan:prep:of*

include both the MI between the noun and the pronoun's parent as well as the MI between the pronoun and the noun's parent as features in our pronoun resolution classifier.

Kehler et al. (2004) saw no apparent gain from using semantic compatibility information, while Yang et al. (2005) saw about a 3% improvement with compatibility data acquired by searching on the world wide web. Section 6 analyzes the contribution of MI to our system.

Bean and Riloff (2004) used bootstrapping to extend their semantic compatibility model, which they called contextual-role knowledge, by identifying certain cases of easily-resolved anaphors and antecedents. They give the example "Mr. Bush disclosed the policy by reading it." Once we identify that *it* and *policy* are coreferent, we include *read:obj:policy* as part of the compatibility model.

Rather than using manually-defined heuristics to bootstrap additional semantic compatibility information, we wanted to enhance our MI statistics automatically with coreferent paths. Consider the phrase, "Saddam's wife got a Jordanian lawyer for her husband." It is unlikely we would see "wife's husband" in text; in other words, we would not know that *husband:gen:wife* is, in fact, semantically compatible and thereby we would discourage selection of "wife" as the antecedent at resolution time. However, because "*Noun* gets ... for *pronoun*'s husband" is a coreferent path, we could capture the above relationship by adding a *parent:rel:node* for every pronoun connected to a noun phrase along a coreferent path in text.

We developed context models with and without these path enhancements, but ultimately we could find no subset of coreferent paths that improve the semantic compatibility's contribution to training set accuracy. A mutual information model trained on 85 GB of text is fairly robust on its own, and any kind of bootstrapped extension seems to cause more damage by increased noise than can be compensated by increased coverage. Although we like knowing audiences have noses, e.g. "the audience turned up its nose at the performance," such phrases are apparently quite rare in actual test sets.

5 Experimental Design

The noun-pronoun path coreference can be used directly as a feature in a pronoun resolution system. However, path coreference is undefined for cases where there is no path between the pronoun and the candidate noun – for example, when the candidate is in the previous sentence. Therefore, rather than using path coreference directly, we have features that are true if $C(p)$ is above or below certain thresholds. The features are thus set when coreference between the pronoun and candidate noun is likely (a coreferent path) or unlikely (a non-coreferent path).

We now evaluate the utility of path coreference within a state-of-the-art machine-learned resolution system for third-person pronouns with nominal antecedents. A standard set of features is used along with the bootstrapped gender/number, semantic compatibility, and path coreference information. We refer to these features as our "probabilistic features" (Prob. Features) and run experiments using the full system trained and tested with each absent, in turn (Table 5). We have 29 features in total, including measures of candidate distance, frequency, grammatical role, and different kinds of parallelism between the pronoun and the candidate noun. Several reliable features are used as hard constraints, removing candidates before consideration by the scoring algorithm.

All of the parsing, noun-phrase identification, and named-entity recognition are done automatically with Minipar. Candidate antecedents are considered in the current and previous sentence only. We use SVM^{light} (Joachims, 1999) to learn a linear-kernel classifier on pairwise examples in the training set. When resolving pronouns, we select the candidate with the farthest positive distance from the SVM classification hyperplane.

Our training set is the anaphora-annotated portion of the American National Corpus (ANC) used in Bergsma (2005), containing 1270 anaphoric pronouns[4]. We test on the ANC Test set (1291 instances) also used in Bergsma (2005) (highest resolution accuracy reported: 73.3%), the anaphora-labelled portion of AQUAINT used in Cherry and Bergsma (2005) (1078 instances, highest accuracy: 71.4%), and the anaphoric pronoun subset of the MUC7 (1997) coreference evaluation formal test set (169 instances, highest *precision* of 62.1 reported on all pronouns in (Ng and Cardie, 2002)). These particular corpora were chosen so we could test our approach using the same data as comparable machine-learned systems exploiting probabilistic information sources. Parameters

[4] See http://www.cs.ualberta.ca/˜bergsma/CorefTags/ for instructions on acquiring annotations

Table 5: Resolution accuracy (%)

Dataset		ANC	AQT	MUC
1	Previous noun	36.7	34.5	30.8
2	No Prob. Features	58.1	60.9	49.7
3	No Prob. Gender	65.8	71.0	68.6
4	No MI	71.3	73.5	69.2
5	No $C(p)$	72.3	73.7	69.8
6	Full System	73.9	75.0	71.6
7	Upper Bound	93.2	92.3	91.1

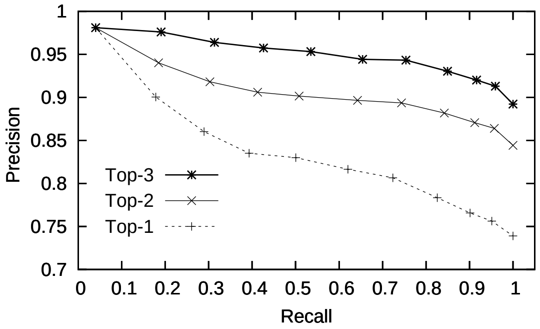


Figure 2: ANC pronoun resolution accuracy for varying SVM-thresholds.

were set using cross-validation on the training set; test sets were used only once to obtain the final performance values.

Evaluation Metric: We report results in terms of accuracy: Of all the anaphoric pronouns in the test set, the proportion we resolve correctly.

6 Results and Discussion

We compare the accuracy of various configurations of our system on the ANC, AQT and MUC datasets (Table 5). We include the score from picking the noun immediately preceding the pronoun (after our hard filters are applied). Due to the hard filters and limited search window, it is not possible for our system to resolve every noun to a correct antecedent. We thus provide the performance upper bound (i.e. the proportion of cases with a correct answer in the filtered candidate list). On ANC and AQT, each of the probabilistic features results in a statistically significant gain in performance over a model trained and tested with that feature absent.[5] On the smaller MUC set, none of the differences in 3-6 are statistically significant, however, the relative contribution of the various features remains reassuringly constant.

Aside from missing antecedents due to the hard filters, the main sources of error include inaccurate statistical data and a classifier bias toward preceding *pronouns* of the same gender/number. It would be interesting to see whether performance could be improved by adding WordNet and web-mined features. Path coreference itself could conceivably be determined with a search engine.

Gender is our most powerful probabilistic feature. In fact, inspecting our system's decisions, gender often rules out coreference regardless of path coreference. This is not surprising, since we based the acquisition of $C(p)$ on gender. That is, our bootstrapping assumption was that the majority of times these paths occur, gender indicates coreference or lack thereof. Thus when they occur in our test sets, gender should often sufficiently indicate coreference. Improving the orthogonality of our features remains a future challenge.

Nevertheless, note the decrease in performance on each of the datasets when $C(p)$ is excluded (#5). This is compelling evidence that path coreference is valuable in its own right, beyond its ability to bootstrap extensive and reliable gender data.

Finally, we can add ourselves to the camp of people claiming semantic compatibility is useful for pronoun resolution. Both the MI from the pronoun in the antecedent's context and vice-versa result in improvement. Building a model from enough text may be the key.

The primary goal of our evaluation was to assess the benefit of path coreference within a competitive pronoun resolution system. Our system does, however, outperform previously published results on these datasets. Direct comparison of our scoring system to other current top approaches is made difficult by differences in preprocessing. Ideally we would assess the benefit of our probabilistic features using the same state-of-the-art preprocessing modules employed by others such as (Yang et al., 2005) (who additionally use a search engine for compatibility scoring). Clearly, promoting competitive evaluation of pronoun resolution scoring systems by giving competitors equivalent real-world preprocessing output along the lines of (Barbu and Mitkov, 2001) remains the best way to isolate areas for system improvement.

Our pronoun resolution system is part of a larger information retrieval project where resolution ac-

[5] We calculate significance with McNemar's test, p=0.05.

curacy is not necessarily the most pertinent measure of classifier performance. More than one candidate can be useful in ambiguous cases, and not every resolution need be used. Since the SVM ranks antecedent candidates, we can test this ranking by selecting more than the top candidate (Top-n) and evaluating coverage of the true antecedents. We can also resolve only those instances where the most likely candidate is above a certain distance from the SVM threshold. Varying this distance varies the precision-recall (PR) of the overall resolution. A representative PR curve for the Top-n classifiers is provided (Figure 2). The corresponding information retrieval performance can now be evaluated along the Top-n / PR configurations.

7 Conclusion

We have introduced a novel feature for pronoun resolution called path coreference, and demonstrated its significant contribution to a state-of-the-art pronoun resolution system. This feature aids coreference decisions in many situations not handled by traditional coreference systems. Also, by bootstrapping with the coreferent paths, we are able to build the most complete and accurate table of probabilistic gender information yet available. Preliminary experiments show path coreference bootstrapping can also provide a means of identifying pleonastic pronouns, where pleonastic neutral pronouns are often followed in a dependency path by a terminal noun of different gender, and cataphoric constructions, where the pronouns are often followed by nouns of matching gender.

References

Chinatsu Aone and Scott William Bennett. 1995. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 122–129.

Catalina Barbu and Ruslan Mitkov. 2001. Evaluation tool for rule-based anaphora resolution methods. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 34–41.

David L. Bean and Ellen Riloff. 2004. Unsupervised learning of contextual role knowledge for coreference resolution. In *HLT-NAACL*, pages 297–304.

Shane Bergsma. 2005. Automatic acquisition of gender information for anaphora resolution. In *Proceedings of the Eighteenth Canadian Conference on Artificial Intelligence (Canadian AI'2005)*, pages 342–353.

Colin Cherry and Shane Bergsma. 2005. An expectation maximization approach to pronoun resolution. In *Proceedings of the Ninth Conference on Natural Language Learning (CoNLL-2005)*, pages 88–95.

Kenneth Ward Church and Patrick Hanks. 1989. Word association norms, mutual information, and lexicography. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL'89)*, pages 76–83.

Ido Dagan and Alan Itai. 1990. Automatic processing of large corpora for the resolution of anaphora references. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING-90)*, volume 3, pages 330–332, Helsinki, Finland.

Niyu Ge, John Hale, and Eugene Charniak. 1998. A statistical approach to anaphora resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 161–171.

Liliane Haegeman. 1994. *Introduction to Government & Binding theory: Second Edition*. Basil Blackwell, Cambridge, UK.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf and C. Burges, editors, *Advances in Kernel Methods*. MIT-Press.

Andrew Kehler, Douglas Appelt, Lara Taylor, and Aleksandr Simma. 2004. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proceedings of HLT/NAACL-04*, pages 289–296.

Shalom Lappin and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *Proceedings of the Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation*.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography*, 3(4):235–244.

Ruslan Mitkov. 1997. Factors in anaphora resolution: they are not the only things that matter. a case study based on two different approaches. In *Proceedings of the ACL '97 / EACL '97 Workshop on Operational Factors in Practical, Robust Anaphora Resolution*, pages 14–21.

MUC-7. 1997. Coreference task definition (v3.0, 13 Jul 97). In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111.

Wee Meng Soon, 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.

Xiaofeng Yang, Jian Su, and Chew Lim Tan. 2005. Improving pronoun resolution using statistics-based semantic compatibility information. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 165–172, June.

Kernel-Based Pronoun Resolution with Structured Syntactic Knowledge

Xiaofeng Yang[†] **Jian Su**[†] **Chew Lim Tan**[‡]

[†]Institute for Infocomm Research
21 Heng Mui Keng Terrace,
Singapore, 119613
{xiaofengy,sujian}@i2r.a-star.edu.sg

[‡] Department of Computer Science
National University of Singapore,
Singapore, 117543
tancl@comp.nus.edu.sg

Abstract

Syntactic knowledge is important for pronoun resolution. Traditionally, the syntactic information for pronoun resolution is represented in terms of features that have to be selected and defined heuristically. In the paper, we propose a kernel-based method that can automatically mine the syntactic information from the parse trees for pronoun resolution. Specifically, we utilize the parse trees directly as a structured feature and apply kernel functions to this feature, as well as other normal features, to learn the resolution classifier. In this way, our approach avoids the efforts of decoding the parse trees into the set of flat syntactic features. The experimental results show that our approach can bring significant performance improvement and is reliably effective for the pronoun resolution task.

1 Introduction

Pronoun resolution is the task of finding the correct antecedent for a given pronominal anaphor in a document. Prior studies have suggested that syntactic knowledge plays an important role in pronoun resolution. For a practical pronoun resolution system, the syntactic knowledge usually comes from the parse trees of the text. The issue that arises is how to effectively incorporate the syntactic information embedded in the parse trees to help resolution. One common solution seen in previous work is to define a set of features that represent particular syntactic knowledge, such as the grammatical role of the antecedent candidates, the governing relations between the candidate and the pronoun, and so on. These features are calculated by mining the parse trees, and then could be used for resolution by using manually designed rules (Lappin and Leass, 1994; Kennedy and Boguraev, 1996; Mitkov, 1998), or using machine-learning methods (Aone and Bennett, 1995; Yang et al., 2004; Luo and Zitouni, 2005).

However, such a solution has its limitation. The syntactic features have to be selected and defined manually, usually by linguistic intuition. Unfortunately, what kinds of syntactic information are effective for pronoun resolution still remains an open question in this research community. The heuristically selected feature set may be insufficient to represent all the information necessary for pronoun resolution contained in the parse trees.

In this paper we will explore how to utilize the syntactic parse trees to help learning-based pronoun resolution. Specifically, we directly utilize the parse trees as a structured feature, and then use a kernel-based method to automatically mine the knowledge embedded in the parse trees. The structured syntactic feature, together with other normal features, is incorporated in a trainable model based on Support Vector Machine (SVM) (Vapnik, 1995) to learn the decision classifier for resolution. Indeed, using kernel methods to mine structural knowledge has shown success in some NLP applications like parsing (Collins and Duffy, 2002; Moschitti, 2004) and relation extraction (Zelenko et al., 2003; Zhao and Grishman, 2005). However, to our knowledge, the application of such a technique to the pronoun resolution task still remains unexplored.

Compared with previous work, our approach has several advantages: (1) The approach utilizes the parse trees as a structured feature, which avoids the efforts of decoding the parse trees into a set of syntactic features in a heuristic manner. (2) The approach is able to put together the structured feature and the normal flat features in a trainable model, which allows different types of

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 41–48,
Sydney, July 2006. ©2006 Association for Computational Linguistics

information to be considered in combination for both learning and resolution. (3) The approach is applicable for practical pronoun resolution as the syntactic information can be automatically obtained from machine-generated parse trees. And our study shows that the approach works well under the commonly available parsers.

We evaluate our approach on the ACE data set. The experimental results over the different domains indicate that the structured syntactic feature incorporated with kernels can significantly improve the resolution performance (by 5%∼8% in the success rates), and is reliably effective for the pronoun resolution task.

The remainder of the paper is organized as follows. Section 2 gives some related work that utilizes the structured syntactic knowledge to do pronoun resolution. Section 3 introduces the framework for the pronoun resolution, as well as the baseline feature space and the SVM classifier. Section 4 presents in detail the structured feature and the kernel functions to incorporate such a feature in the resolution. Section 5 shows the experimental results and has some discussion. Finally, Section 6 concludes the paper.

2 Related Work

One of the early work on pronoun resolution relying on parse trees was proposed by Hobbs (1978). For a pronoun to be resolved, Hobbs' algorithm works by searching the parse trees of the current text. Specifically, the algorithm processes one sentence at a time, using a left-to-right breadth-first searching strategy. It first checks the current sentence where the pronoun occurs. The first NP that satisfies constraints, like number and gender agreements, would be selected as the antecedent. If the antecedent is not found in the current sentence, the algorithm would traverse the trees of previous sentences in the text. As the searching processing is completely done on the parse trees, the performance of the algorithm would rely heavily on the accuracy of the parsing results.

Lappin and Leass (1994) reported a pronoun resolution algorithm which uses the syntactic representation output by McCord's Slot Grammar parser. A set of salience measures (e.g. *Subject*, *Object* or *Accusative* emphasis) is derived from the syntactic structure. The candidate with the highest salience score would be selected as the antecedent. In their algorithm, the weights of salience measures have to be assigned manually.

Category: whether the candidate is a definite noun phrase, indefinite noun phrase, pronoun, named-entity or others.

Reflexiveness: whether the pronominal anaphor is a reflexive pronoun.

Type: whether the pronominal anaphor is a male-person pronoun (like *he*), female-person pronoun (like *she*), single gender-neuter pronoun (like *it*), or plural gender-neuter pronoun (like *they*)

Subject: whether the candidate is a subject of a sentence, a subject of a clause, or not.

Object: whether the candidate is an object of a verb, an object of a preposition, or not.

Distance: the sentence distance between the candidate and the pronominal anaphor.

Closeness: whether the candidate is the candidate closest to the pronominal anaphor.

FirstNP: whether the candidate is the first noun phrase in the current sentence.

Parallelism: whether the candidate has an identical collocation pattern with the pronominal anaphor.

Table 1: Feature set for the baseline pronoun resolution system

Luo and Zitouni (2005) proposed a coreference resolution approach which also explores the information from the syntactic parse trees. Different from Lappin and Leass (1994)'s algorithm, they employed a maximum entropy based model to automatically compute the importance (in terms of weights) of the features extracted from the trees. In their work, the selection of their features is mainly inspired by the government and binding theory, aiming to capture the c-command relationships between the pronoun and its antecedent candidate. By contrast, our approach simply utilizes the parse trees as a structured feature, and lets the learning algorithm discover all possible embedded information that is necessary for pronoun resolution.

3 The Resolution Framework

Our pronoun resolution system adopts the common learning-based framework similar to those by Soon et al. (2001) and Ng and Cardie (2002).

In the learning framework, a training or testing instance is formed by a pronoun and one of its antecedent candidate. During training, for each pronominal anaphor encountered, a positive instance is created by paring the anaphor and its closest antecedent. Also a set of negative instances is formed by paring the anaphor with each of the non-coreferential candidates. Based on the training instances, a binary classifier is generated using a particular learning algorithm. During resolution, a pronominal anaphor to be resolved is paired in turn with each preceding antecedent candidate to form a testing instance. This instance is presented to the classifier which then returns a class label with a confidence value indicating the likelihood that the candidate is the antecedent. The candidate with the highest confidence value will be selected as the antecedent of the pronominal anaphor.

3.1 Feature Space

As with many other learning-based approaches, the knowledge for the reference determination is represented as a set of features associated with the training or test instances. In our baseline system, the features adopted include lexical property, morphologic type, distance, salience, parallelism, grammatical role and so on. Listed in Table 1, all these features have been proved effective for pronoun resolution in previous work.

3.2 Support Vector Machine

In theory, any discriminative learning algorithm is applicable to learn the classifier for pronoun resolution. In our study, we use Support Vector Machine (Vapnik, 1995) to allow the use of kernels to incorporate the structured feature.

Suppose the training set *S* consists of labelled 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

$$f(x) = sgn(\sum_{i=1} y_i a_i x * x_i + b) \qquad (1)$$

where a_i is the learned parameter for a support vector x_i. An instance x is classified as positive (negative) if $f(x) > 0$ ($f(x) < 0$)[1].

One advantage of SVM is that we can use kernel methods to map a feature space to a particular high-dimension space, in case that the current problem could not be separated in a linear way. Thus the dot-product $x_1 * x_2$ is replaced by a kernel function (or kernel) between two vectors, that is $K(x_1, x_2)$. For the learning with the normal features listed in Table 1, we can just employ the well-known polynomial or radial basis kernels that can be computed efficiently. In the next section we will discuss how to use kernels to incorporate the more complex structured feature.

4 Incorporating Structured Syntactic Information

4.1 Main Idea

A parse tree that covers a pronoun and its antecedent candidate could provide us much syntactic information related to the pair. The commonly used syntactic knowledge for pronoun resolution, such as grammatical roles or the governing relations, can be directly described by the tree structure. Other syntactic knowledge that may be helpful for resolution could also be implicitly represented in the tree. Therefore, by comparing the common substructures 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 the tree kernel reflects the similarity between two instances in syntax. Such syntactic similarity can be further combined with other knowledge to compute the overall similarity between two instances, through a composite kernel. And thus a SVM classifier can be learned and then used for resolution. This is just the main idea of our approach.

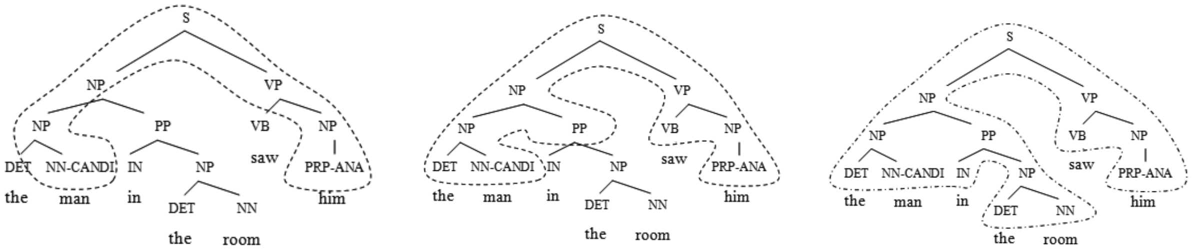
4.2 Structured Syntactic Feature

Normally, parsing is done on the sentence level. However, in many cases a pronoun and an 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. 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:

Min-Expansion This feature records the minimal structure covering both the pronoun and

[1] For our task, the result of $f(x)$ is used as the confidence value of the candidate to be the antecedent of the pronoun described by x.



Min-Expansion Simple-Expansion Full-Expansion

Figure 1: structured-features for the instance i{"him", "the man"}

the candidate in the parse tree. It only includes the nodes occurring in the shortest path connecting the pronoun and the candidate, via the nearest commonly commanding node. For example, considering the sentence "*The man in the room saw him.*", the structured feature for the instance i{"him","the man"} is circled with dash lines as shown in the leftmost picture of Figure 1.

Simple-Expansion *Min-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 *Min-Expansion*, but also includes the first-level children of these nodes[2]. The middle of Figure 1 shows such a feature for i{"him", "the man"}. We can see that the nodes "PP" (for "in the room") and "VB" (for "saw") are included in the feature, which provides clues that the candidate is modified by a prepositional phrase and the pronoun is the object of a verb.

Full-Expansion This feature focusses 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 candidate and the pronoun[3]. Such a feature keeps the most information related to the pronoun and candidate pair. The rightmost picture of Figure 1 shows the structure for feature *Full-Expansion* of i{"him", "the man"}. As illustrated, different from in *Simple-Expansion*, the subtree of "PP" (for "in the room") is fully expanded and all its children nodes are included in *Full-Expansion*.

Note that to distinguish from other words, we explicitly mark up in the structured feature the pronoun and the antecedent candidate under consideration, by appending a string tag "ANA" and "CANDI" in their respective nodes (e.g.,"NN-CANDI" for "man" and "PRP-ANA" for "him" as shown in Figure 1).

4.3 Structural 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 subtrees and use the number of common subtrees as the measure of the similarity between the trees. As has been proved, the convolution kernel can be efficiently computed in polynomial time.

The above tree kernel only aims for the structured feature. We also need a composite kernel to combine together the structured feature and the normal features described in Section 3.1. In our study we define the composite kernel as follows:

$$K_c(x_1, x_2) \quad = \quad \frac{K_n(x_1, x_2)}{|K_n(x_1, x_2)|} * \frac{K_t(x_1, x_2)}{|K_t(x_1, x2)|} \tag{2}$$

where K_t is the convolution tree kernel defined for the structured feature, and K_n is the kernel applied on the normal features. Both kernels are divided by their respective length[4] for normalization. The new composite kernel K_c, defined as the

[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.

[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 $|K(x_1, x_2)| = \sqrt{K(x_1, x_1) * K(x_2, x_2)}$

multiplier of normalized K_t and K_n, will return a value close to 1 only if both the structured features and the normal features from the two vectors have high similarity under their respective kernels.

5 Experiments and Discussions

5.1 Experimental Setup

In our study we focussed on the third-person pronominal anaphora resolution. All the experiments were done on the ACE-2 V1.0 corpus (NIST, 2003), which contain two data sets, training and devtest, used for training and testing respectively. Each of these sets is further divided into three domains: newswire (NWire), newspaper (NPaper), and broadcast news (BNews).

An input raw text was preprocessed automatically by a pipeline of NLP components, including sentence boundary detection, POS-tagging, Text Chunking and Named-Entity Recognition. The texts were parsed using the maximum-entropy-based Charniak parser (Charniak, 2000), based on which the structured features were computed automatically. For learning, the SVM-Light software (Joachims, 1999) was employed with the convolution tree kernel implemented by Moschitti (2004). All classifiers were trained with default learning parameters.

The performance was evaluated based on the metric *success*, the ratio of the number of correctly resolved[5] anaphor over the number of all anaphors. For each anaphor, the NPs occurring within the current and previous two sentences were taken as the initial antecedent candidates. Those with mismatched number and gender agreements were filtered from the candidate set. Also, pronouns or NEs that disagreed in person with the anaphor were removed in advance. For training, there were 1207, 1440, and 1260 pronouns with non-empty candidate set found pronouns in the three domains respectively, while for testing, the number was 313, 399 and 271. On average, a pronoun anaphor had 6∼9 antecedent candidates ahead. Totally, we got around 10k, 13k and 8k training instances for the three domains.

5.2 Baseline Systems

Table 2 lists the performance of different systems. We first tested Hobbs' algorithm (Hobbs, 1978). Described in Section 2, the algorithm uses heuristic rules to search the parse tree for the antecedent, and will act as a good baseline to compare with the learned-based approach with the structured feature. As shown in the first line of Table 2, Hobbs' algorithm obtains 66%∼72% success rates on the three domains.

	NWire	NPaper	BNews
Hobbs (1978)	66.1	66.4	72.7
NORM	74.4	77.4	74.2
NORM_MaxEnt	72.8	77.9	75.3
NORM_C5	71.9	75.9	71.6
S_Min	76.4	81.0	76.8
S_Simple	73.2	82.7	82.3
S_Full	73.2	80.5	79.0
NORM+S_Min	77.6	82.5	**82.3**
NORM+S_Simple	79.2	82.7	**82.3**
NORM+S_Full	**81.5**	**83.2**	81.5

Table 2: Results of the syntactic structured features

The second block of Table 2 shows the baseline system (NORM) that uses only the normal features listed in Table 1. Throughout our experiments, we applied the polynomial kernel on the normal features to learn the SVM classifiers. In the table we also compared the SVM-based results with those using other learning algorithms, i.e., Maximum Entropy (Maxent) and C5 decision tree, which are more commonly used in the anaphora resolution task.

As shown in the table, the system with normal features (NORM) obtains 74%∼77% success rates for the three domains. The performance is similar to other published results like those by Keller and Lapata (2003), who adopted a similar feature set and reported around 75% success rates on the ACE data set. The comparison between different learning algorithms indicates that SVM can work as well as or even better than Maxent (NORM_MaxEnt) or C5 (NORM_C5).

5.3 Systems with Structured Features

The last two blocks of Table 2 summarize the results using the three syntactic structured features, i.e, *Min_Expansion* (S_MIN), *Simple_Expansion* (S_SIMPLE) and *Full_Expansion* (S_FULL). Between them, the third block is for the systems using the individual structured feature alone. We can see that all the three structured features per-

[5] An anaphor was deemed correctly resolved if the found antecedent is in the same coreference chain of the anaphor.

	NWire			NPaper			BNews		
Sentence Distance	0	1	2	0	1	2	0	1	2
(Number of Prons)	(192)	(102)	(19)	(237)	(147)	(15)	(175)	(82)	(14)
NORM	80.2	72.5	26.3	81.4	75.5	33.3	80.0	65.9	50.0
S_Simple	79.7	70.6	21.1	**87.3**	**81.0**	26.7	**89.7**	70.7	**57.1**
NORM+S_Simple	**85.4**	**76.5**	**31.6**	**87.3**	79.6	**40.0**	88.6	**74.4**	50.0

Table 3: The resolution results for pronouns with antecedent in different sentences apart

	NWire		NPaper		BNews	
Type	person	neuter	person	neuter	person	neuter
(Number of Prons)	(171)	(142)	(250)	(149)	(153)	(118)
NORM	81.9	65.5	80.0	73.2	74.5	73.7
S_Simple	81.9	62.7	83.2	**81.9**	82.4	**82.2**
NORM+S_Simple	**87.1**	**69.7**	**83.6**	81.2	**86.9**	76.3

Table 4: The resolution results for different types of pronouns

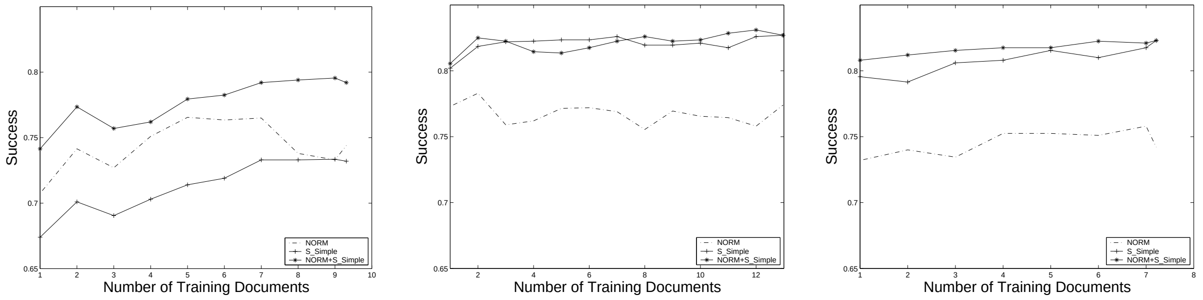
form better than the normal features for NPaper (up to 5.3% *success*) and BNews (up to 8.1% *success*), or equally well ($\pm 1 \sim 2\%$ in *success*) for NWire. When used together with the normal features, as shown in the last block, the three structured features all outperform the baselines. Especially, the combinations of NORM+S_SIMPLE and NORM+S_FULL can achieve significantly[6] better results than NORM, with the success rate increasing by (4.8%, 5.3% and 8.1%) and (7.1%, 5.8%, 7.2%) respectively. All these results prove that the structured syntactic feature is effective for pronoun resolution.

We further compare the performance of the three different structured features. As shown in Table 2, when used together with the normal features, *Full_Expansion* gives the highest success rates in NWire and NPaper, but nevertheless the lowest in BNews. This should be because feature *Full-Expansion* captures a larger portion of the parse trees, and thus can provide more syntactic information than *Min_Expansion* or *Simple_Expansion*. However, if the texts are less-formally structured as those in BNews, *Full-Expansion* would inevitably involve more noises and thus adversely affect the resolution performance. By contrast, feature *Simple_Expansion* would achieve balance between the information and the noises to be introduced: from Table 2 we can find that compared with the other two features, *Simple_Expansion* is capable of producing average results for all the three domains. And for this reason, our subsequent reports will focus on *Simple_Expansion*, unless otherwise specified.

As described, to compute the structured feature, parse trees for different sentences are connected to form a large tree for the text. It would be interesting to find how the structured feature works for pronouns whose antecedents reside in different sentences. For this purpose we tested the success rates for the pronouns with the closest antecedent occurring in the same sentence, one-sentence apart, and two-sentence apart. Table 3 compares the learning systems with/without the structured feature present. From the table, for all the systems, the success rates drop with the increase of the distances between the pronoun and the antecedent. However, in most cases, adding the structured feature would bring consistent improvement against the baselines regardless of the number of sentence distance. This observation suggests that the structured syntactic information is helpful for both intra-sentential and inter-sentential pronoun resolution.

We were also concerned about how the structured feature works for different types of pronouns. Table 4 lists the resolution results for two types of pronouns: person pronouns (i.e., “he”, “she”) and neuter-gender pronouns (i.e., “it” and “they”). As shown, with the structured feature incorporated, the system NORM+S_Simple can significantly boost the performance of the baseline (NORM), for both personal pronoun and neuter-gender pronoun resolution.

[6] $p < 0.05$ by a 2-tailed *t* test.



NWire NPaper BNews

Figure 2: Learning curves of systems with different features

5.4 Learning Curves

Figure 2 plots the learning curves for the systems with three feature sets, i.e, normal features (NORM), structured feature alone (S_Simple), and combined features (NORM+S_Simple). We trained each system with different number of instances from 1k, 2k, 3k, . . . , till the full size. Each point in the figures was the average over two trails with instances selected forwards and backwards respectively. From the figures we can find that (1) Used in combination (NORM+S_Simple), the structured feature shows superiority over NORM, achieving results consistently better than the normal features (NORM) do in all the three domains. (2) With training instances above 3k, the structured feature, used either in isolation (S_Simple) or in combination (NORM+S_Simple), leads to steady increase in the success rates and exhibit smoother learning curves than the normal features (NORM). These observations further prove the reliability of the structured feature in pronoun resolution.

5.5 Feature Analysis

In our experiment we were also interested to compare the structured feature with the normal flat features extracted from the parse tree, like feature *Subject* and *Object*. For this purpose we took out these two grammatical features from the normal feature set, and then trained the systems again. As shown in Table 5, the two grammatical-role features are important for the pronoun resolution: removing these features results in up to 5.7% (NWire) decrease in *success*. However, when the structured feature is included, the loss in *success* reduces to 1.9% and 1.1% for NWire and BNews, and a slight improvement can even be achieved for NPaper. This indicates that the structured feature can effectively provide the syntactic information important for pronoun resolution.

	NWire	NPaper	BNews
NORM	74.4	77.4	74.2
NORM - subj/obj	68.7	76.2	72.7
NORM + S_Simple	79.2	82.7	82.3
NORM + S_Simple - subj/obj	77.3	83.0	81.2
NORM + Luo05	75.7	77.9	74.9

Table 5: Comparison of the structured feature and the flat features extracted from parse trees

Feature	Parser	NWire	NPaper	BNews
S_Simple	Charniak00	73.2	82.7	82.3
	Collins99	75.1	83.2	80.4
NORM+ S_Simple	Charniak00	79.2	82.7	82.3
	Collins99	80.8	81.5	82.3

Table 6: Results using different parsers

We also tested the flat syntactic feature set proposed in Luo and Zitouni (2005)'s work. As described in Section 2, the feature set is inspired the binding theory, including those features like whether the candidate is c_commanding the pronoun, and the counts of "NP", "VP", "S" nodes in the commanding path. The last line of Table 5 shows the results by adding these features into the normal feature set. In line with the reports in (Luo and Zitouni, 2005) we do observe the performance improvement against the baseline (NORM) for all the domains. However, the increase in the success rates (up to 1.3%) is not so large as by adding the structured feature (NORM+S_Simple) instead.

5.6 Comparison with Different Parsers

As mentioned, the above reported results were based on Charniak (2000)'s parser. It would be interesting to examine the influence of different parsers on the resolution performance. For this purpose, we also tried the parser by Collins (1999)

(Mode II)[7], and the results are shown in Table 6. We can see that Charniak (2000)'s parser leads to higher success rates for NPaper and BNews, while Collins (1999)'s achieves better results for NWire. However, the difference between the results of the two parsers is not significant (less than 2% *success*) for the three domains, no matter whether the structured feature is used alone or in combination.

6 Conclusion

The purpose of this paper is to explore how to make use of the structured syntactic knowledge to do pronoun resolution. Traditionally, syntactic information from parse trees is represented as a set of flat features. However, the features are usually selected and defined by heuristics and may not necessarily capture all the syntactic information provided by the parse trees. In the paper, we propose a kernel-based method to incorporate the information from parse trees. Specifically, we directly utilize the syntactic parse tree as a structured feature, and then apply kernels to such a feature, together with other normal features, to learn the decision classifier and do the resolution. Our experimental results on ACE data set show that the system with the structured feature included can achieve significant increase in the success rate by around 5%∼8%, for all the different domains. The deeper analysis on various factors like training size, feature set or parsers further proves that the structured feature incorporated with our kernel-based method is reliably effective for the pronoun resolution task.

References

C. Aone and S. W. Bennett. 1995. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of the 33rd Annual Meeting of the Association for Compuational Linguistics*, pages 122–129.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of North American chapter of the Association for Computational Linguistics annual meeting*, pages 132–139.

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)*, pages 263–270.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

J. Hobbs. 1978. Resolving pronoun references. *Lingua*, 44:339–352.

T. Joachims. 1999. Making large-scale svm learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

F. Keller and M. Lapata. 2003. Using the web to obtain freqencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

C. Kennedy and B. Boguraev. 1996. Anaphora for everyone: pronominal anaphra resolution without a parser. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 113–118, Copenhagen, Denmark.

S. Lappin and H. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):525–561.

X. Luo and I. Zitouni. 2005. Milti-lingual coreference resolution with syntactic features. In *Proceedings of Human Language Techonology conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 660–667.

R. Mitkov. 1998. Robust pronoun resolution with limited knowledge. In *Proceedings of the 17th Int. Conference on Computational Linguistics*, pages 869–875.

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)*, pages 335–342.

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Philadelphia.

W. Soon, H. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

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*, pages 127–134, Barcelona.

D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3(6):1083 – 1106.

S. Zhao and R. Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of 43rd Annual Meeting of the Association for Computational Linguistics (ACL05)*, pages 419–426.

[7] As in their pulic reports on Section 23 of WSJ TreeBank, Charniak (2000)'s parser achieves 89.6% recall and 89.5% precision with 0.88 crossing brackets (words $\leq$ 100), against Collins (1999)'s 88.1% recall and 88.3% precision with 1.06 crossing brackets.

A Finite-State Model of Human Sentence Processing

Jihyun Park and Chris Brew
Department of Linguisitcs
The Ohio State University
Columbus, OH, USA
{park|cbrew}@ling.ohio-state.edu

Abstract

It has previously been assumed in the psycholinguistic literature that finite-state models of language are crucially limited in their explanatory power by the locality of the probability distribution and the narrow scope of information used by the model. We show that a simple computational model (a bigram part-of-speech tagger based on the design used by Corley and Crocker (2000)) makes correct predictions on processing difficulty observed in a wide range of empirical sentence processing data. We use two modes of evaluation: one that relies on comparison with a control sentence, paralleling practice in human studies; another that measures probability drop in the disambiguating region of the sentence. Both are surprisingly good indicators of the processing difficulty of garden-path sentences. The sentences tested are drawn from published sources and systematically explore five different types of ambiguity: previous studies have been narrower in scope and smaller in scale. We do not deny the limitations of finite-state models, but argue that our results show that their usefulness has been underestimated.

1 Introduction

The main purpose of the current study is to investigate the extent to which a probabilistic part-of-speech (POS) tagger can correctly model human sentence processing data. Syntactically ambiguous sentences have been studied in great depth in psycholinguistics because the pattern of ambiguity resolution provides a window onto the human sentence processing mechanism (HSPM). *Prima facie* it seems unlikely that such a tagger will be adequate, because almost all previous researchers have assumed, following standard linguistic theory, that a formally adequate account of recursive syntactic structure is an essential component of any model of the behaviour. In this study, we tested a bigram POS tagger on different types of structural ambiguities and (as a sanity check) to the well-known asymmetry of subject and object relative clause processing.

Theoretically, the garden-path effect is defined as processing difficulty caused by reanalysis. Empirically, it is attested as comparatively slower reading time or longer eye fixation at a disambiguating region in an ambiguous sentence compared to its control sentences (Frazier and Rayner, 1982; Trueswell, 1996). That is, the garden-path effect detected in many human studies, in fact, is measured through a "comparative" method.

This characteristic of the sentence processing research design is reconstructed in the current study using a probabilistic POS tagging system. Under the assumption that larger probability decrease indicates slower reading time, the test results suggest that the probabilistic POS tagging system can predict reading time penalties at the disambiguating region of garden-path sentences compared to that of non-garden-path sentences (i.e. control sentences).

2 Previous Work

Corley and Crocker (2000) present a probabilistic model of lexical category disambiguation based on a bigram statistical POS tagger. Kim et al. (2002) suggest the feasibility of modeling human syntactic processing as lexical ambiguity resolution using a syntactic tagging system called Super-Tagger

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 49–56,
Sydney, July 2006. ©2006 Association for Computational Linguistics

(Joshi and Srinivas, 1994; Bangalore and Joshi, 1999). Probabilistic parsing techniques also have been used for sentence processing modeling (Jurafsky, 1996; Narayanan and Jurafsky, 2002; Hale, 2001; Crocker and Brants, 2000). Jurafsky (1996) proposed a probabilistic model of HSPM using a parallel beam-search parsing technique based on the stochastic context-free grammar (SCFG) and subcategorization probabilities. Crocker and Brants (2000) used broad coverage statistical parsing techniques in their modeling of human syntactic parsing. Hale (2001) reported that a probabilistic Earley parser can make correct predictions of garden-path effects and the subject/object relative asymmetry. These previous studies have used small numbers of examples of, for example, the Reduced-relative clause ambiguity and the Direct-Object/Sentential-Complement ambiguity.

The current study is closest in spirit to a previous attempt to use the technology of part-of-speech tagging (Corley and Crocker, 2000). Among the computational models of the HSPM mentioned above, theirs is the simplest. They tested a statistical bigram POS tagger on lexically ambiguous sentences to investigate whether the POS tagger correctly predicted reading-time penalty. When a previously preferred POS sequence is less favored later, the tagger makes a repair. They claimed that the tagger's reanalysis can model the processing difficulty in human's disambiguating lexical categories when there exists a discrepancy between lexical bias and resolution.

3 Experiments

In the current study, Corley and Crocker's model is further tested on a wider range of so-called structural ambiguity types. A Hidden Markov Model POS tagger based on bigrams was used. We made our own implementation to be sure of getting as close as possible to the design of Corley and Crocker (2000). Given a word string, $w_0, w_1, \cdots, w_n$, the tagger calculates the probability of every possible tag path, $t_0, \cdots, t_n$. Under the Markov assumption, the joint probability of the given word sequence and each possible POS sequence can be approximated as a product of conditional probability and transition probability as shown in (1).

(1) $P(w_0, w_1, \cdots, w_n, t_0, t_1, \cdots, t_n)$

$\approx \Pi_{i=1}^{n} P(w_i|t_i) \cdot P(t_i|t_{i-1})$, where $n \geq 1$.

Using the Viterbi algorithm (Viterbi, 1967), the tagger finds the most likely POS sequence for a given word string as shown in (2).

(2) $\arg\max P(t_0, t_1, \cdots, t_n | w_0, w_1, \cdots, w_n, \mu)$.

This is known technology, see Manning and Schütze (1999), but the particular use we make of it is unusual. The tagger takes a word string as an input, outputs the most likely POS sequence and the final probability. Additionally, it presents accumulated probability at each word break and probability re-ranking, if any. Note that the running probability at the beginning of a sentence will be 1, and will keep decreasing at each word break since it is a product of conditional probabilities.

We tested the predictability of the model on empirical reading data with the probability decrease and the presence or absence of probability re-ranking. Adopting the standard experimental design used in human sentence processing studies, where word-by-word reading time or eye-fixation time is compared between an experimental sentence and its control sentence, this study compares probability at each word break between a pair of sentences. Comparatively faster or larger drop of probability is expected to be a good indicator of comparative processing difficulty. Probability re-ranking, which is a simplified model of the reanalysis process assumed in many human studies, is also tested as another indicator of garden-path effect. Given a word string, all the possible POS sequences compete with each other based on their probability. Probability re-ranking occurs when an initially dispreferred POS sub-sequence becomes the preferred candidate later in the parse, because it fits in better with later words.

The model parameters, $P(w_i|t_i)$ and $P(t_i|t_{i-1})$, are estimated from a small section (970,995 tokens,47,831 distinct words) of the British National Corpus (BNC), which is a 100 million-word collection of British English, both written and spoken, developed by Oxford University Press (Burnard, 1995). The BNC was chosen for training the model because it is a POS-annotated corpus, which allows supervised training. In the implementation we use log probabilities to avoid underflow, and we report log probabilities in the sequel.

3.1 Hypotheses

If the HSPM is affected by frequency information, we can assume that it will be easier to process events with higher frequency or probability compared to those with lower frequency or probability. Under this general assumption, the overall difficulty of a sentence is expected to be measured or predicted by the mean size of probability decrease. That is, probability will drop faster in garden-path sentences than in control sentences (e.g. unambiguous sentences or ambiguous but non-garden-path sentences).

More importantly, the probability decrease pattern at disambiguating regions will predict the trends in the reading time data. All other things being equal, we might expect a reading time penalty when the size of the probability decrease at the disambiguating region in garden-path sentences is greater compared to the control sentences. This is a simple and intuitive assumption that can be easily tested. We could have formed the sum over all possible POS sequences in association with the word strings, but for the present study we simply used the Viterbi path: justifying this because this is the best single-path approximation to the joint probability.

Lastly, re-ranking of POS sequences is expected to predict reanalysis of lexical categories. This is because re-ranking in the tagger is parallel to reanalysis in human subjects, which is known to be cognitively costly.

3.2 Materials

In this study, five different types of ambiguity were tested including Lexical Category ambiguity, Reduced Relative ambiguity (RR ambiguity), Prepositional Phrase Attachment ambiguity (PP ambiguity), Direct-Object/Sentential-Complement ambiguity (DO/SC ambiguity), and Clausal Boundary ambiguity. The following are example sentences for each ambiguity type, shown with the ambiguous region italicized and the disambiguating region bolded. All of the example sentences are garden-path sentneces.

(3) Lexical Category ambiguity
The foreman knows that the warehouse *prices* **the** beer very modestly.

(4) RR ambiguity
The horse *raced* past the barn **fell**.

(5) PP ambiguity
Katie laid the dress *on the floor* **onto** the bed.

(6) DO/SC ambiguity
He forgot Pam **needed** a ride with him.

(7) Clausal Boundary ambiguity
Though George kept on reading *the story* really **bothered** him.

There are two types of control sentences: unambiguous sentences and ambiguous but non-garden-path sentences as shown in the examples below. Again, the ambiguous region is italicized and the disambiguating region is bolded.

(8) Garden-Path Sentence
The horse *raced* past the barn **fell**.

(9) Ambiguous but Non-Garden-Path Control
The horse *raced* past the barn **and** fell.

(10) Unambiguous Control
The horse that was raced past the barn fell.

Note that the garden-path sentence (8) and its ambiguous control sentence (9) share exactly the same word sequence except for the disambiguating region. This allows direct comparison of probability at the critical region (i.e. disambiguating region) between the two sentences. Test materials used in experimental studies are constructed in this way in order to control extraneous variables such as word frequency. We use these sentences in the same form as the experimentalists so we inherit their careful design.

In this study, a total of 76 sentences were tested: 10 for lexical category ambiguity, 12 for RR ambiguity, 20 for PP ambiguity, 16 for DO/SC ambiguity, and 18 for clausal boundary ambiguity. This set of materials is, to our knowledge, the most comprehensive yet subjected to this type of study. The sentences are directly adopted from various psycholinguistic studies (Frazier, 1978; Trueswell, 1996; Frazier and Clifton, 1996; Ferreira and Clifton, 1986; Ferreira and Henderson, 1986).

As a baseline test case of the tagger, the well-established asymmetry between subject- and object-relative clauses was tested as shown in (11).

(11) a. The editor who kicked the writer fired the entire staff. (Subject-relative)

b. The editor who the writer kicked fired the entire staff. (Object-relative)

The reading time advantage of subject-relative clauses over object-relative clauses is robust in English (Traxler et al., 2002) as well as other languages (Mak et al., 2002; Homes et al., 1981). For this test, materials from Traxler et al. (2002) (96 sentences) are used.

4 Results

4.1 The Probability Decrease per Word

Unambiguous sentences are usually longer than garden-path sentences. To compare sentences of different lengths, the joint probability of the whole sentence and tags was divided by the number of words in the sentence. The result showed that the average probability decrease was greater in garden-path sentences compared to their unambiguous control sentences. This indicates that garden-path sentences are more difficult than unambiguous sentences, which is consistent with empirical findings.

Probability decreased faster in object-relative sentences than in subject relatives as predicted. In the psycholinguistics literature, the comparative difficulty of object-relative clauses has been explained in terms of verbal working memory (King and Just, 1991), distance between the gap and the filler (Bever and McElree, 1988), or perspective shifting (MacWhinney, 1982). However, the test results in this study provide a simpler account for the effect. That is, the comparative difficulty of an object-relative clause might be attributed to its less frequent POS sequence. This account is particularly convincing since each pair of sentences in the experiment share the exactly same set of words except their order.

4.2 Probability Decrease at the Disambiguating Region

A total of 30 pairs of a garden-path sentence and its ambiguous, non-garden-path control were tested for a comparison of the probability decrease at the disambiguating region. In 80% of the cases, the probability drops more sharply in garden-path sentences than in control sentences at the critical word. The test results are presented in (12) with the number of test sets for each ambiguous type and the number of cases where the model correctly predicted reading-time penalty of garden-path sentences.

(12) Ambiguity Type (Correct Predictions/Test Sets)
a. Lexical Category Ambiguity (4/4)
b. PP Ambiguity (10/10)
c. RR Ambiguity (3/4)
d. DO/SC Ambiguity (4/6)
e. Clausal Boundary Ambiguity (3/6)

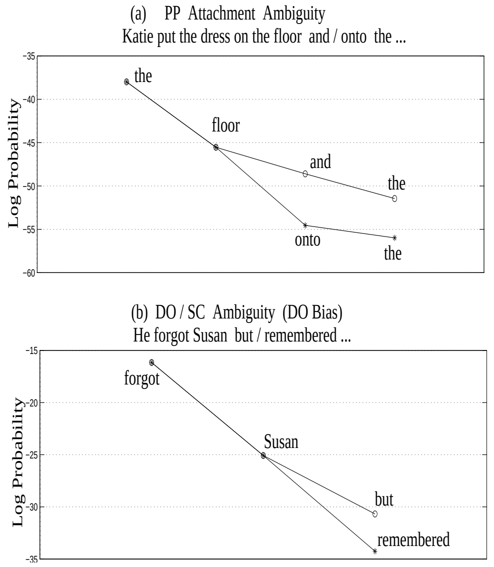


Figure 1: Probability Transition (Garden-Path vs. Non Garden-Path)

(a) $-\circ-$: Non-Garden-Path (Adjunct PP), $-\ast-$: Garden -Path (Complement PP)
(b) $-\circ-$: Non-Garden-Path (DO-Biased, DO-Resolved), $-\ast-$: Garden-Path (DO-Biased, SC-Resolved)

The two graphs in Figure 1 illustrate the comparison of probability decrease between a pair of sentence. The *y*-axis of both graphs in Figure 1 is log probability. The first graph compares the probability drop for the prepositional phrase (PP) attachment ambiguity (*Katie put the dress on the floor and/onto the bed....*) The empirical result for this type of ambiguity shows that reading time penalty is observed when the second PP, *onto the bed*, is introduced, and there is no such effect for the other sentence. Indeed, the sharper probability drop indicates that the additional PP is less likely, which makes a prediction of a comparative processing difficulty. The second graph exhibits the probability comparison for the DO/SC ambiguity. The verb *forget* is a DO-biased verb and thus processing difficulty is observed when it has a sentential complement. Again, this effect was replicated here.

The results showed that the disambiguating word given the previous context is more difficult in garden-path sentences compared to control sentences. There are two possible explanations for the processing difficulty. One is that the POS sequence of a garden-path sentence is less probable than that of its control sentence. The other account is that the disambiguating word in a garden-path sentence is a lower frequency word compared to that of its control sentence.

For example, slower reading time was observed in (13a) and (14a) compared to (13b) and (14b) at the disambiguating region that is bolded.

(13) Different POS at the Disambiguating Region

a. Katie laid the dress *on the floor* **onto** (-57.80) the bed.

b. Katie laid the dress *on the floor* **after** (-55.77) her mother yelled at her.

(14) Same POS at the Disambiguating Region

a. The umpire helped the child ***on*** (-42.77) third base.

b. The umpire helped the child ***to*** (-42.23) third base.

The log probability for each disambiguating word is given at the end of each sentence. As expected, the probability at the disambiguating region in (13a) and (14a) is lower than in (13b) and (14b) respectively. The disambiguating words in (13) have different POS's; Preposition in (13a) and Conjunction (13b). This suggests that the probabilities of different POS sequences can account for different reading time at the region. In (14), however, both disambiguating words are the same POS (i.e. Preposition) and the POS sequences for both sentences are identical. Instead, "on" and "to", have different frequencies and this information is reflected in the conditional probability $P(word_i|state)$. Therefore, the slower reading time in (14b) might be attributable to the lower frequency of the disambiguating word, "to" compared to "on".

4.3 Probability Re-ranking

The probability re-ranking reported in Corley and Crocker (2000) was replicated. The tagger successfully resolved the ambiguity by reanalysis when the ambiguous word was immediately followed by the disambiguating word (e.g. Without *her* **he** was lost.). If the disambiguating word did not immediately follow the ambiguous region, (e.g. Without *her* contributions **would** be very inadequate.) the ambiguity is sometimes incorrectly resolved.

When revision occurred, probability dropped more sharply at the revision point and at the disambiguation region compared to the control sentences. When the ambiguity was not correctly resolved, the probability comparison correctly modeled the comparative difficulty of the garden-path sentences

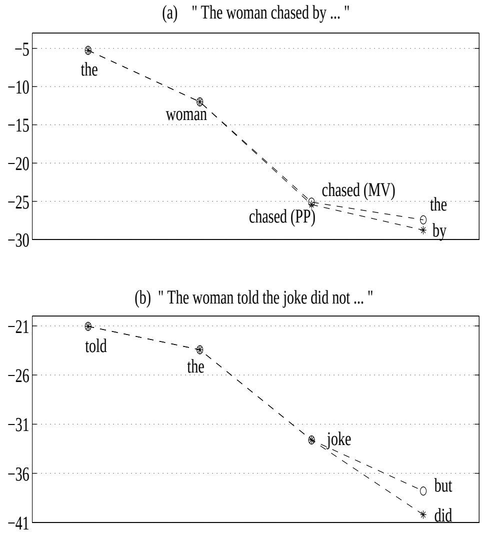


Figure 2: Probability Transition in the RR Ambiguity

(a) $-\circ-$: Non-Garden-Path (Past Tense Verb), $-*-$: Garden-Path (Past Participle)
(b) $-\circ-$: Non-Garden-Path (Past Tense Verb), $-*-$: Garden-Path, (Past Participle)

Of particular interest in this study is RR ambiguity resolution. The tagger predicted the processing difficulty of the RR ambiguity with probability re-ranking. That is, the tagger initially favors the main-verb interpretation for the ambiguous *-ed* form, and later it makes a repair when the ambiguity is resolved as a past-participle.

In the first graph of Figure 2, "chased" is resolved as a past participle also with a revision since the disambiguating word "by" is immediately following. When revision occurred, probability dropped more sharply at the revision point and at the disambiguation region compared to the control sentences. When the disambiguating word is not immediately followed by the ambiguous word as in the second graph of Figure 2, the ambiguity was not resolved correctly, but the probababiltiy decrease at the disambiguating regions correctly predict that the garden-path sentence would be harder.

The RR ambiguity is often categorized as a syntactic ambiguity, but the results suggest that the ambiguity can be resolved locally and its processing difficulty can be detected by a finite state model. This suggests that we should be cautious in assuming that a structural explanation is needed for the RR ambiguity resolution, and it could be that similar cautions are in order for other ambiguities usually seen as syntactic.

Although the probability re-ranking reported in the previous studies (Corley and Crocker, 2000; Frazier, 1978) is correctly replicated, the tagger sometimes made undesired revisions. For example, the tagger did not make a repair for the sentence *The friend accepted by the man was very impressed* (Trueswell, 1996) because *accepted* is biased as a past participle. This result is compatible with the findings of Trueswell (1996). However, the bias towards past-participle produces a repair in the control sentence, which is unexpected. For the sentence, *The friend accepted the man who was very impressed*, the tagger showed a repair since it initially preferred a past-participle analysis for *accepted* and later it had to reanalyze. This is a limitation of our model, and does not match any previous empirical finding.

5 Discussion

The current study explores Corley and Crocker's model(2000) further on the model's account of human sentence processing data seen in empirical studies. Although there have been studies on a POS tagger evaluating it as a potential cognitive module of lexical category disambiguation, there has been little work that tests it as a modeling tool of syntactically ambiguous sentence processing.

The findings here suggest that a statistical POS tagging system is more informative than Crocker and Corley demonstrated. It has a predictive power of processing delay not only for lexically ambiguous sentences but also for structurally garden-pathed sentences. This model is attractive since it is computationally simpler and requires few statistical parameters. More importantly, it is clearly defined what predictions can be and cannot be made by this model. This allows systematic testability and refutability of the model unlike some other probabilistic frameworks. Also, the model training and testing is transparent and observable, and true probability rather than transformed weights are used, all of which makes it easy to understand the mechanism of the proposed model.

Although the model we used in the current study is not a novelty, the current work largely differs from the previous study in its scope of data used and the interpretation of the model for human sentence processing. Corley and Crocker clearly state that their model is strictly limited to lexical ambiguity resolution, and their test of the model was bounded to the noun-verb ambiguity. However, the findings in the current study play out differently. The experiments conducted in this study are parallel to empirical studies with regard to the design of experimental method and the test material. The garden-path sentences used in this study are authentic, most of them are selected from the cited literature, not conveniently coined by the authors. The word-by-word probability comparison between garden-path sentences and their controls is parallel to the experimental design widely adopted in empirical studies in the form of region-by-region reading or eye-gaze time comparison. In the word-by-word probability comparison, the model is tested whether or not it correctly predicts the comparative processing difficulty at the garden-path region. Contrary to the major claim made in previous empirical studies, which is that the garden-path phenomena are either modeled by syntactic principles or by structural frequency, the findings here show that the same phenomena can be predicted without such structural information.

Therefore, the work is neither a mere extended application of Corley and Crocker's work to a broader range of data, nor does it simply confirm earlier observations that finite state machines might accurately account for psycholinguistic results to some degree. The current study provides more concrete answers to what finite state machine is relevant to what kinds of processing difficulty and to what extent.

6 Future Work

Even though comparative analysis is a widely adopted research design in experimental studies, a sound scientific model should be independent of this comparative nature and should be able to make systematic predictions. Currently, probability re-ranking is one way to make systematic module-internal predictions about the garden-path effect. This brings up the issue of encoding more information in lexical entries and increasing ambiguity so that other ambiguity types also can be disambiguated in a similar way via lexical category disambiguation. This idea has been explored as one of the lexicalist approaches to sentence processing (Kim et al., 2002; Bangalore and Joshi, 1999).

Kim et al. (2002) suggest the feasibility of modeling structural analysis as lexical ambiguity resolution. They developed a connectionist neural network model of word recognition, which takes orthographic information, semantic information, and the previous two words as its input and outputs a SuperTag for the current word. A SuperTag is an elementary syntactic tree, or simply a structural description composed of features like POS, the number of complements, category of each complement, and the position of complements. In their view, structural disambiguation is simply another type of lexical category disambiguation, i.e. SuperTag disambiguation. When applied to DO/SC ambiguous fragments, such as "The economist decided ...", their model showed a general bias toward the NP-complement structure. This NP-complement bias was overcome by lexical information from high-frequency S-biased verbs, meaning that if the S-biased verb was a high frequency word, it was correctly tagged, but if the verb had low frequency, then it was more likely to be tagged as NP-complement verb. This result is also reported in other constraint-based model studies (e.g. Juliano and Tanenhaus (1994)), but the difference between the previous constraint-based studies and Kim et. al is that the result of the latter is based on training of the model on noisier data (sentences that were not tailored to the specific research purpose). The implementation of SuperTag advances the formal specification of the constraint-based lexicalist theory. However, the scope of their sentence processing model is limited to the DO/SC ambiguity, and the description of their model is not clear. In addition, their model is far beyond a simple statistical model: the interaction of different sources of information is not transparent. Nevertheless, Kim et al. (2002) provides a future direction for the current study and a starting point for considering what information should be included in the lexicon.

The fundamental goal of the current research is to explore a model that takes the most restrictive position on the size of parameters until additional parameters are demanded by data. Equally important, the quality of architectural simplicity should be maintained. Among the different sources of information manipulated by Kim et. al., the so-called elementary structural information is considered as a reasonable and ideal parameter for addition to the current model. The implementation and the evaluation of the model will be exactly the same as a statistical POS tagger provided with a large parsed corpus from which elementary trees can be extracted.

7 Conclusion

Our studies show that, at least for the sample of test materials that we culled from the standard literature, a statistical POS tagging system can predict processing difficulty in structurally ambiguous garden-path sentences. The statistical POS tagger was surprisingly effective in modeling sentence processing data, given the locality of the probability distribution. The findings in this study provide an alternative account for the garden-path effect observed in empirical studies, specifically, that the slower processing times associated with garden-path sentences are due in part to their relatively unlikely POS sequences in comparison with those of non-garden-path sentences and in part to differences in the emission probabilities that the tagger learns. One attractive future direction is to carry out simulations that compare the evolution of probabilities in the tagger with that in a theoretically more powerful model trained on the same data, such as an incremental statistical parser (Kim et al., 2002; Roark, 2001). In so doing we can find the places where the prediction problem faced both by the HSPM and the machines that aspire to emulate it actually warrants the greater power of structurally sensitive models, using this knowledge to mine large corpora for future experiments with human subjects.

We have not necessarily cast doubt on the hypothesis that the HSPM makes crucial use of structural information, but we have demonstrated that much of the relevant behavior can be captured in a simple model. The 'structural' regularities that we observe are reasonably well encoded into this model. For purposes of initial real-time processing it could be that the HSPM is using a similar encoding of structural regularities into convenient probabilistic or neural form. It is as yet unclear what the final form of a cognitively accurate model along these lines would be, but it is clear from our study that it is worthwhile, for the sake of clarity and explicit testability, to consider models that are simpler and more precisely specified than those assumed by dominant theories of human sentence processing.

Acknowledgments

This project was supported by the Cognitive Science Summer 2004 Research Award at the Ohio State University. We acknowledge support from NSF grant IIS 0347799.

References

S. Bangalore and A. K. Joshi. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2):237–266, 1999.

T. G. Bever and B. McElree. Empty categories access their antecedents during comprehension. *Linguistic Inquiry*, 19:35–43, 1988.

L Burnard. *Users Guide for the British National Corpus*. British National Corpus Consortium, Oxford University Computing Service, 1995.

S. Corley and M. W Crocker. *The Modular Statistical Hypothesis: Exploring Lexical Category Ambiguity*. Architectures and Mechanisms for Language Processing, M. Crocker, M. Pickering. and C. Charles (Eds.) Cambridge University Press, 2000.

W. C. Crocker and T. Brants. Wide-coverage probabilistic sentence processing, 2000.

F. Ferreira and C. Clifton. The independence of syntactic processing. *Journal of Memory and Language*, 25:348–368, 1986.

F. Ferreira and J. Henderson. Use of verb information in syntactic parsing: Evidence from eye movements and word-by-word self-paced reading. *Journal of Experimental Psychology*, 16: 555–568, 1986.

L. Frazier. On comprehending sentences: Syntactic parsing strategies. *Ph.D. dissertation, University of Massachusetts*, Amherst, MA, 1978.

L. Frazier and C. Clifton. *Construal*. Cambridge, MA: MIT Press, 1996.

L. Frazier and K. Rayner. Making and correcting errors during sentence comprehension: Eye movements in the analysis of structurally ambiguous sentences. *Cognitive Psychology*, 14: 178–210, 1982.

J. Hale. A probabilistic earley parser as a psycholinguistic model. *Proceedings of NAACL-2001*, 2001.

V. M. Homes, J. O'Regan, and K.G. Evensen. Eye fixation patterns during the reading of relative clause sentences. *Journal of Verbal Learning and Verbal Behavior*, 20:417–430, 1981.

A. K. Joshi and B. Srinivas. Disambiguation of super parts of speech (or supertags): almost parsing. *The Proceedings of the 15th International Confer-ence on Computational Lingusitics (COLING* $'94$*)*, pages 154–160, 1994.

C. Juliano and M.K. Tanenhaus. A constraint-based lexicalist account of the subject-object attachment preference. *Journal of Psycholinguistic Research*, 23:459–471, 1994.

D Jurafsky. A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science*, 20:137–194, 1996.

A. E. Kim, Bangalore S., and J. Trueswell. A computational model of the grammatical aspects of word recognition as supertagging. paola merlo and suzanne stevenson (eds.). *The Lexical Basis of Sentence Processing: Formal, computational and experimental issues*, University of Geneva University of Toronto:109–135, 2002.

J. King and M. A. Just. Individual differences in syntactic processing: The role of working memory. *Journal of Memory and Language*, 30:580–602, 1991.

B. MacWhinney. Basic syntactic processes. *Language acquisition; Syntax and semantics, S. Kuczaj (Ed.)*, 1:73–136, 1982.

W. M. Mak, Vonk W., and H. Schriefers. The influence of animacy on relative clause processing. *Journal of Memory and Language,*, 47:50–68, 2002.

C.D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

S. Narayanan and D Jurafsky. A bayesian model predicts human parse preference and reading times in sentence processing. *Proceedings of Advances in Neural Information Processing Systems*, 2002.

B. Roark. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27 (2):249–276, 2001.

M. J. Traxler, R. K. Morris, and R. E. Seely. Processing subject and object relative clauses: evidence from eye movements. *Journal of Memory and Language*, 47:69–90, 2002.

J. C. Trueswell. The role of lexical frequency in syntactic ambiguity resolution. *Journal of Memory and Language*, 35:556–585, 1996.

A. Viterbi. Error bounds for convolution codes and an asymptotically optimal decoding algorithm. *IEEE Transactions of Information Theory*, 13: 260–269, 1967.

Acceptability Prediction by Means of Grammaticality Quantification

Philippe Blache, Barbara Hemforth & Stéphane Rauzy
Laboratoire Parole & Langage
CNRS - Université de Provence
29 Avenue Robert Schuman
13621 Aix-en-Provence, France
{blache,hemforth,rauzy}@lpl.univ-aix.fr

Abstract

We propose in this paper a method for quantifying sentence grammaticality. The approach based on *Property Grammars*, a constraint-based syntactic formalism, makes it possible to evaluate a grammaticality index for any kind of sentence, including ill-formed ones. We compare on a sample of sentences the grammaticality indices obtained from PG formalism and the acceptability judgements measured by means of a psycholinguistic analysis. The results show that the derived grammaticality index is a fairly good tracer of acceptability scores.

1 Introduction

Syntactic formalisms make it possible to describe precisely the question of grammaticality. When a syntactic structure can be associated to a sentence, according to a given grammar, we can decide whether or not the sentence is grammatical. In this conception, a language (be it natural or not) is *produced* (or generated) by a grammar by means of a specific mechanism, for example *derivation*. However, when no structure can be built, nothing can be said about the input to be parsed except, eventually, the origin of the failure. This is a problem when dealing with non canonical inputs such as spoken language, e-mails, non-native speaker productions, etc. From this perspective, we need robust approaches that are at the same time capable of describing precisely the form of the input, the source of the problem and to continue the parse. Such capabilities render it possible to arrive at a precise evaluation of the grammaticality of the input. In other words, instead of deciding on the grammaticality of the input, we can give an *indication* of its grammaticality, quantified on the basis of the description of the properties of the input.

This paper addresses the problem of ranking the grammaticality of different sentences. This question is of central importance for the understanding of language processing, both from an automatic and from a cognitive perspective. As for NLP, ranking grammaticality makes it possible to control dynamically the parsing process (in choosing the most adequate structures) or to find the best structure among a set of solutions (in case of non-deterministic approaches). Likewise the description of cognitive processes involved in language processing by human has to explain how things work when faced with unexpected or non canonical material. In this case too, we have to explain why some productions are more acceptable and easier to process than others.

The question of ranking grammaticality has been addressed from time to time in linguistics, without being a central concern. Chomsky, for example, mentioned this problem quite regularly (see for example (Chomsky75)). However he rephrases it in terms of "degrees of 'belongingness' to the language", a somewhat fuzzy notion both formally and linguistically. More recently, several approaches have been proposed illustrating the interest of describing these mechanisms in terms of constraint violations. The idea consists in associating weights to syntactic constraints and to evaluate, either during or after the parse, the weight of violated constraints. This approach is at the basis of *Linear Optimality Theory* (see (Keller00), and (Sorace05) for a more general perspective) in which grammaticality is judged on the basis of the total weights of violated constraints. It is then possible to rank different candidate struc-

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 57–64,
Sydney, July 2006. ©2006 Association for Computational Linguistics

tures. A similar idea is proposed in the framework of *Constraint Dependency Grammar* (see (Menzel98), (Schröder02)). In this case too, acceptability is function of the violated constraints weights.

However, constraint violation cannot in itself constitute a measure of grammaticality without taking into account other parameters as well. The type and the number of constraints that are satisfied are of central importance in acceptability judgment: a construction violating 1 constraint and satisfying 15 of them is more acceptable than one violating the same constraint but satisfying only 5 others. In the same way, other informations such as the position of the violation in the structure (whether it occurs in a deeply embedded constituent or higher one in the structure) plays an important role as well.

In this paper, we propose an approach overcoming such limitations. It takes advantage of a fully constraint-based syntactic formalism (called *Property Grammars*, cf. (Blache05b)) that offers the possibility of calculating a grammaticality index, taking into account automatically derived parameters as well as empirically determined weights. This index is evaluated automatically and we present a psycholinguistic study showing how the parser predictions converge with acceptability judgments.

2 Constraint-based parsing

Constraints are generally used in linguistics as a control process, verifying that a syntactic structure (e.g. a tree) verifies some well-formedness conditions. They can however play a more general role, making it possible to express syntactic information without using other mechanism (such as a generation function). *Property Grammars* (noted hereafter *PG*) are such a fully constraint-based formalism. In this approach, constraints stipulate different kinds of relation between categories such as linear precedence, imperative co-occurrence, dependency, repetition, etc. Each of these syntactic relations corresponds to a type of constraint (also called property):

- *Linear precedence*: $Det \prec N$ (a determiner precedes the noun)
- *Dependency*: $AP \rightsquigarrow N$ (an adjectival phrase depends on the noun)
- *Requirement*: $V[inf] \Rightarrow to$ (an infinitive comes with *to*)
- *Exclusion*: *seems* $\not\Leftrightarrow$ *ThatClause[subj]* (the verb *seems* cannot have *That* clause subjects)
- *Uniqueness* : $Uniq_{NP}\{Det\}$ (the determiner is unique in a *NP*)
- *Obligation* : $Oblig_{NP}\{N, Pro\}$ (a pronoun or a noun is mandatory in a *NP*)
- *Constituency* : $Const_{NP}\{Det, AP, N, Pro\}$ (set of possible constituents of *NP*)

In PG, each category of the grammar is described with a set of properties. A grammar is then made of a set of properties. Parsing an input consists in verifying for each category of description the set of corresponding properties in the grammar. More precisely, the idea consists in verifying, for each subset of constituents, the properties for which they are relevant (i.e. the constraints that can be evaluated). Some of these properties are satisfied, some others possibly violated. The result of a parse, for a given category, is the set of its relevant properties together with their evaluation. This result is called *characterization* and is formed by the subset of the satisfied properties, noted P^+, and the set of the violated ones, noted P^-.

For example, the characterizations associated to the *NP*s "*the book*" and "*book the*" are respectively of the form:
P^+=$\{Det \prec N;\ Det \rightsquigarrow N;\ N \not\Leftrightarrow Pro;\ Uniq(Det), Oblig(N), etc.\}$, P^-=$\emptyset$
P^+=$\{Det \rightsquigarrow N;\ N \not\Leftrightarrow Pro;\ Uniq(Det), Oblig(N), etc.\}$, P^-=$\{Det \prec N\}$

This approach allows to characterize any kind of syntactic object. In PG, following the proposal made in *Construction Grammar* (see (Fillmore98), (Kay99)), all such objects are called constructions. They correspond to a phrase (NP, PP, etc.) as well as a syntactic turn (cleft, wh-questions, etc.). All these objects are described by means of a set of properties (see (Blache05b)).

In terms of parsing, the mechanism consists in exhibiting the potential constituents of a given construction. This stage corresponds, in constraint solving techniques, to the search of an assignment satisfying the constraint system. The particularity in PG comes from constraint relaxation. Here, the goal is not to find the assignment satisfying the constraint system, but the best assignment (i.e. the one satisfying as much as possible the system). In this way, the PG approach permits to deal with more or less grammatical sentences. Provided that some control mechanisms are added to the process, PG parsing can be robust and efficient (see (Blache06)) and parse different material, including spoken language corpora.

Using a constraint-based approach such as the one proposed here offers several advantages. First, constraint relaxation techniques make it possible to process any kind of input. When parsing non canonical sentences, the system identifies precisely, for each constituent, the satisfied constraints as well as those which are violated. It furnishes the possibility of parsing any kind of input, which is a pre-requisite for identifying a graded scale of grammaticality. The second important interest of constraints lies in the fact that syntactic information is represented in a non-holistic manner or, in other words, in a decentralized way. This characteristic allows to evaluate precisely the syntactic description associated with the input. As shown above, such a description is made of sets of satisfied and violated constraints. The idea is to take advantage of such a representation for proposing a quantitative evaluation of these descriptions, elaborated from different indicators such as the number of satisfied or violated constraints or the number of evaluated constraints.

The hypothesis, in the perspective of a gradience account, is to exhibit a relation between a quantitative evaluation and the level of grammaticality: the higher the evaluation value, the more grammatical the construction. The value is then an indication of the quality of the input, according to a given grammar. In the next section we propose a method for computing this value.

3 Characterization evaluation

The first idea that comes to mind when trying to quantify the quality of a characterization is to calculate the ratio of satisfied properties with respect to the total set of evaluated properties. This information is computed as follows:

Let C a construction defined in the grammar by means of a set of properties S_C, let A_C an assignment for the construction C,

- P^+ = set of satisfied properties for A_C
- P^- = set of violated properties for A_C
- N^+ : number of satisfied properties $N^+ = card(P^+)$
- N^- : number of violated properties $N^- = card(P^-)$
- *Satisfaction ratio* (SR): the number of satisfied properties divided by the number of evaluated properties $SR = \frac{N^+}{E}$

The SR value varies between 0 and 1, the two extreme values indicating that no properties are satisfied (*SR=0*) or none of them are violated (*SR=1*). However, SR only relies on the evaluated properties. It is also necessary to indicate whether a characterization uses a small or a large subpart of the properties describing the construction in the grammar. For example, the *VP* in our grammar is described by means of 25 constraints whereas the *PP* only uses 7 of them. Let's imagine the case where 7 constraints can be evaluated for both constructions, with an equal *SR*. However, the two constructions do not have the same quality: one relies on the evaluation of all the possible constraints (in the *PP*) whereas the other only uses a few of them (in the *VP*). The following formula takes these differences into account :

- E : number of relevant (i.e. evaluated) properties $E = N^+ + N^-$
- T= number of properties specifying construction $C = card(SC)$
- Completeness coefficient (CC) : the number of evaluated properties divided by the number of properties describing the construction in the grammar $CC = \frac{E}{T}$

These purely quantitative aspects have to be contrasted according to the constraint types. Intuitively, some constraints, for a given construction, play a more important role than some others. For example, linear precedence in languages with poor morphology such as English or French may have a greater importance than obligation (i.e. the necessity of realizing the head). To its turn, obligation may be more important than uniqueness (i.e. impossible repetition). In this case, violating a property would have different consequences according to its relative importance. The following examples illustrate this aspect:

(1) a. *The the man who spoke with me is my brother.*
b. *The who spoke with me man is my brother.*

In (1a), the determiner is repeated, violating a uniqueness constraint of the first *NP*, whereas (1c) violates a linearity constraint of the same *NP*.

Clearly, (1a) seems to be more grammatical than (1b) whereas in both cases, only one constraint is violated. This contrast has to be taken into account in the evaluation. Before detailing this aspect, it is important to note that this intuition does not mean that constraints have to be organized into a ranking scheme, as with the *Optimality Theory* (see (Prince93)). The parsing mechanism remains the same with or without this information and the hierarchization only plays the role of a process control.

Identifying a relative importance of the types of constraints comes to associate them with a weight. Note that at this stage, we assign weights to constraint types, not directly to the constraints, differently from other approaches (cf. (Menzel98), (Foth05)). The experiment described in the next section will show that this weighting level seems to be efficient enough. However, in case of necessity, it remains possible to weight directly some constraints into a given construction, overriding thus the default weight assigned to the constraint types.

The notations presented hereafter are used to describe constraint weighting. Remind that P^+ and P^- indicate the set of satisfied and violated properties of a given construction.

- p_i^+ : property belonging to P^+
- p_i^- : property belonging to P^-
- $w(p)$: weight of the property of type p
- W^+ : sum of the satisfied properties weights

$$W^+ = \sum_{i=1}^{N^+} w(p_i^+)$$

- W^- : sum of the violated properties weights

$$W^- = \sum_{i=1}^{N^-} w(p_i^-)$$

One indication of the relative importance of the constraints involved in the characterization of a construction is given by the following formula:

- *QI*: the quality index of a construction

$$QI = \frac{W^+ - W^-}{W^+ + W^-}$$

The QI index varies then between -1 and 1. A negative value indicates that the set of violated constraints has a greater importance than the set of satisfied one. This does not mean that more constraints are violated than satisfied, but indicates the importance of the violated ones.

We now have three different indicators that can be used in the evaluation of the characterization: the satisfaction ratio (noted SR) indicating the ratio of satisfied constraints, the completeness coefficient (noted CC) specifying the ratio of evaluated constraints, and the quality index (noted QI) associated to the quality of the characterization according to the respective degree of importance of evaluated constraints. These three indices are used to form a global precision index (noted PI). These three indicators do not have the same impact in the evaluation of the characterization, they are then balanced with coefficients in the normalized formula:

- $PI = \frac{(k \times QI)+(l \times SR)+(m \times CC)}{3}$

As such, PI constitutes an evaluation of the characterization for a given construction. However, it is necessary to take into account the "quality" of the constituents of the construction as well. A construction can satisfy all the constraints describing it, but can be made of embedded constituents more or less well formed. The overall indication of the quality of a construction has then to integrate in its evaluation the quality of each of its constituents. This evaluation depends finally on the presence or not of embedded constructions. In the case of a construction made of lexical constituents, no embedded construction is present and the final evaluation is the precision index *PI* as described above. We will call hereafter the evaluation of the quality of the construction the "*grammaticality index*" (noted GI). It is calculated as follows:

- Let d the number of embedded constructions
- If $d = 0$ then $GI = PI$, else

$$GI = PI \times \frac{\sum_{i=1}^{d} GI(C_i)}{d}$$

In this formula, we note $GI(C_i)$ the grammaticality index of the construction C_i. The general formula for a construction C is then a function of its precision index and of the sum of the grammaticality indices of its embedded constituents. This formula implements the propagation of the quality of each constituent. This means that the grammaticality index of a construction can be lowered when its constituents violate some properties. Reciprocally, this also means that violating a property at an embedded level can be partially compensated at the upper levels (provided they have a good grammaticality index).

4 Grammaticality index from PG

We describe in the remainder of the paper predictions of the model as well as the results of a psycholinguistic evaluation of these predictions. The idea is to evaluate for a given set of sentences on the one hand the grammaticality index (done automatically), on the basis of a PG grammar, and on the other hand the acceptability judgment given by a set of subjects. This experiment has been done for French, a presentation of the data and the experiment itself will be given in the next section. We present in this section the evaluation of grammaticality index.

Before describing the calculation of the different indicators, we have to specify the constraints weights and the balancing coefficients used in PI. These values are language-dependent, they are chosen intuitively and partly based on earlier analysis, this choice being evaluated by the experiment as described in the next section. In the remainder, the following values are used:

Constraint type	Weight
Exclusion, Uniqueness, Requirement	2
Obligation	3
Linearity, Constituency	5

Concerning the balancing coefficients, we give a greater importance to the quality index (coefficient *k=2*), which seems to have important consequences on the acceptability, as shown in the previous section. The two other coefficients are significatively less important, the satisfaction ratio being at the middle position (coefficient *l=1*) and the completeness at the lowest (coefficient *m=0,5*).

Let's start with a first example, illustrating the process in the case of a sentence satisfying all constraints.

(2) Marie a emprunté un très long chemin pour le retour.
Mary took a very long way for the return.

The first NP contains one lexical constituent, Mary. Three constraints, among the 14 describing the *NP*, are evaluated and all satisfied: *Oblig(N)*, stipulating that the head is realized, *Const(N)*, indicating the category *N* as a possible constituent, and *Excl(N, Pro)*, verifying that *N* is not realized together with a pronoun. The following values come from this characterization:

N+	N-	E	T	W+	W-	QI	SR	CC	PI	GI
3	0	3	14	10	0	1	1	0.21	1.04	1.04

We can see that, according to the fact that all evaluated constraints are satisfied, *QI* and *SR* equal 1. However, the fact that only 3 constraints among 14 are evaluated lowers down the grammatical index. This last value, insofar as no constituents are embedded, is the same as *PI*.

These results can be compared with another constituent of the same sentence, the *VP*. This construction also only contains satisfied properties. Its characterization is the following : *Char(VP)=Const(Aux, V, NP, PP) ; Oblig(V) ; Uniq(V) ; Uniq(NP) ; Uniq(PP) ; Aux⇒V[part] ; V≺NP ; Aux≺V ; V≺PP*. On top of this set of evaluated constraints (9 among the possible 25), the VP includes two embedded constructions : a *PP* and a *NP*. A grammaticality index has been calculated for each of them: *GI(PP) = 1.24 GI(NP)=1.23*. The following table indicates the different values involved in the calculation of the *GI*.

N+	N-	E	T	W+	W-	QI	SR	CC	PI
9	0	9	25	31	0	1	1	0.36	1.06

GI_Emb_Const	GI
1.23	1.31

The final *GI* of the *VP* reaches a high value. It benefits on the one hand from its own quality (indicated by *PI*) and on another hand from that of its embedded constituents. In the end, the final *GI* obtained at the sentence level is function of its own *PI* (very good) and the *NP* and *VP GI*s, as shown in the table:

N+	N-	E	T	W+	W-	QI	SR	CC	PI
5	0	5	9	17	0	1	1	0.56	1.09

GI_Emb_Const	GI
1.17	1.28

Let's compare now these evaluations with those obtained for sentences with violated constraints, as in the following examples:

(3) a. Marie a emprunté très long chemin un pour le retour.
Mary took very long way a for the return.

b. Marie a emprunté un très chemin pour le retour.
Mary took a very way for the return.

In (2a), 2 linear constraints are violated: a determiner follows a noun and an *AP* in "*très long chemin un*". Here are the figures calculated for this *NP*:

N+	N-	E	T	W+	W-	QI	SR	CC	PI	GI
8	2	10	14	23	10	0.39	0.80	0.71	0.65	0.71

The *QI* indicator is very low, the violated constraints being of heavy weight. The grammaticality index is a little bit higher because a lot of constraints are also satisfied. The *NP GI* is then propagated to its dominating construction, the *VP*. This phrase is well formed and also contains a well-formed construction (*PP*) as sister of the *NP*. Note that in the following table summarizing the *VP* indicators, the *GI* product of the embedded constituents is higher than the *GI* of the *NP*. This is due to the well-formed *PP* constituent. In the end, the *GI* index of the *VP* is better than that of the ill-formed *NP*:

N+	N-	E	T	W+	W-	QI	SR	CC	PI
9	0	9	25	31	0	1	1	0.36	1.06

GI_Emb_Const	GI
0.97	1.03

For the same reasons, the higher level construction *S* also compensates the bad score of the *NP*. However, in the end, the final *GI* of the sentence is much lower than that of the corresponding well-formed sentence (see above).

N+	N-	E	T	W+	W-	QI	SR	CC	PI
5	0	5	9	17	0	1	1	0.56	1.09

GI_Emb_Const	GI
1.03	1.13

The different figures of the sentence (2b) show that the violation of a unique constraint (in this case the *Oblig(Adj)* indicating the absence of the head in the *AP*) can lead to a global lower *GI* than the violation of two heavy constraints as for (2a). In this case, this is due to the fact that the AP only contains one constituent (a modifier) that does not suffice to compensate the violated constraint. The following table indicates the indices of the different phrases. Note that in this table, each phrase is a constituent of the following (i.e. *AP* belongs to *NP* itself belonging to *VP*, and so on).

	N+	N-	E	T	W+	W-	QI	SR	CC	PI
AP	2	1	3	7	7	3	0.40	0.67	0.43	0.56
NP	10	0	10	14	33	0	1	1	0.71	1.12
VP	9	0	9	25	31	0	1	1	0.36	1.06
S	5	0	5	9	17	0	1	1	0.56	1.09

	GI_Emb_Const	GI
AP	1	0.56
NP	0.56	0.63
VP	0.93	0.99
S	1.01	1.11

5 Judging acceptability of violations

We ran a questionnaire study presenting participants with 60 experimental sentences like (11) to (55) below. 44 native speakers of French completed the questionnaire giving acceptability judgements following the Magnitude Estimation technique. 20 counterbalanced forms of the questionnaire were constructed. Three of the 60 experimental sentences appeared in each version in each form of the questionnaire, and across the 20 forms, each experimental sentence appeared once in each condition. Each sentence was followed by a question concerning its acceptability. These 60 sentences were combined with 36 sentences of various forms varying in complexity (simple main clauses, simple embeddings and doubly nested embeddings) and plausibility (from fully plausible to fairly implausible according to the intuitions of the experimenters). One randomization was made of each form.

Procedure: The rating technique used was magnitude estimation (ME, see (Bard96)). Participants were instructed to provide a numeric score that indicates how much better (or worse) the current sentence was compared to a given reference sentence (Example: If the reference sentence was given the reference score of 100, judging a target sentence five times better would result in 500, judging it five times worse in 20). Judging the acceptability ratio of a sentence in this way results in a scale which is open-ended on both sides. It has been demonstrated that ME is therefore more sensitive than fixed rating-scales, especially for scores that would approach the ends of such rating scales (cf. (Bard96)). Each questionnaire began with a written instruction where the subject was made familiar with the task based on two examples. After that subjects were presented with a reference sentence for which they had to provide a reference score. All following sentences had to be judged in relation to the reference sentence. Individual judgements were logarithmized (to arrive at a linear scale) and normed (z-standardized) before statistical analyses.

Global mean scores are presented figure 1. We tested the reliability of results for different randomly chosen subsets of the materials. Constructions for which the judgements remain highly stable across subsets of sentences are marked by an asterisk ($rs > 0.90; p < 0.001$). The mean reliability across subsets is $rs > 0.65$ ($p < 0.001$).

What we can see in these data is that in particular violations within prepositional phrases are not judged in a very stable way. The way they are judged appears to be highly dependent on the preposition used and the syntactic/semantic context. This is actually a very plausible result, given that heads of prepositional phrases are closed class items that are much more predictable in many syntactic and semantic environments than heads of noun phrases and verb phrases. We will therefore base our further analyses mainly on violations within noun phrases, verb phrases, and adjectival phrases. Results including prepositional phrases will be given in parentheses. Since the constraints described above do not make any predictions for semantic violations, we excluded examples 25, 34, 45, and 55 from further analyses.

6 Acceptability versus grammaticality index

We compare in this section the results coming from the acceptability measurements described in section 5 and the values of grammaticality indices obtained as proposed section 4.

From the sample of 20 sentences presented in figure 1, we have discarded 4 sentences, namely sentence 25, 34, 45 and 55, for which the property violation is of semantic order (see above). We are left with 16 sentences, the reference sentence satisfying all the constraints and 15 sentences violating one of the syntactic constraints. The results are presented figure 2. Acceptability judgment (ordinate) versus grammaticality index (abscissa) is plotted for each sentence. We observe a high coefficient of correlation ($\rho = 0.76$) between the two distributions, indicating that the grammaticality index derived from PG is a fairly good tracer of the observed acceptability measurements.

The main contribution to the grammaticality index comes from the quality index *QI* ($\rho = 0.69$) while the satisfaction ratio *SR* and the completeness coefficient CC contributions, although significant, are more modest ($\rho = 0.18$ and $\rho = 0.17$ respectively).

No violations	
11. Marie *a emprunté un très long chemin pour le retour*	0.465
NP-violations	
21. *Marie a emprunté très long chemin un pour le retour*	-0.643 *
22. *Marie a emprunté un très long chemin chemin pour le retour*	-0.161 *
23. *Marie a emprunté un très long pour le retour*	-0.871 *
24. *Marie a emprunté très long chemin pour le retour*	-0.028 *
25. *Marie a emprunté un très heureux chemin pour le retour*	-0.196 *
AP-violations	
31. *Marie a emprunté un long très chemin pour le retour*	-0.41 *
32. *Marie a emprunté un très long long chemin pour le retour*	-0.216 -
33. *Marie a emprunté un très chemin pour le retour*	-0.619 -
34. *Marie a emprunté un grossièrement long chemin pour le retour*	-0.058 *
PP-violations	
41. Marie *a emprunté un très long chemin le retour pour*	-0.581 -
42. *Marie a emprunté un très long chemin pour pour le retour*	-0.078 -
43. *Marie a emprunté un très long chemin le retour*	-0.213 -
44. Marie *a emprunté un très long chemin pour*	-0.385 -
45. *Marie a emprunté un très long chemin dans le retour*	-0.415 -
VP-violations	
51. *Marie un très long chemin a emprunté pour le retour*	-0.56 *
52.*Marie a emprunté emprunté un très long chemin pour le retour*	-0.194 *
53.Marie *un très long chemin pour le retour*	-0.905 *
54. *Marie emprunté un très long chemin pour le retour*	-0.322 *
55. *Marie a persuadé un très long chemin pour le retour*	-0.394 *

Figure 1: Acceptability results

We present in figure 3 the correlation between acceptability judgements and grammaticality indices after the removal of the 4 sentences presenting *PP* violations. The analysis of the experiment described in section 5 shows indeed that acceptability measurements of the PP-violation sentences is less reliable than for others phrases. We thus expect that removing these data from the sample will strengthen the correlation between the two distributions. The coefficient of correlation of the 12 remaining data jumps to $\rho = 0.87$, as expected.

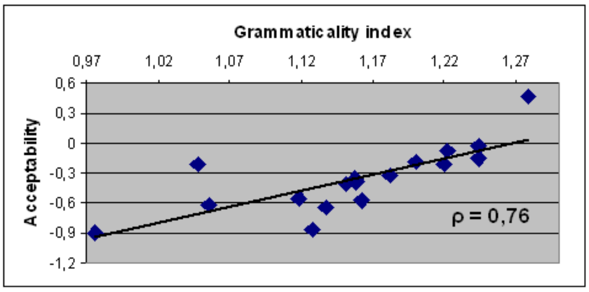


Figure 2: Correlation between acceptability judgement and grammaticality index

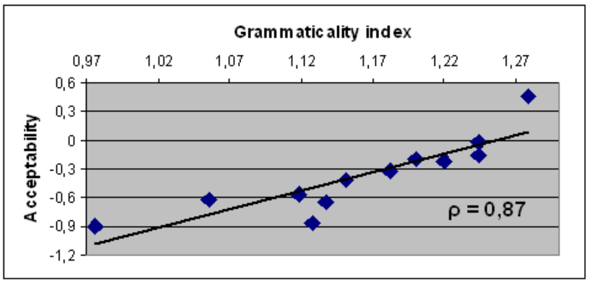


Figure 3: Correlation between acceptability judgement and grammaticality index removing PP violations

Finally, the adequacy of the PG grammaticality indices to the measurements was investigated by means of resultant analysis. We adapted the parameters of the model in order to arrive at a good fit based on half of the sentences materials (randomly chosen from the full set), with a correlation of $\rho = 0.85$ ($\rho = 0.76$ including *PP*s) between the grammaticality index and acceptability judgements. Surprisingly, we arrived at the best fit with only two different weights: A weight of 2 for *Exclusion*, *Uniqueness*, and *Requirement*, and a weight of 5 for *Obligation*, *Linearity*, and *Constituency*. This result converges with the hard and soft constraint repartition idea as proposed by (Keller00).

The fact that the grammaticality index is based on these properties as well as on the number of constraints to be evaluated, the number of constraints to the satisfied, and the goodness of embedded constituents apparently results in a fined grained and highly adequate prediction even with this very basic distinction of constraints.

Fixing these parameters, we validated the predictions of the model for the remaining half of the materials. Here we arrived at a highly reliable correlation of $\rho = 0.86$ ($\rho = 0.67$ including *PP*s) between PG grammaticality indices and acceptability judgements.

7 Conclusion

The method described in this paper makes it possible to give a quantified indication of sentence grammaticality. This approach is direct and takes advantage of a constraint-based representation of syntactic information, making it possible to represent precisely the syntactic characteristics of an input in terms of satisfied and (if any) violated constraints. The notion of *grammaticality index* we have proposed here integrates different kind of information: the quality of the description (in terms of well-formedness degree), the density of information (the quantity of constraints describing an element) as well as the structure itself. These three parameters are the basic indicators of the grammaticality index.

The relevance of this method has been experimentally shown, and the results described in this paper illustrate the correlation existing between the prediction (automatically calculated) expressed in terms of GI and the acceptability judgment given by subjects.

This approach also presents a practical interest: it can be directly implemented into a parser. The next step of our work will be its validation on large corpora. Our parser will associate a grammatical index to each sentence. This information will be validated by means of acceptability judgments acquired on the basis of a sparse sampling strategy.

References

Bard E., D. Robertson & A. Sorace (1996) "Magnitude Estimation of Linguistic Acceptability", Language 72:1.

Blache P. & J.-P. Prost (2005) "Gradience, Constructions and Constraint Systems", in H. Christiansen & al. (eds), *Constraint Solving and NLP*, Lecture Notes in Computer Science, Springer.

Blache P. (2005) "Property Grammars: A Fully Constraint-Based Theory", in H. Christiansen & al. (eds), *Constraint Solving and NLP*, Lecture Notes in Computer Science, Springer.

Blache P. (2006) "A Robust and Efficient Parser for Non-Canonical Inputs", in proceedings of *Robust Methods in Analysis of Natural Language Data*, EACL workshop.

Chomsky N.. (1975) *The Logical Structure of Linguistic Theory*, Plenum Press

Croft W. & D. Cruse (2003) *Cognitive Linguistics*, Cambridge University Press.

Foth K., M. Daum & W. Menzel (2005) "Parsing Unrestricted German Text with Defeasible Constraints", in H. Christiansen & al. (eds), *Constraint Solving and NLP*, Lecture Notes in Computer Science, Springer.

Fillmore C. (1998) "Inversion and Contructional Inheritance", in *Lexical and Constructional Aspects of Linguistic Explanation*, Stanford University.

Kay P. & C. Fillmore (1999) "Grammatical Constructions and Linguistic Generalizations: the *what's x doing y* construction", Language.

Keller F. (2000) *Gradience in Grammar. Experimental and Computational Aspects of Degrees of Grammaticality*, Phd Thesis, University of Edinburgh.

Keller F. (2003) "A probabilistic Parser as a Model of Global Processing Difficulty", in proceedings of *ACCSS-03*

Menzel W. & I. Schroder (1998) "Decision procedures for dependency parsing using graded constraints", in S. Kahane & A. Polguère (eds), Proc. ColingACL Workshop on Processing of Dependency-based Grammars.

Prince A. & Smolensky P. (1993) *Optimality Theory: Constraint Interaction in Generative Grammars*, Technical Report RUCCS TR-2, Rutgers Center for Cognitive Science.

Sag I., T. Wasow & E. Bender (2003) *Syntactic Theory. A Formal Introduction*, CSLI.

Schröder I. (2002) *Natural Language Parsing with Graded Constraints*. PhD Thesis, University of Hamburg.

Sorace A. & F. Keller (2005) "Gradience in Linguistic Data", in *Lingua*, 115.

Discriminative Word Alignment with Conditional Random Fields

Phil Blunsom and **Trevor Cohn**
Department of Software Engineering and Computer Science
University of Melbourne
{pcbl,tacohn}@csse.unimelb.edu.au

Abstract

In this paper we present a novel approach for inducing word alignments from sentence aligned data. We use a Conditional Random Field (CRF), a discriminative model, which is estimated on a small supervised training set. The CRF is conditioned on both the source and target texts, and thus allows for the use of arbitrary and overlapping features over these data. Moreover, the CRF has efficient training and decoding processes which both find globally optimal solutions.

We apply this alignment model to both French-English and Romanian-English language pairs. We show how a large number of highly predictive features can be easily incorporated into the CRF, and demonstrate that even with only a few hundred word-aligned training sentences, our model improves over the current state-of-the-art with alignment error rates of 5.29 and 25.8 for the two tasks respectively.

1 Introduction

Modern phrase based statistical machine translation (SMT) systems usually break the translation task into two phases. The first phase induces word alignments over a sentence-aligned bilingual corpus, and the second phase uses statistics over these predicted word alignments to decode (translate) novel sentences. This paper deals with the first of these tasks: word alignment.

Most current SMT systems (Och and Ney, 2004; Koehn et al., 2003) use a generative model for word alignment such as the freely available GIZA++ (Och and Ney, 2003), an implementation of the IBM alignment models (Brown et al., 1993). These models treat word alignment as a hidden process, and maximise the probability of the observed $(\mathbf{e}, \mathbf{f})$ sentence pairs[1] using the expectation maximisation (EM) algorithm. After the maximisation process is complete, the word alignments are set to maximum posterior predictions of the model.

While GIZA++ gives good results when trained on large sentence aligned corpora, its generative models have a number of limitations. Firstly, they impose strong independence assumptions between features, making it very difficult to incorporate non-independent features over the sentence pairs. For instance, as well as detecting that a source word is aligned to a given target word, we would also like to encode syntactic and lexical features of the word pair, such as their parts-of-speech, affixes, lemmas, etc. Features such as these would allow for more effective use of sparse data and result in a model which is more robust in the presence of unseen words. Adding these non-independent features to a generative model requires that the features' inter-dependence be modelled explicitly, which often complicates the model (eg. Toutanova et al. (2002)). Secondly, the later IBM models, such as Model 4, have to resort to heuristic search techniques to approximate forward-backward and Viterbi inference, which sacrifice optimality for tractability.

This paper presents an alternative discriminative method for word alignment. We use a conditional random field (CRF) sequence model, which allows for globally optimal training and decoding (Lafferty et al., 2001). The inference algo-

[1] We adopt the standard notation of $\mathbf{e}$ and $\mathbf{f}$ to denote the target (English) and source (foreign) sentences, respectively.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 65–72,
Sydney, July 2006. ©2006 Association for Computational Linguistics

rithms are tractable and efficient, thereby avoiding the need for heuristics. The CRF is conditioned on both the source and target sentences, and therefore supports large sets of diverse and overlapping features. Furthermore, the model allows regularisation using a prior over the parameters, a very effective and simple method for limiting over-fitting. We use a similar graphical structure to the directed hidden Markov model (HMM) from GIZA++ (Och and Ney, 2003). This models one-to-many alignments, where each target word is aligned with zero or more source words. Many-to-many alignments are recoverable using the standard techniques for superimposing predicted alignments in both translation directions.

The paper is structured as follows. Section 2 presents CRFs for word alignment, describing their form and their inference techniques. The features of our model are presented in Section 3, and experimental results for word aligning both French-English and Romanian-English sentences are given in Section 4. Section 5 presents related work, and we describe future work in Section 6. Finally, we conclude in Section 7.

2 Conditional random fields

CRFs are undirected graphical models which define a conditional distribution over a label sequence given an observation sequence. We use a CRF to model many-to-one word alignments, where each source word is aligned with zero or one target words, and therefore each target word can be aligned with many source words. Each source word is labelled with the index of its aligned target, or the special value *null*, denoting no alignment. An example word alignment is shown in Figure 1, where the hollow squares and circles indicate the correct alignments. In this example the French words *une* and *autre* would both be assigned the index 24 – for the English word *another* – when French is the source language. When the source language is English, *another* could be assigned either index 25 or 26; in these ambiguous situations we take the first index.

The joint probability density of the alignment, $\mathbf{a}$ (a vector of target indices), conditioned on the source and target sentences, $\mathbf{e}$ and $\mathbf{f}$, is given by:

$$p_\Lambda(\mathbf{a}|\mathbf{e},\mathbf{f}) = \frac{\exp \sum_t \sum_k \lambda_k h_k(t, a_{t-1}, a_t, \mathbf{e}, \mathbf{f})}{Z_\Lambda(\mathbf{e},\mathbf{f})} \tag{1}$$

where we make a first order Markov assumption

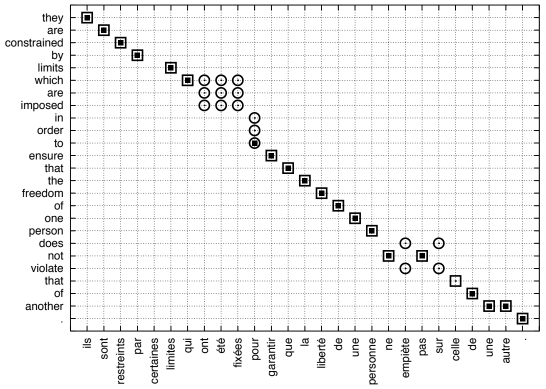


Figure 1. A word-aligned example from the Canadian Hansards test set. Hollow squares represent gold standard sure alignments, circles are gold possible alignments, and filled squares are predicted alignments.

over the alignment sequence. Here t ranges over the indices of the source sentence ($\mathbf{f}$), k ranges over the model's features, and $\Lambda = \{\lambda_k\}$ are the model parameters (weights for their corresponding features). The feature functions h_k are predefined real-valued functions over the source and target sentences coupled with the alignment labels over adjacent times (source sentence locations), t. These feature functions are unconstrained, and may represent overlapping and non-independent features of the data. The distribution is globally normalised by the partition function, $Z_\Lambda(\mathbf{e},\mathbf{f})$, which sums out the numerator in (1) for every possible alignment:

$$Z_\Lambda(\mathbf{e},\mathbf{f}) = \sum_{\mathbf{a}} \exp \sum_t \sum_k \lambda_k h_k(t, a_{t-1}, a_t, \mathbf{e}, \mathbf{f})$$

We use a linear chain CRF, which is encoded in the feature functions of (1).

The parameters of the CRF are usually estimated from a fully observed training sample (word aligned), by maximising the likelihood of these data. I.e. $\Lambda^{ML} = \arg\max_\Lambda p_\Lambda(\mathcal{D})$, where $\mathcal{D} = \{(\mathbf{a},\mathbf{e},\mathbf{f})\}$ are the training data. Because maximum likelihood estimators for log-linear models have a tendency to overfit the training sample (Chen and Rosenfeld, 1999), we define a prior distribution over the model parameters and derive a maximum *a posteriori* (MAP) estimate, $\Lambda^{MAP} = \arg\max_\Lambda p_\Lambda(\mathcal{D})p(\Lambda)$. We use a zero-mean Gaussian prior, with the probability density function $p_0(\lambda_k) \propto \exp\left(-\frac{\lambda_k^2}{2\sigma_k^2}\right)$. This yields a log-likelihood objective function of:

$$\mathcal{L} = \sum_{(\mathbf{a},\mathbf{e},\mathbf{f})\in\mathcal{D}} \log p_\Lambda(\mathbf{a}|\mathbf{e},\mathbf{f}) + \sum_k \log p_0(\lambda_k)$$

$$= \sum_{(\mathbf{a},\mathbf{e},\mathbf{f})\in\mathcal{D}} \sum_t \sum_k \lambda_k h_k(t, a_{t-1}, a_t, \mathbf{e}, \mathbf{f})$$

$$- \log Z_\Lambda(\mathbf{e}, \mathbf{f}) - \sum_k \frac{\lambda_k^2}{2\sigma_k^2} + \mathit{const.} \quad (2)$$

In order to train the model, we maximize (2). While the log-likelihood cannot be maximised for the parameters, Λ, in closed form, it is a convex function, and thus we resort to numerical optimisation to find the globally optimal parameters. We use L-BFGS, an iterative quasi-Newton optimisation method, which performs well for training log-linear models (Malouf, 2002; Sha and Pereira, 2003). Each L-BFGS iteration requires the objective value and its gradient with respect to the model parameters. These are calculated using forward-backward inference, which yields the partition function, $Z_\Lambda(\mathbf{e}, \mathbf{f})$, required for the log-likelihood, and the pair-wise marginals, $p_\Lambda(a_{t-1}, a_t|\mathbf{e}, \mathbf{f})$, required for its derivatives.

The Viterbi algorithm is used to find the maximum posterior probability alignment for test sentences, $\mathbf{a}^* = \arg\max_{\mathbf{a}} p_\Lambda(\mathbf{a}|\mathbf{e}, \mathbf{f})$. Both the forward-backward and Viterbi algorithm are dynamic programs which make use of the Markov assumption to calculate efficiently the exact marginal distributions.

3 The alignment model

Before we can apply our CRF alignment model, we must first specify the feature set – the functions h_k in (1). Typically CRFs use binary indicator functions as features; these functions are only active when the observations meet some criteria and the label a_t (or label pair, (a_{t-1}, a_t)) matches a pre-specified label (pair). However, in our model the labellings are word indices in the target sentence and cannot be compared readily to labellings at other sites in the same sentence, or in other sentences with a different length. Such naive features would only be active for one labelling, therefore this model would suffer from serious sparse data problems.

We instead define features which are functions of the source-target word match implied by a labelling, rather than the labelling itself. For example, from the sentence in Figure 1 for the labelling of $f_{24} = de$ with $a_{24} = 16$ (for $e_{16} = of$) we might detect the following feature:

$$h(t, a_{t-1}, a_t, \mathbf{f}, \mathbf{e}) = \begin{cases} 1, & \text{if } e_{a_t} = \text{‘}of\text{’} \land f_t = \text{‘}de\text{’} \\ 0, & \text{otherwise} \end{cases}$$

Note that it is the target word indexed by a_t, rather than the index itself, which determines whether the feature is active, and thus the sparsity of the index label set is not an issue.

3.1 Features

One of the main advantages of using a conditional model is the ability to explore a diverse range of features engineered for a specific task. In our CRF model we employ two main types of features: those defined on a candidate aligned pair of words; and Markov features defined on the alignment sequence predicted by the model.

Dice and Model 1 As we have access to only a small amount of word aligned data we wish to be able to incorporate information about word association from any sentence aligned data available. A common measure of word association is the Dice coefficient (Dice, 1945):

$$Dice(e, f) = \frac{2 \times C_{EF}(e, f)}{C_E(e) + C_F(e)}$$

where C_E and C_F are counts of the occurrences of the words e and f in the corpus, while C_{EF} is their co-occurrence count. We treat these Dice values as *translation scores*: a high (low) value incidates that the word pair is a good (poor) candidate translation.

However, the Dice score often over-estimates the association between common words. For instance, the words *the* and *of* both score highly when combined with either *le* or *de*, simply because these common words frequently co-occur. The GIZA++ models can be used to provide better translation scores, as they enforce competition for alignment beween the words. For this reason, we used the translation probability distribution from Model 1 in addition to the DICE scores. Model 1 is a simple position independent model which can be trained quickly and is often used to bootstrap parameters for more complex models. It models the conditional probability distribution:

$$p(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \frac{p(|\mathbf{f}|||\mathbf{e}|)}{(|\mathbf{e}| + 1)^{|\mathbf{f}|}} \times \prod_{t=1}^{|\mathbf{f}|} p(f_t|e_{a_t})$$

where $p(f|e)$ are the word translation probabilities.

We use both the Dice value and the Model 1 translation probability as real-valued features for each candidate pair, as well as a normalised score over all possible candidate alignments for each target word. We derive a feature from both the Dice and Model 1 translation scores to allow competition between sources words for a particular target alignment. This feature indicates whether a given alignment has the highest translation score of all the candidate alignments for a given *target* word. For the example in Figure 1, the words *la, de* and *une* all receive a high translation score when paired with *the*. To discourage all of these French words from aligning with *the*, the best of these (*la*) is flagged as the best candidate. This allows for competition between source words which would otherwise not occur.

Orthographic features Features based on string overlap allow our model to recognise cognates and orthographically similar translation pairs, which are particularly common between European languages. Here we employ a number of string matching features inspired by similar features in Taskar et al. (2005). We use an indicator feature for every possible source-target word pair in the training data. In addition, we include indicator features for an exact string match, both with and without vowels, and the edit-distance between the source and target words as a real-valued feature. We also used indicator features to test for matching prefixes and suffixes of length three. As stated earlier, the Dice translation score often erroneously rewards alignments with common words. In order to address this problem, we include the absolute difference in word length as a real-valued feature and an indicator feature testing whether both words are shorter than 4 characters. Together these features allow the model to disprefer alignments between words with very different lengths – i.e. aligning rare (long) words with frequent (short) determiners, verbs etc.

POS tags Part-of-speech tags are an effective method for addressing the sparsity of the lexical features. Observe in Figure 2 that the noun-adjective pair *Canadian experts* aligns with the adjective-noun pair *spécialistes canadiens*: the alignment exactly matches the parts-of-speech. Access to the words' POS tags will allow simple modelling of such effects. POS can also be useful for less closely related language pairs, such as English and Japanese where English determiners are never aligned; nor are Japanese case markers.

For our French-English language pair we POS tagged the source and target sentences with TreeTagger.[2] We created indicator features over the POS tags of each candidate source and target word pair, as well as over the source word and target POS (and vice-versa). As we didn't have access to a Romanian POS tagger, these features were not used for the Romanian-English language pair.

Bilingual dictionary Dictionaries are another source of information for word alignment. We use a single indicator feature which detects when the source and target words appear in an entry of the dictionary. For the English-French dictionary we used FreeDict,[3] which contains 8,799 English words. For Romanian-English we used a dictionary compiled by Rada Mihalcea,[4] which contains approximately 38,000 entries.

Markov features Features defined over adjacent aligment labels allow our model to reflect the tendency for monotonic alignments between European languages. We define a real-valued alignment index jump width feature:

$$jump\_width(t-1,t) = abs(a_t - a_{t-1} - 1)$$

this feature has a value of 0 if the alignment labels follow the downward sloping diagonal, and is positive otherwise. This differs from the GIZA++ hidden Markov model which has individual parameters for each different jump width (Och and Ney, 2003; Vogel et al., 1996): we found a single feature (and thus parameter) to be more effective.

We also defined three indicator features over **null** transitions to allow the modelling of the probability of transition between, to and from **null** labels.

Relative sentence postion A feature for the absolute difference in relative sentence position ($abs(\frac{a_t}{|\mathbf{e}|} - \frac{t}{|\mathbf{f}|})$) allows the model to learn a preference for aligning words close to the alignment matrix diagonal. We also included two conjunction features for the relative sentence position multiplied by the Dice and Model 1 translation scores.

Null We use a number of variants on the above features for alignments between a source word and the *null* target. The maximum translation score between the source and one of the target words

[2] http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger

[3] http://www.freedict.de

[4] http://lit.csci.unt.edu/˜rada/downloads/RoNLP/R.E.tralex

model	precision	recall	f-score	AER
Model 4 refined	87.4	95.1	91.1	9.81
Model 4 intersection	97.9	86.0	91.6	7.42
French → English	96.7	85.0	90.5	9.21
English → French	97.3	83.0	89.6	10.01
intersection	98.7	78.6	87.5	12.02
refined	95.7	89.2	92.3	7.37

Table 1. Results on the Hansard data using all features

model	precision	recall	f-score	AER
Model 4 refined	80.49	64.10	71,37	28.63
Model 4 intersected	95.94	53.56	68.74	31.26
Romanian → English	82.9	61.3	70.5	29.53
English → Romanian	82.8	60.6	70.0	29.98
intersection	94.4	52.5	67.5	32.45
refined	77.1	68.5	72.6	27.41

Table 2. Results on the Romanian data using all features

is used as a feature to represent whether there is a strong alignment candidate. The sum of these scores is also used as a feature. Each source word and POS tag pair are used as indicator features which allow the model to learn particular words of tags which tend to commonly (or rarely) align.

3.2 Symmetrisation

In order to produce many-to-many alignments we combine the outputs of two models, one for each translation direction. We use the refined method from Och and Ney (2003) which starts from the intersection of the two models' predictions and 'grows' the predicted alignments to neighbouring alignments which only appear in the output of one of the models.

4 Experiments

We have applied our model to two publicly available word aligned corpora. The first is the English-French Hansards corpus, which consists of 1.1 million aligned sentences and 484 word-aligned sentences. This data set was used for the 2003 NAACL shared task (Mihalcea and Pedersen, 2003), where the word-aligned sentences were split into a 37 sentence trial set and a 447 sentence testing set. Unlike the unsupervised entrants in the 2003 task, we require word-aligned training data, and therefore must cannibalise the test set for this purpose. We follow Taskar et al. (2005) by using the first 100 test sentences for training and the remaining 347 for testing. This means that our results should not be directly compared to those entrants, other than in an approximate manner. We used the original 37 sentence trial set for feature engineering and for fitting a Gaussian prior.

The word aligned data are annotated with both sure (S) and possible (P) alignments ($S \subseteq P$; Och and Ney (2003)), where the possible alignments indicate ambiguous or idiomatic alignments. We measure the performance of our model using *alignment error rate* (AER), which is defined as:

$$AER(A,S,P) = 1 - \frac{|A \cap S| + |A \cap P|}{|A| + |S|}$$

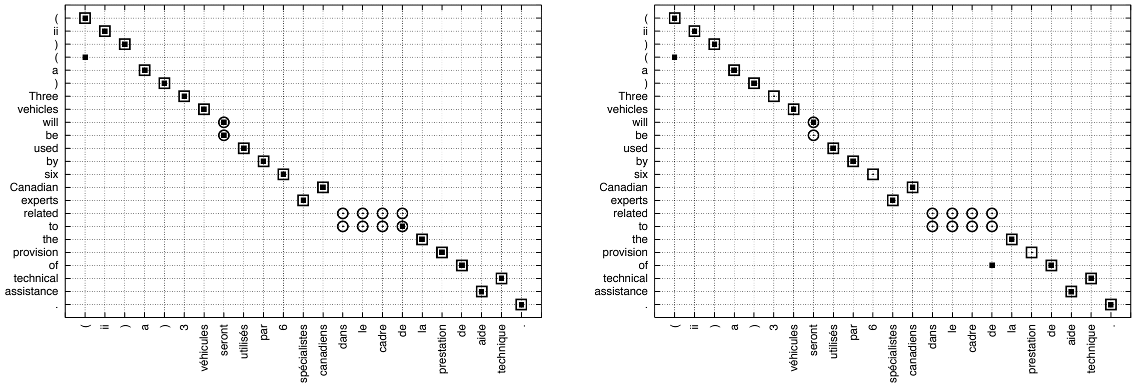
where A is the set of predicted alignments.

The second data set is the Romanian-English parallel corpus from the 2005 ACL shared task (Martin et al., 2005). This consists of approximately 50,000 aligned sentences and 448 word-aligned sentences, which are split into a 248 sentence trial set and a 200 sentence test set. We used these as our training and test sets, respectively. For parameter tuning, we used the 17 sentence trial set from the Romanian-English corpus in the 2003 NAACL task (Mihalcea and Pedersen, 2003). For this task we have used the same test data as the competition entrants, and therefore can directly compare our results. The word alignments in this corpus were only annotated with sure (S) alignments, and therefore the AER is equivalent to the F_1 score. In the shared task it was found that models which were trained on only the first four letters of each word obtained superior results to those using the full words (Martin et al., 2005). We observed the same result with our model on the trial set and thus have only used the first four letters when training the Dice and Model 1 translation probabilities.

Tables 1 and 2 show the results when all feature types are employed on both language pairs. We report the results for both translation directions and when combined using the refined and intersection methods. The Model 4 results are from GIZA++ with the default parameters and the training data lowercased. For Romanian, Model 4 was trained using the first four letters of each word.

The Romanian results are close to the best reported result of 26.10 from the ACL shared task (Martin et al., 2005). This result was from a system based on Model 4 plus additional parameters such as a dictionary. The standard Model 4 implementation in the shared task achieved a result of 31.65, while when only the first 4 letters of each word were used it achieved 28.80.[5]

[5] These results differ slightly our Model 4 results reported in Table 2.



(a) With Markov features

(b) Without Markov features

Figure 2. An example from the Hansard test set, showing the effect of the Markov features.

feature group	Rom ↔ Eng	Fre ↔ Eng
ALL	27.41	7.37
–orthographic	27.30	7.25
–Dice	27.68	7.73
–dictionary	27.72	7.21
–sentence position	28.30	8.01
–POS	–	8.19
–Model 1	28.62	8.45
–alignment word pair	32.41	7.20
–Markov	32.75	12.44
–Dice & –Model 1	35.43	14.10

Table 3. The resulting AERs after removing individual groups of features from the full model.

Table 3 shows the effect of removing each of the feature types in turn from the full model. The most useful features are the Dice and Model 1 values which allow the model to incorporate translation probabilities from the large sentence aligned corpora. This is to be expected as the amount of word aligned data are extremely small, and therefore the model can only estimate translation probabilities for only a fraction of the lexicon. We would expect the dependence on sentence aligned data to decrease as more word aligned data becomes available.

The effect of removing the Markov features can be seen from comparing Figures 2 (a) and (b). The model has learnt to prefer alignments that follow the diagonal, thus alignments such as *3* ↔ *three* and *prestation* ↔ *provision* are found, and miss-alignments such as *de* ↔ *of*, which lie well off the diagonal, are avoided.

The differing utility of the alignment word pair feature between the two tasks is probably a result of the different proportions of word- to sentence-aligned data. For the French data, where a very large lexicon can be estimated from the million sentence alignments, the sparse word pairs learnt on the word aligned sentences appear to lead to overfitting. In contrast, for Romanian, where more word alignments are used to learn the translation pair features and much less sentence aligned data are available, these features have a significant impact on the model. Suprisingly the orthographic features actually worsen the performance in the tasks (incidentally, these features help the trial set). Our explanation is that the other features (eg. Model 1) already adequately model these correspondences, and therefore the orthographic features do not add much additional modelling power. We expect that with further careful feature engineering, and a larger trial set, these orthographic features could be much improved.

The Romanian-English language pair appears to offer a more difficult modelling problem than the French-English pair. With both the translation score features (Dice and Model 1) removed – the sentence aligned data are not used – the AER of the Romanian is more than twice that of the French, despite employing more word aligned data. This could be caused by the lack of possible (P) alignment markup in the Romanian data, which provide a boost in AER on the French data set, rewarding what would otherwise be considered errors. Interestingly, without any features derived from the sentence aligned corpus, our model achieves performance equivalent to Model 3 trained on the full corpus (Och and Ney, 2003). This is a particularly strong result, indicating that this method is ideal for data-impoverished alignment tasks.

4.1 Training with *possible* alignments

Up to this point our Hansards model has been trained using only the sure (S) alignments. As the data set contains many possible (P) alignments, we would like to use these to improve our model. Most of the possible alignments flag blocks of ambiguous or idiomatic (or just difficult) phrase level alignments. These many-to-many alignments cannot be modelled with our many-to-one setup. However, a number of possibles flag one-to-one or many-to-one aligments: for this experiment we used these possibles in training to investigate their effect on recall. Using these additional alignments our refined precision decreased from 95.7 to 93.5, while recall increased from 89.2 to 92.4. This resulted in an overall decrease in AER to 6.99. We found no benefit from using many-to-many possible alignments as they added a significant amount of noise to the data.

4.2 Model 4 as a feature

Previous work (Taskar et al., 2005) has demonstrated that by including the output of Model 4 as a feature, it is possible to achieve a significant decrease in AER. We trained Model 4 in both directions on the two language pairs. We added two indicator features (one for each direction) to our CRF which were active if a given word pair were aligned in the Model 4 output. Table 4 displays the results on both language pairs when these additional features are used with the refined model. This produces a large increase in performance, and when including the possibles, produces AERs of 5.29 and 25.8, both well below that of Model 4 alone (shown in Tables 1 and 2).

model	precision	recall	f-score	AER
Rom ↔ Eng	79.0	70.0	74.2	25.8
Fre ↔ Eng	97.9	90.8	94.2	5.49
Fre ↔ Eng (P)	95.5	93.7	94.6	5.29

Table 4. Results using features from Model 4 bi-directional alignments, training with and without the possible (P) alignments.

4.3 Cross-validation

Using 10-fold cross-validation we are able to generate results on the whole of the Hansards test data which are comparable to previously published results. As the sentences in the test set were randomly chosen from the training corpus we can expect cross-validation to give an unbiased estimate of generalisation performance. These results are displayed in Table 5, using the possible (P) alignments for training. As the training set for each fold is roughly four times as big previous training set, we see a small improvement in AER.

model	precision	recall	f-score	AER
Fre ↔ Eng	94.6	92.2	93.4	6.47
Fre ↔ Eng (Model 4)	96.1	93.3	94.7	5.19

Table 5. 10-fold cross-validation results, with and without Model 4 features.

The final results of 6.47 and 5.19 with and without Model 4 features both exceed the performance of Model 4 alone. However the unsupervised Model 4 did not have access to the word-alignments in our training set. Callison-Burch et al. (2004) demonstrated that the GIZA++ models could be trained in a semi-supervised manner, leading to a slight decrease in error. To our knowledge, our AER of 5.19 is the best reported result, generative or discriminative, on this data set.

5 Related work

Recently, a number of discriminative word alignment models have been proposed, however these early models are typically very complicated with many proposing intractable problems which require heuristics for approximate inference (Liu et al., 2005; Moore, 2005).

An exception is Taskar et al. (2005) who presented a word matching model for discriminative alignment which they they were able to solve optimally. However, their model is limited to only providing one-to-one alignments. Also, no features were defined on label sequences, which reduced the model's ability to capture the strong monotonic relationships present between European language pairs. On the French-English Hansards task, using the same training/testing setup as our work, they achieve an AER of 5.4 with Model 4 features, and 10.7 without (compared to 5.29 and 6.99 for our CRF). One of the strengths of the CRF MAP estimation is the powerful smoothing offered by the prior, which allows us to avoid heuristics such as early stopping and hand weighted loss-functions that were needed for the maximum-margin model.

Liu et al. (2005) used a conditional log-linear model with similar features to those we have employed. They formulated a global model, without making a Markovian assumption, leading to the need for a sub-optimal heuristic search strategies.

Ittycheriah and Roukos (2005) trained a discriminative model on a corpus of ten thousand word aligned Arabic-English sentence pairs that outperformed a GIZA++ baseline. As with other approaches, they proposed a model which didn't allow a tractably optimal solution and thus had to resort to a heuristic beam search. They employed a log-linear model to learn the observation probabilities, while using a fixed transition distribution. Our CRF model allows both the observation and transition components of the model to be jointly optimised from the corpus.

6 Further work

The results presented in this paper were evaluated in terms of AER. While a low AER can be expected to improve end-to-end translation quality, this is may not necessarily be the case. Therefore, we plan to assess how the recall and precision characteristics of our model affect translation quality. The tradeoff between recall and precision may affect the quality and number of phrases extracted for a phrase translation table.

7 Conclusion

We have presented a novel approach for inducing word alignments from sentence aligned data. We showed how conditional random fields could be used for word alignment. These models allow for the use of arbitrary and overlapping features over the source and target sentences, making the most of small supervised training sets. Moreover, we showed how the CRF's inference and estimation methods allowed for efficient processing without sacrificing optimality, improving on previous heuristic based approaches.

On both French-English and Romanian-English we showed that many highly predictive features can be easily incorporated into the CRF, and demonstrated that with only a few hundred word-aligned training sentences, our model outperforms the generative Model 4 baseline. When no features are extracted from the sentence aligned corpus our model still achieves a low error rate. Furthermore, when we employ features derived from Model 4 alignments our CRF model achieves the highest reported results on both data sets.

Acknowledgements

Special thanks to Miles Osborne, Steven Bird, Timothy Baldwin and the anonymous reviewers for their feedback and insightful comments.

References

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

C. Callison-Burch, D. Talbot, and M. Osborne. 2004. Statistical machine translation with word- and sentence-aligned parallel corpora. In *Proceedings of ACL*, pages 175–182, Barcelona, Spain, July.

S. Chen and R. Rosenfeld. 1999. A survey of smoothing techniques for maximum entropy models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50.

L. R. Dice. 1945. Measures of the amount of ecologic association between species. *Journal of Ecology*, 26:297–302.

A. Ittycheriah and S. Roukos. 2005. A maximum entropy word aligner for Arabic-English machine translation. In *Proceedings of HLT-EMNLP*, pages 89–96, Vancouver, British Columbia, Canada, October.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 81–88, Edmonton, Alberta.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labelling sequence data. In *Proceedings of ICML*, pages 282–289.

Y. Liu, Q. Liu, and S. Lin. 2005. Log-linear models for word alignment. In *Proceedings of ACL*, pages 459–466, Ann Arbor.

R. Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of CoNLL*, pages 49–55.

J. Martin, R. Mihalcea, and T. Pedersen. 2005. Word alignment for languages with scarce resources. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pages 65–74, Ann Arbor, Michigan, June.

R. Mihalcea and T. Pedersen. 2003. An evaluation exercise for word alignment. In *Proceedings of HLT-NAACL 2003 Workshop, Building and Using Parrallel Texts: Data Driven Machine Translation and Beyond*, pages 1–6, Edmonton, Alberta.

R. C. Moore. 2005. A discriminative framework for bilingual word alignment. In *Proceedings of HLT-EMNLP*, pages 81–88, Vancouver, Canada.

F. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52.

F. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

F. Sha and F. Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL*, pages 213–220.

B. Taskar, S. Lacoste-Julien, and D. Klein. 2005. A discriminative matching approach to word alignment. In *Proceedings of HLT-EMNLP*, pages 73–80, Vancouver, British Columbia, Canada, October.

K. Toutanova, H. Tolga Ilhan, and C Manning. 2002. Extentions to HMM-based statistical word alignment models. In *Proceedings of EMNLP*, pages 87–94, Philadelphia, July.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proceedings of 16th Int. Conf. on Computational Linguistics*, pages 836–841.

Named Entity Transliteration with Comparable Corpora

Richard Sproat, Tao Tao, ChengXiang Zhai
University of Illinois at Urbana-Champaign, Urbana, IL, 61801
`rws@uiuc.edu, {taotao,czhai}@cs.uiuc.edu`

Abstract

In this paper we investigate Chinese-English name transliteration using *comparable corpora,* corpora where texts in the two languages deal in some of the same topics — and therefore share references to named entities — but are not translations of each other. We present two distinct methods for transliteration, one approach using phonetic transliteration, and the second using the temporal distribution of candidate pairs. Each of these approaches works quite well, but by combining the approaches one can achieve even better results. We then propose a novel score propagation method that utilizes the co-occurrence of transliteration pairs within document pairs. This propagation method achieves further improvement over the best results from the previous step.

1 Introduction

As part of a more general project on multilingual named entity identification, we are interested in the problem of name transliteration across languages that use different scripts. One particular issue is the discovery of named entities in "comparable" texts in multiple languages, where by comparable we mean texts that are about the same topic, but are *not* in general translations of each other. For example, if one were to go through an English, Chinese and Arabic newspaper on the same day, it is likely that the more important international events in various topics such as politics, business, science and sports, would each be covered in each of the newspapers. Names of the same persons, locations and so forth — which are often *transliterated* rather than translated — would be found in comparable stories across the three papers.[1] We wish to use this expectation to leverage transliteration, and thus the identification of named entities across languages. Our idea is that the occurrence of a cluster of names in, say, an English text, should be useful if we find a cluster of what looks like the same names in a Chinese or Arabic text.

An example of what we are referring to can be found in Figure 1. These are fragments of two stories from the June 8, 2001 Xinhua English and Chinese newswires, each covering an international women's badminton championship. Though these two stories are from the same newswire source, and cover the same event, they are *not* translations of each other. Still, not surprisingly, a lot of the names that occur in one, also occur in the other. Thus *(Camilla) Martin* shows up in the Chinese version as 马尔廷 *ma-er-ting*; *Judith Meulendijks* is 于 · 默伦迪克斯 *yu mo-lun-di-ke-si*; and *Mette Sorensen* is 迈 · 索伦森*mai su-lun-sen*. Several other correspondences also occur. While some of the transliterations are "standard" — thus Martin is conventionally transliterated as 马尔廷 *ma-er-ting* — many of them were clearly more novel, though all of them follow the standard Chinese conventions for transliterating foreign names.

These sample documents illustrate an important point: if a document in language L_1 has a set of names, and one finds a document in L_2 containing a set of names that look as if they could be transliterations of the names in the L_1 document, then this should boost one's confidence that the two sets of names are indeed transliterations of each other. We will demonstrate that this intuition is correct.

[1] Many names, particularly of organizations, may be translated rather than transliterated; the transliteration method we discuss here obviously will not account for such cases, though the time correlation and propagation methods we discuss will still be useful.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 73–80,
Sydney, July 2006. ©2006 Association for Computational Linguistics

Dai Yun Nips World No. 1 Martin to Shake off Olympic Shadow . . . In the day's other matches, second seed Zhou Mi overwhelmed Ling Wan Ting of Hong Kong, China 11-4, 11-4, Zhang Ning defeat Judith Meulendijks of Netherlands 11-2, 11-9 and third seed Gong Ruina took 21 minutes to eliminate Tine Rasmussen of Denmark 11-1, 11-1, enabling China to claim five quarterfinal places in the women's singles.

羽毛球世锦赛中国女单选手全部跻身八强 . . . 马尔廷还认为,她不可能连续战胜4个中国人,即使 . . . 三号种子龚睿那今晚以两个11:1轻取丹麦选手蒂・拉斯姆森,张宁在上午以11:2和11:9淘汰了荷兰的于・默伦迪克斯,周蜜在下午以11:4和11:1战胜了中国香港选手凌婉婷

Figure 1: Sample from two stories about an international women's badminton championship.

2 Previous Work

In previous work on Chinese named-entity transliteration — e.g. (Meng et al., 2001; Gao et al., 2004), the problem has been cast as the problem of producing, for a given Chinese name, an English equivalent such as one might need in a machine translation system. For example, for the name 维・威廉姆斯 *wei wei-lian-mu-si*, one would like to arrive at the English name *V(enus) Williams*. Common approaches include source-channel methods, following (Knight and Graehl, 1998) or maximum-entropy models.

Comparable corpora have been studied extensively in the literature (e.g.,(Fung, 1995; Rapp, 1995; Tanaka and Iwasaki, 1996; Franz et al., 1998; Ballesteros and Croft, 1998; Masuichi et al., 2000; Sadat et al., 2003)), but transliteration in the context of comparable corpora has not been well addressed.

The general idea of exploiting frequency correlations to acquire word translations from comparable corpora has been explored in several previous studies (e.g., (Fung, 1995; Rapp, 1995; Tanaka and Iwasaki, 1996)).Recently, a method based on Pearson correlation was proposed to mine word pairs from comparable corpora (Tao and Zhai, 2005), an idea similar to the method used in (Kay and Roscheisen, 1993) for sentence alignment. In our work, we adopt the method proposed in (Tao and Zhai, 2005) and apply it to the problem of transliteration. We also study several variations of the similarity measures.

Mining transliterations from multilingual web pages was studied in (Zhang and Vines, 2004); Our work differs from this work in that we use comparable corpora (in particular, news data) and leverage the time correlation information naturally available in comparable corpora.

3 Chinese Transliteration with Comparable Corpora

We assume that we have comparable corpora, consisting of newspaper articles in English and Chinese from the same day, or almost the same day. In our experiments we use data from the English and Chinese stories from the Xinhua News agency for about 6 months of 2001.[2] We assume that we have identified names for persons and locations—two types that have a strong tendency to be transliterated wholly or mostly phonetically—in the English text; in this work we use the named-entity recognizer described in (Li et al., 2004), which is based on the SNoW machine learning toolkit (Carlson et al., 1999).

To perform the transliteration task, we propose the following general three-step approach:

1. Given an English name, identify candidate Chinese character n-grams as possible transliterations.

2. Score each candidate based on how likely the candidate is to be a transliteration of the English name. We propose two different scoring methods. The first involves phonetic scoring, and the second uses the frequency profile of the candidate pair over time. We will show that each of these approaches works quite well, but by combining the approaches one can achieve even better results.

3. Propagate scores of all the candidate transliteration pairs globally based on their co-occurrences in document pairs in the comparable corpora.

The intuition behind the third step is the following. Suppose several high-confidence name transliteration pairs occur in a pair of English and Chinese documents. Intuitively, this would increase our confidence in the other plausible transliteration pairs in the same document pair. We thus propose a score propagation method to allow these high-confidence pairs to propagate some of their

[2] Available from the LDC via the English Gigaword (LDC2003T05) and Chinese Gigaword (LDC2003T09) corpora.

scores to other co-occurring transliteration pairs. As we will show later, such a propagation strategy can generally further improve the transliteration accuracy; in particular, it can further improve the already high performance from combining the two scoring methods.

3.1 Candidate Selection

The English named entity candidate selection process was already described above. Candidate Chinese transliterations are generated by consulting a list of characters that are frequently used for transliterating foreign names. As discussed elsewhere (Sproat et al., 1996), a subset of a few hundred characters (out of several thousand) tends to be used overwhelmingly for transliterating foreign names into Chinese. We use a list of 495 such characters, derived from various online dictionaries. A sequence of three or more characters from the list is taken as a possible name. If the character “ · ” occurs, which is frequently used to represent the space between parts of an English name, then at least one character to the left and right of this character will be collected, even if the character in question is not in the list of “foreign” characters.

Armed with the English and Chinese candidate lists, we then consider the pairing of every English candidate with every Chinese candidate. Obviously it would be impractical to do this for all of the candidates generated for, say, an entire year: we consider as plausible pairings those candidates that occur within a day of each other in the two corpora.

3.2 Candidate scoring based on pronunciation

We adopt a source-channel model for scoring English-Chinese transliteration pairs. In general, we seek to estimate $P(e|c)$, where e is a word in Roman script, and c is a word in Chinese script. Since Chinese transliteration is mostly based on pronunciation, we estimate $P(e'|c')$, where e' is the pronunciation of e and c' is the pronunciation of c. Again following standard practice, we decompose the estimate of $P(e'|c')$ as $P(e'|c') = \prod_i P(e'_i|c'_i)$. Here, e'_i is the ith subsequence of the English phone string, and c'_i is the ith subsequence of the Chinese phone string. Since Chinese transliteration attempts to match the syllable-sized characters to equivalent sounding spans of the English language, we fix the c'_i to be syllables, and let the e'_i range over all possible subsequences of the English phone string. For training data we have a small list of 721 names in Roman script and their Chinese equivalent.[3] Pronunciations for English words are obtained using the Festival text-to-speech system (Taylor et al., 1998); for Chinese, we use the standard pinyin transliteration of the characters. English-Chinese pairs in our training dictionary were aligned using the alignment algorithm from (Kruskal, 1999), and a hand-derived set of 21 rules-of-thumb: for example, we have rules that encode the fact that Chinese /l/ can correspond to English /r/, /n/ or /er/; and that Chinese /w/ may be used to represent /v/. Given that there are over 400 syllables in Mandarin (not counting tone) and each of these syllables can match a large number of potential English phone spans, this is clearly not enough training data to cover all the parameters, and so we use Good-Turing estimation to estimate probabilities for unseen correspondences. Since we would like to filter implausible transliteration pairs we are less lenient than standard estimation techniques in that we are willing to assign zero probability to some correspondences. Thus we set a hard rule that for an English phone span to correspond to a Chinese syllable, the initial phone of the English span must have been seen in the training data as corresponding to the initial of the Chinese syllable some minimum number of times. For consonant-initial syllables we set the minimum to 4. We omit further details of our estimation technique for lack of space. This phonetic correspondence model can then be used to score putative transliteration pairs.

3.3 Candidate Scoring based on Frequency Correlation

Names of the same entity that occur in different languages often have correlated frequency patterns due to common triggers such as a major event. Thus if we have comparable news articles over a sufficiently long time period, it is possible to exploit such correlations to learn the associations of names in different languages. The idea of exploiting frequency correlation has been well studied. (See the previous work section.) We adopt the method proposed in (Tao and Zhai, 2005), which

[3] The LDC provides a much larger list of transliterated Chinese-English names, but we did not use this here for two reasons. First, we have found it it be quite noisy. Secondly, we were interested in seeing how well one could do with a limited resource of just a few hundred names, which is a more realistic scenario for languages that have fewer resources than English and Chinese.

works as follows: We pool all documents in a single day to form a large pseudo-document. Then, for each transliteration candidate (both Chinese and English), we compute its frequency in each of those pseudo-documents and obtain a raw frequency vector. We further normalize the raw frequency vector so that it becomes a frequency distribution over all the time points (days). In order to compute the similarity between two distribution vectors, The Pearson correlation coefficient was used in (Tao and Zhai, 2005); here we also considered two other commonly used measures – **cosine** (Salton and McGill, 1983), and **Jensen-Shannon divergence** (Lin, 1991), though our results show that Pearson correlation coefficient performs better than these two other methods.

3.4 Score Propagation

In both scoring methods described above, scoring of each candidate transliteration pair is *independent* of the other. As we have noted, document pairs that contain lots of plausible transliteration pairs should be viewed as more plausible document pairs; at the same time, in such a situation we should also trust the putative transliteration pairs more. Thus these document pairs and transliteration pairs mutually "reinforce" each other, and this can be exploited to further optimize our transliteration scores by allowing transliteration pairs to propagate their scores to each other according to their co-occurrence strengths.

Formally, suppose the current generation of transliteration scores are $(e_i, c_i, w_i)\ \ i = 1, ..., n$, where (e_i, c_i) is a distinct pair of English and Chinese names. Note that although for any $i \neq j$, we have $(e_i, c_i) \neq (e_j, c_j)$, it is possible that $e_i = e_j$ or $c_i = c_j$ for some $i \neq j$. w_i is the transliteration score of (e_i, c_i).

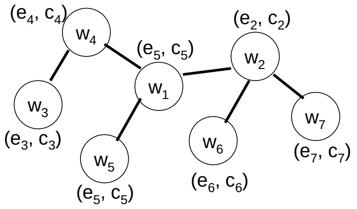


Figure 2: Graph representing transliteration pairs and cooccurence relations.

These pairs along with their co-occurrence relation computed based on our comparable corpora can be formally represented by a graph as shown in Figure 2. In such a graph, a node represents (e_i, c_i, w_i). An edge between (e_i, c_i, w_i) and (e_j, c_j, w_j) is constructed iff (e_i, c_i) and (e_j, c_j) co-occur in a certain document pair (E_t, C_t), i.e. there exists a document pair (E_t, C_t), such that $e_i, e_j \in E_t$ and $c_i, c_j \in C_t$. Given a node (e_i, c_i, w_i), we refer to all its directly-connected nodes as its "neighbors". The documents do not appear explicitly in the graph, but they implicitly affect the graph's topology and the weight of each edge. Our idea of score propagation can now be formulated as the following recursive equation for updating the scores of all the transliteration pairs.

$$w_i^{(k)} = \alpha \times w_i^{(k-1)} + (1-\alpha) \times \sum_{j \neq i, j=1}^{n} (w_j^{(k-1)} \times P(j|i)),$$

where $w_i^{(k)}$ is the new score of the pair (e_i, c_i) after an iteration, while $w_i^{(k-1)}$ is its old score before updating; $\alpha \in [0, 1]$ is a parameter to control the overall amount of propagation (when $\alpha = 1$, no propagation occurs); $P(j|i)$ is the conditional probability of propagating a score from node (e_j, c_j, w_j) to node (e_i, c_i, w_i).

We estimate $P(j|i)$ in two different ways: 1) The number of cooccurrences in the whole collection (Denote as CO). $P(j|i) = \frac{C(i,j)}{\sum_{j'} C(i,j')}$, where $C(i, j)$ is the cooccurrence count of (e_i, c_i) and (e_j, c_j); 2) A mutual information-based method (Denote as MI). $P(j|i) = \frac{MI(i,j)}{\sum_{j'} MI(i,j')}$, where $MI(i, j)$ is the mutual information of (e_i, c_i) and (e_j, c_j). As we will show, the CO method works better. Note that the transition probabilities between indirect neighbors are always 0. Thus propagation only happens between direct neighbors.

This formulation is very similar to PageRank, a link-based ranking algorithm for Web retrieval (Brin and Page, 1998). However, our motivation is propagating scores to exploit cooccurrences, so we do not necessarily want the equation to converge. Indeed, our results show that although the initial iterations always help improve accuracy, too many iterations actually would decrease the performance.

4 Evaluation

We use a comparable English-Chinese corpus to evaluate our methods for Chinese transliteration. We take one day's worth of comparable news articles (234 Chinese stories and 322 English stories), generate about 600 English names with the entity recognizer (Li et al., 2004) as described above, and find potential Chinese transliterations also as previously described. We generated 627 Chinese candidates. In principle, all these 600×627 pairs are potential transliterations. We then apply the phonetic and time correlation methods to score and rank all the candidate Chinese-English correspondences.

To evaluate the proposed transliteration methods quantitatively, we measure the accuracy of the ranked list by Mean Reciprocal Rank (MRR), a measure commonly used in information retrieval when there is precisely one correct answer (Kantor and Voorhees, 2000). The reciprocal rank is the reciprocal of the rank of the correct answer. For example, if the correct answer is ranked as the first, the reciprocal rank would be 1.0, whereas if it is ranked the second, it would be 0.5, and so forth. To evaluate the results for a set of English names, we take the mean of the reciprocal rank of each English name.

We attempted to create a complete set of answers for all the English names in our test set, but a small number of English names do not seem to have any standard transliteration according to the resources that we consulted. We ended up with a list of about 490 out of the 600 English names judged. We further notice that some answers (about 20%) are not in our Chinese candidate set. This could be due to two reasons: (1) The answer does not occur in the Chinese news articles we look at. (2) The answer is there, but our candidate generation method has missed it. In order to see more clearly how accurate each method is for ranking the candidates, we also compute the MRR for the subset of English names whose transliteration answers are in our candidate list. We distinguish the MRRs computed on these two sets of English names as “AllMRR” and “CoreMRR”.

Below we first discuss the results of each of the two methods. We then compare the two methods and discuss results from combining the two methods.

4.1 Phonetic Correspondence

We show sample results for the phonetic scoring method in Table 1. This table shows the 10 highest scoring transliterations for each Chinese character sequence based on all texts in the Chinese and English Xinhua newswire for the 13th of August, 2001. 8 out of these 10 are correct. For all the English names the MRR is 0.3, and for the core names it is 0.89. Thus on average, the correct answer, if it is included in our candidate list, is ranked mostly as the first one.

∗paris	佩雷斯	pei-lei-si	3.51
iraq	伊拉克	yi-la-ke	3.74
staub	斯塔伯	si-ta-bo	4.45
canada	加拿大	jia-na-da	4.85
belfast	贝尔法斯特	bei-er-fa-si-te	4.90
fischer	菲舍尔	fei-she-er	4.91
philippine	菲律宾	fei-lü-bin	4.97
lesotho	莱索托	lai-suo-two	5.12
∗tirana	铁路内	tye-lu-na	5.15
freeman	弗里曼	fu-li-man	5.26

Table 1: Ten highest-scoring matches for the Xinhua corpus for 8/13/01. The final column is the $-log\ P$ estimate for the transliteration. Starred entries are incorrect.

4.2 Frequency correlation

Similarity	AllMRR	CoreMRR
Pearson	0.1360	0.3643
Cosine	0.1141	0.3015
JS-div	0.0785	0.2016

Table 2: MRRs of the frequency correlation methods.

We proposed three similarity measures for the frequency correlation method, i.e., the Cosine, Pearson coefficient, and Jensen-Shannon divergence. In Table 2, we show their MRRs. Given that the only resource the method needs is comparable text documents over a sufficiently long period, these results are quite encouraging. For example, with Pearson correlation, when the Chinese transliteration of an English name is included in our candidate list, the correct answer is, on average, ranked at the 3rd place or better. The results thus show that the idea of exploiting frequency correlation does work. We also see that among the three similarity measures, Pearson correlation performs the best; it performs better than Cosine, which is better than JS-divergence.

Compared with the phonetic correspondence method, the performance of the frequency correlation method is in general much worse, which is not surprising, given the fact that terms may be correlated merely because they are topically related.

4.3 Combination of phonetic correspondence and frequency correlation

Method	AllMRR	CoreMRR
Phonetic	0.2999	0.8895
Freq	0.1360	0.3643
Freq+PhoneticFilter	0.3062	0.9083
Freq+PhoneticScore	0.3194	0.9474

Table 3: Effectiveness of combining the two scoring methods.

Since the two methods exploit complementary resources, it is natural to see if we can improve performance by combining the two methods. Indeed, intuitively the best candidate is the one that has a good pronunciation alignment as well as a correlated frequency distribution with the English name. We evaluated two strategies for combining the two methods. The first strategy is to use the phonetic model to filter out (clearly impossible) candidates and then use the frequency correlation method to rank the candidates. The second is to combine the scores of these two methods. Since the correlation coefficient has a maximum value of 1, we normalize the phonetic correspondence score by dividing all scores by the maximum score so that the maximum normalized value is also 1. We then take the average of the two scores and rank the candidates based on their average scores. Note that the second strategy implies the application of the first strategy.

The results of these two combination strategies are shown in Table 3 along with the results of the two individual methods. We see that both combination strategies are effective and the MRRs of the combined results are all better than those of the two individual methods. It is interesting to see that the benefit of applying the phonetic correspondence model as a filter is quite significant. Indeed, although the performance of the frequency correlation method alone is much worse than that of the phonetic correspondence method, when working on the subset of candidates passing the phonetic filter (i.e., those candidates that have a reasonable phonetic alignment with the English name), it can outperform the phonetic correspondence method. This once again indicates that exploiting the frequency correlation can be effective. When combining the scores of these two methods, we not only (implicitly) apply the phonetic filter, but also exploit the discriminative power provided by the phonetic correspondence scores and this is shown to bring in additional benefit, giving the best performance among all the methods.

4.4 Error Analysis

From the results above, we see that the MRRs for the core English names are substantially higher than those for all the English names. This means that our methods perform very well whenever we have the answer in our candidate list, but we have also missed the answers for many English names. The missing of an answer in the candidate list is thus a major source of errors. To further understand the upper bound of our method, we manually add the missing correct answers to our candidate set and apply all the methods to rank this augmented set of candidates. The performance is reported in Table 4 with the corresponding performance on the original candidate set. We see that, as expected, the performance on the augmented candidate list, which can be interpreted as an upper bound of our method, is indeed much better, suggesting that if we can somehow improve the candidate generation method to include the answers in the list, we can expect to significantly improve the performance for all the methods. This is clearly an interesting topic for further research. The relative performance of different methods on this augmented candidate list is roughly the same as on the original candidate list, except that the "Freq+PhoneticFilter" is slightly worse than that of the phonetic method alone, though it is still much better than the performance of the frequency correlation alone. One possible explanation may be that since these names do not necessarily occur in our comparable corpora, we may not have sufficient frequency observations for some of the names.

Method	ALLMRR	
	Original	Augmented
Phonetic	0.2999	0.7157
Freq	0.1360	0.3455
Freq+PhoneticFilter	0.3062	0.6232
Freq+PhoneticScore	0.3194	0.7338

Table 4: MRRs on the augmented candidate list.

Method	AllMRR			CoreMRR		
	init.	CO	MI	init.	CO	MI
Freq+PhoneticFilter	0.3171	0.3255	0.3255	0.9058	0.9372	0.9372
Freq+PhoneticScore	0.3290	0.3373	0.3392	0.9422	0.9659	0.9573

Table 5: Effectiveness of score propagation.

4.5 Experiments on score propagation

To demonstrate that score propagation can further help transliteration, we use the combination scores in Table 3 as the initial scores, and apply our propagation algorithm to iteratively update them. We remove the entries when they do not co-occur with others. There are 25 such English name candidates. Thus, the initial scores are actually slightly different from the values in Table 3. We show the new scores and the best propagation scores in Table 5. In the table, "init." refers to the initial scores. and "CO" and "MI" stand for best scores obtained using either the co-occurrence or mutual information method. While both methods result in gains, CO very slightly outperforms the MI approach. In the score propagation process, we introduce two additional parameters: the interpolation parameter α and the number of iterations k. Figure 3 and Figure 4 show the effects of these parameters. Intuitively, we want to preserve the initial score of a pair, but add a slight boost from its neighbors. Thus, we set α very close to 1 (0.9 and 0.95), and allow the system to perform 20 iterations. In both figures, the first few iterations certainly leverage the transliteration, demonstrating that the propagation method works. However, we observe that the performance drops when more iterations are used, presumably due to noise introduced from more distantly connected nodes. Thus, a relatively conservative approach is to choose a high α value, and run only a few iterations. Note, finally, that the CO method seems to be more stable than the MI method.

5 Conclusions and Future Work

In this paper we have discussed the problem of Chinese-English name transliteration as one component of a system to find matching names in comparable corpora. We have proposed two methods for transliteration, one that is more traditional and based on phonetic correspondences, and one that is based on word distributions and adopts methods from information retrieval. We have shown that both methods yield good results, and that even better results can be achieved by combining the methods. We have further showed that one can improve upon the combined model by using reinforcement via score propagation when transliteration pairs cluster together in document pairs.

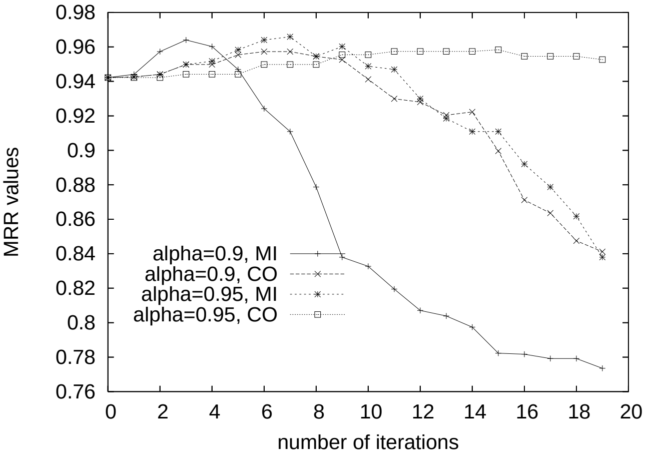


Figure 3: Propagation: Core items

The work we report is ongoing. We are investigating transliterations among several language pairs, and are extending these methods to Korean, Arabic, Russian and Hindi — see (Tao et al., 2006).

6 Acknowledgments

This work was funded by Dept. of the Interior contract NBCHC040176 (REFLEX). We also thank three anonymous reviewers for ACL06.

References

Lisa Ballesteros and W. Bruce Croft. 1998. Resolving ambiguity for cross-language retrieval. In *Research and Development in Information Retrieval*, pages 64–71.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30:107–117.

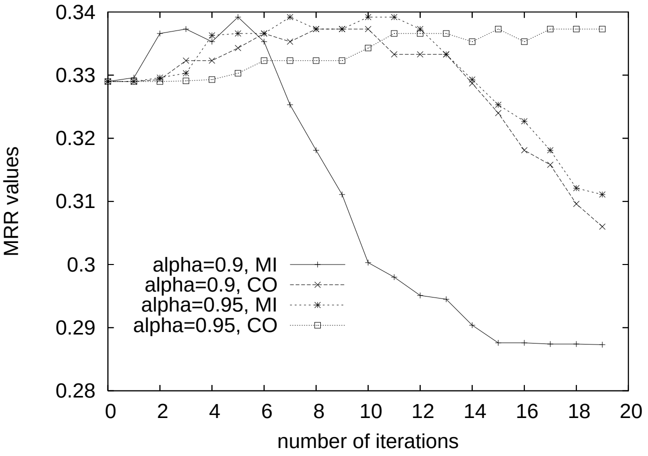


Figure 4: Propagation: All items

A. Carlson, C. Cumby, J. Rosen, and D. Roth. 1999. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC CS Dept.

Martin Franz, J. Scott McCarley, and Salim Roukos. 1998. Ad hoc and multilingual information retrieval at IBM. In *Text REtrieval Conference*, pages 104–115.

Pascale Fung. 1995. A pattern matching method for finding noun and proper noun translations from noisy parallel corpora. In *Proceedings of ACL 1995*, pages 236–243.

W. Gao, K.-F. Wong, and W. Lam. 2004. Phoneme-based transliteration of foreign names for OOV problem. In *IJCNLP*, pages 374–381, Sanya, Hainan.

P. Kantor and E. Voorhees. 2000. The TREC-5 confusion track: Comparing retrieval methods for scanned text. *Information Retrieval*, 2:165–176.

M. Kay and M. Roscheisen. 1993. Text translation alignment. *Computational Linguistics*, 19(1):75–102.

K. Knight and J. Graehl. 1998. Machine transliteration. *CL*, 24(4).

J. Kruskal. 1999. An overview of sequence comparison. In D. Sankoff and J. Kruskal, editors, *Time Warps, String Edits, and Macromolecules*, chapter 1, pages 1–44. CSLI, 2nd edition.

X. Li, P. Morie, and D. Roth. 2004. Robust reading: Identification and tracing of ambiguous names. In *NAACL-2004*.

J. Lin. 1991. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151.

H. Masuichi, R. Flournoy, S. Kaufmann, and S. Peters. 2000. A bootstrapping method for extracting bilingual text pairs.

H.M. Meng, W.K Lo, B. Chen, and K. Tang. 2001. Generating phonetic cognates to handle named entities in English-Chinese cross-languge spoken document retrieval. In *Proceedings of the Automatic Speech Recognition and Understanding Workshop*.

R. Rapp. 1995. Identifying word translations in non-parallel texts. In *Proceedings of ACL 1995*, pages 320–322.

Fatiha Sadat, Masatoshi Yoshikawa, and Shunsuke Uemura. 2003. Bilingual terminology acquisition from comparable corpora and phrasal translation to cross-language information retrieval. In *ACL '03*, pages 141–144.

G. Salton and M. McGill. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill.

R. Sproat, C. Shih, W. Gale, and N. Chang. 1996. A stochastic finite-state word-segmentation algorithm for Chinese. *CL*, 22(3).

K. Tanaka and H. Iwasaki. 1996. Extraction of lexical translation from non-aligned corpora. In *Proceedings of COLING 1996*.

Tao Tao and ChengXiang Zhai. 2005. Mining comparable bilingual text corpora for cross-language information integration. In *KDD'05*, pages 691–696.

Tao Tao, Su-Youn Yoon, Andrew Fister, Richard Sproat, and ChengXiang Zhai. 2006. Unsupervised named entity transliteration using temporal and phonetic correlation. In *EMNLP 2006*, Sydney, July.

P. Taylor, A. Black, and R. Caley. 1998. The architecture of the Festival speech synthesis system. In *Proceedings of the Third ESCA Workshop on Speech Synthesis*, pages 147–151, Jenolan Caves, Australia.

Ying Zhang and Phil Vines. 2004. Using the web for automated translation extraction in cross-language information retrieval. In *SIGIR '04*, pages 162–169.

Extracting Parallel Sub-Sentential Fragments from Non-Parallel Corpora

Dragos Stefan Munteanu
University of Southern California
Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA, 90292
dragos@isi.edu

Daniel Marcu
University of Southern California
Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA, 90292
marcu@isi.edu

Abstract

We present a novel method for extracting parallel sub-sentential fragments from comparable, non-parallel bilingual corpora. By analyzing potentially similar sentence pairs using a signal processing-inspired approach, we detect which segments of the source sentence are translated into segments in the target sentence, and which are not. This method enables us to extract useful machine translation training data even from very non-parallel corpora, which contain no parallel sentence pairs. We evaluate the quality of the extracted data by showing that it improves the performance of a state-of-the-art statistical machine translation system.

1 Introduction

Recently, there has been a surge of interest in the automatic creation of parallel corpora. Several researchers (Zhao and Vogel, 2002; Vogel, 2003; Resnik and Smith, 2003; Fung and Cheung, 2004a; Wu and Fung, 2005; Munteanu and Marcu, 2005) have shown how fairly good-quality parallel sentence pairs can be automatically extracted from comparable corpora, and used to improve the performance of machine translation (MT) systems. This work addresses a major bottleneck in the development of Statistical MT (SMT) systems: the lack of sufficiently large parallel corpora for most language pairs. Since comparable corpora exist in large quantities and for many languages – tens of thousands of words of news describing the same events are produced daily – the ability to exploit them for parallel data acquisition is highly beneficial for the SMT field.

Comparable corpora exhibit various degrees of parallelism. Fung and Cheung (2004a) describe corpora ranging from noisy parallel, to comparable, and finally to very non-parallel. Corpora from the last category contain "... disparate, very non-parallel bilingual documents that could either be on the same topic (on-topic) or not". This is the kind of corpora that we are interested to exploit in the context of this paper.

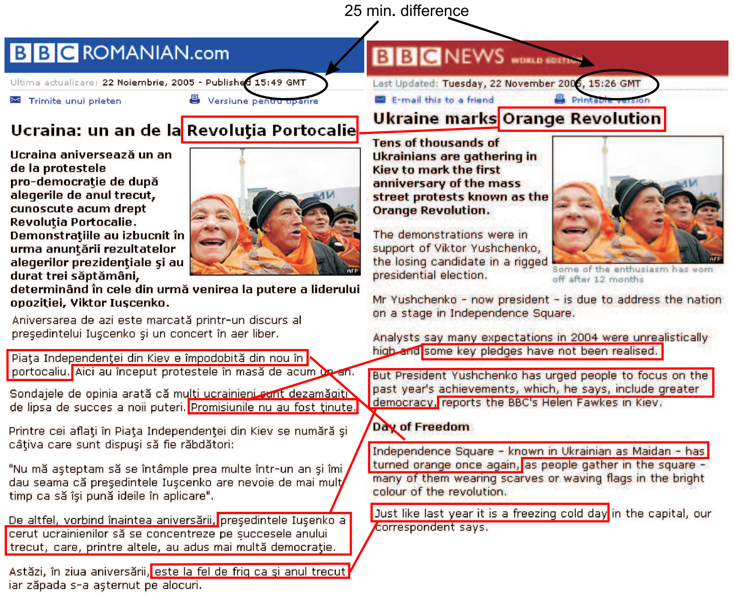
Existing methods for exploiting comparable corpora look for parallel data at the sentence level. However, we believe that very non-parallel corpora have none or few good sentence pairs; most of their parallel data exists at the sub-sentential level. As an example, consider Figure 1, which presents two news articles from the English and Romanian editions of the BBC. The articles report on the same event (the one-year anniversary of Ukraine's Orange Revolution), have been published within 25 minutes of each other, and express overlapping content.

Although they are "on-topic", these two documents are non-parallel. In particular, they contain no parallel sentence pairs; methods designed to extract full parallel sentences will not find any useful data in them. Still, as the lines and boxes from the figure show, some parallel fragments of data do exist; but they are present at the sub-sentential level.

In this paper, we present a method for extracting such parallel fragments from comparable corpora. Figure 2 illustrates our goals. It shows two sentences belonging to the articles in Figure 1, and highlights and connects their parallel fragments.

Although the sentences share some common meaning, each of them has content which is not translated on the other side. The English phrase *reports the BBC's Helen Fawkes in Kiev*, as well

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 81–88,
Sydney, July 2006. ©2006 Association for Computational Linguistics



BBC ROMANIAN.com

Ultima actualizare: 22 Noiembrie, 2005 - Published 15:49 GMT

Trimite unui prieten | Versiune pentru tiparire

Ucraina: un an de la Revoluţia Portocalie

Ucraina aniversează un an de la protestele pro-democraţie de după alegerile de anul trecut, cunoscute acum drept Revoluţia Portocalie. Demonstraţiile au izbucnit în urma anunţării rezultatelor alegerilor prezidenţiale şi au durat trei săptămâni, determinând în cele din urmă venirea la putere a liderului opoziţiei, Viktor Iuşcenko.

Aniversarea de azi este marcată printr-un discurs al preşedintelui Iuşcenko şi un concert în aer liber.

Piaţa Independenţei din Kiev e împodobită din nou în portocaliu. Aici au început protestele în masă de acum un an.

Sondajele de opinia arată că mulţi ucrainieni sunt dezamăgiţi de lipsa de succes a noii puteri. Promisiunile nu au fost ţinute.

Printre cei aflaţi în Piaţa Independenţei din Kiev se numără şi câţiva care sunt dispuşi să fie răbdători:

"Nu mă aşteptam să se întâmple prea multe într-un an şi îmi dau seama că preşedintele Iuşcenko are nevoie de mai mult timp ca să îşi pună ideile în aplicare".

De altfel, vorbind înaintea aniversării, preşedintele Iuşenko a cerut ucrainienilor să se concentreze pe succesele anului trecut, care, printre altele, au adus mai multă democraţie.

Astăzi, în ziua aniversării, este la fel de frig ca şi anul trecut iar zăpada s-a aşternut pe alocuri.

BBC NEWS WORLD EDITION

Last Updated: Tuesday, 22 November 2005, 15:26 GMT

E-mail this to a friend | Printable version

Ukraine marks Orange Revolution

Tens of thousands of Ukrainians are gathering in Kiev to mark the first anniversary of the mass street protests known as the Orange Revolution.

The demonstrations were in support of Viktor Yushchenko, the losing candidate in a rigged presidential election.

Some of the enthusiasm has worn off after 12 months

Mr Yushchenko - now president - is due to address the nation on a stage in Independence Square.

Analysts say many expectations in 2004 were unrealistically high and some key pledges have not been realised.

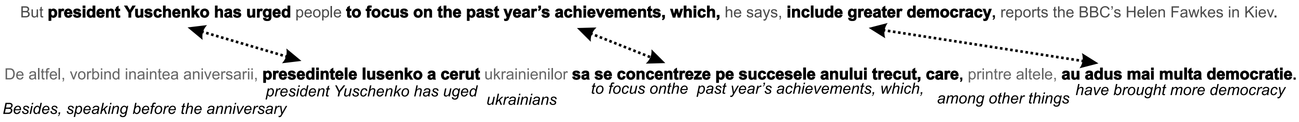
But President Yushchenko has urged people to focus on the past year's achievements, which, he says, include greater democracy, reports the BBC's Helen Fawkes in Kiev.

Day of Freedom

Independence Square - known in Ukrainian as Maidan - has turned orange once again, as people gather in the square - many of them wearing scarves or waving flags in the bright colour of the revolution.

Just like last year it is a freezing cold day in the capital, our correspondent says.

Figure 1: A pair of comparable, non-parallel documents



But **president Yuschenko has urged** people **to focus on the past year’s achievements, which,** he says, **include greater democracy**, reports the BBC’s Helen Fawkes in Kiev.

De altfel, vorbind inaintea aniversarii, **presedintele Iusenko a cerut** ucrainienilor **sa se concentreze pe succesele anului trecut, care,** printre altele, **au adus mai multa democratie.**

Besides, speaking before the anniversary
president Yuschenko has uged
ukrainians
to focus onthe past year’s achievements, which,
among other things
have brought more democracy

Figure 2: A pair of comparable sentences.

as the Romanian one *De altfel, vorbind inaintea aniversarii* have no translation correspondent, either in the other sentence or anywhere in the whole document. Since the sentence pair contains so much untranslated text, it is unlikely that any parallel sentence detection method would consider it useful. And, even if the sentences would be used for MT training, considering the amount of noise they contain, they might do more harm than good for the system’s performance. The best way to make use of this sentence pair is to extract and use for training just the translated (highlighted) fragments. This is the aim of our work.

Identifying parallel subsentential fragments is a difficult task. It requires the ability to recognize translational equivalence in very noisy environments, namely sentence pairs that express different (although overlapping) content. However, a good solution to this problem would have a strong impact on parallel data acquisition efforts. Enabling the exploitation of corpora that do not share parallel sentences would greatly increase the amount of comparable data that can be used for SMT.

2 Finding Parallel Sub-Sentential Fragments in Comparable Corpora

2.1 Introduction

The high-level architecture of our parallel fragment extraction system is presented in Figure 3.

The first step of the pipeline identifies document pairs that are similar (and therefore more likely to contain parallel data), using the Lemur information retrieval toolkit[1] (Ogilvie and Callan, 2001); each document in the source language is translated word-for-word and turned into a query, which is run against the collection of target language documents. The top 20 results are retrieved and paired with the query document. We then take all sentence pairs from these document pairs and run them through the second step in the pipeline, the candidate selection filter. This step discards pairs which have very few words that are translations of each other. To all remaining sentence pairs we apply the fragment detection method (described in Section 2.3), which produces the output of the system.

We use two probabilistic lexicons, learned au-

[1] `http://www-2.cs.cmu.edu/$\sim$lemur`

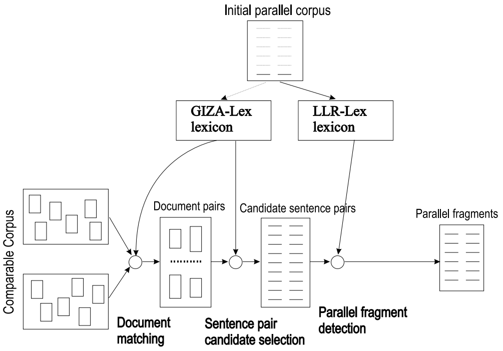


Figure 3: A Parallel Fragment Extraction System

tomatically from the same initial parallel corpus. The first one, *GIZA-Lex*, is obtained by running the GIZA++[2] implementation of the IBM word alignment models (Brown et al., 1993) on the initial parallel corpus. One of the characteristics of this lexicon is that each source word is associated with many possible translations. Although most of its high-probability entries are good translations, there are a lot of entries (of non-negligible probability) where the two words are at most related. As an example, in our *GIZA-Lex* lexicon, each source word has an average of 12 possible translations. This characteristic is useful for the first two stages of the extraction pipeline, which are not intended to be very precise. Their purpose is to accept most of the existing parallel data, and not too much of the non-parallel data; using such a lexicon helps achieve this purpose.

For the last stage, however, precision is paramount. We found empirically that when using *GIZA-Lex*, the incorrect correspondences that it contains seriously impact the quality of our results; we therefore need a cleaner lexicon. In addition, since we want to distinguish between source words that have a translation on the target side and words that do not, we also need a measure of the probability that two words are *not* translations of each other. All these are part of our second lexicon, *LLR-Lex*, which we present in detail in Section 2.2. Subsequently, in Section 2.3, we present our algorithm for detecting parallel sub-sentential fragments.

2.2 Using Log-Likelihood-Ratios to Estimate Word Translation Probabilities

Our method for computing the probabilistic translation lexicon *LLR-Lex* is based on the the Log-Likelihood-Ratio (LLR) statistic (Dunning, 1993), which has also been used by Moore (2004a; 2004b) and Melamed (2000) as a measure of word association. Generally speaking, this statistic gives a measure of the likelihood that two samples are not independent (i.e. generated by the same probability distribution). We use it to estimate the independence of pairs of words which cooccur in our parallel corpus.

If source word f and target word e are independent (i.e. they are not translations of each other), we would expect that $p(e|f) = p(e|\neg f) = p(e)$, i.e. the distribution of e given that f is present is the same as the distribution of e when f is not present. The LLR statistic gives a measure of the likelihood of this hypothesis. The LLR score of a word pair is low when these two distributions are very similar (i.e. the words are independent), and high otherwise (i.e. the words are strongly associated). However, high LLR scores can indicate either a positive association (i.e. $p(e|f) > p(e|\neg f)$) or a negative one; and we can distinguish between them by checking whether $p(e, f) > p(e)p(f)$.

Thus, we can split the set of cooccurring word pairs into positively and negatively associated pairs, and obtain a measure for each of the two association types. The first type of association will provide us with our (cleaner) lexicon, while the second will allow us to estimate probabilities of words *not* being translations of each other.

Before describing our new method more formally, we address the notion of word cooccurrence. In the work of Moore (2004a) and Melamed (2000), two words cooccur if they are present in a pair of aligned sentences in the parallel training corpus. However, most of the words from aligned sentences are actually unrelated; therefore, this is a rather weak notion of cooccurrence. We follow Resnik et. al (2001) and adopt a stronger definition, based not on sentence alignment but on word alignment: two words cooccur if they are linked together in the word-aligned parallel training corpus. We thus make use of the significant amount of knowledge brought in by the word alignment procedure.

We compute $LLR(e, f)$, the LLR score for words e and f, using the formula presented by Moore (2004b), which we do not repeat here due to lack of space. We then use these values to compute two conditional probability distributions: $P^+(e|f)$, the probability that source word f trans-

[2] `http://www.fjoch.com/GIZA++.html`

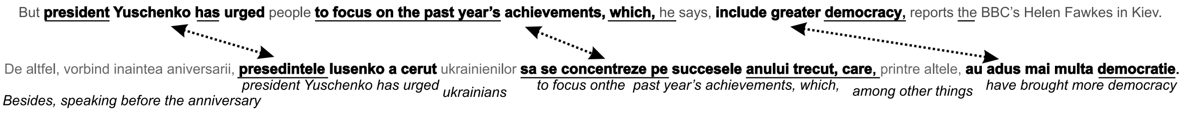

But president Yuschenko has urged people to focus on the past year's achievements, which, he says, include greater democracy, reports the BBC's Helen Fawkes in Kiev.
De altfel, vorbind inaintea aniversarii, presedintele Iusenko a cerut ukrainienilor sa se concentreze pe succesele anului trecut, care, printre altele, au adus mai multa democratie.
president Yuschenko has urged
ukrainians
to focus onthe past year's achievements, which,
among other things
have brought more democracy
Besides, speaking before the anniversary

Figure 4: Translated fragments, according to the lexicon.

lates into target word e, and $P^-(e|f)$, the probability that f does not translate into e. We obtain the distributions by normalizing the LLR scores for each source word.

The whole procedure follows:

- Word-align the parallel corpus. Following Och and Ney (2003), we run GIZA++ in both directions, and then symmetrize the alignments using the refined heuristic.

- Compute all LLR scores. There will be an LLR score for each pair of words which are linked at least once in the word-aligned corpus

- Classify all $LLR(e, f)$ as either $LLR^+(e, f)$ (positive association) if $p(e, f) > p(e)p(f)$, or $LLR^-(e, f)$ (negative association) otherwise.

- For each f, compute the normalizing factors $\sum_e LLR^+(e, f)$ and $\sum_e LLR^-(e, f)$.

- Divide all $LLR^+(e, f)$ terms by the corresponding normalizing factors to obtain $P^+(e|f)$.

- Divide all $LLR^-(e, f)$ terms by the corresponding normalizing factors to obtain $P^-(e|f)$.

In order to compute the $P(f|e)$ distributions, we reverse the source and target languages and repeat the procedure.

As we mentioned above, in *GIZA-Lex* the average number of possible translations for a source word is 12. In *LLR-Lex* that average is 5, which is a significant decrease.

2.3 Detecting Parallel Sub-Sentential Fragments

Intuitively speaking, our method tries to distinguish between source fragments that have a translation on the target side, and fragments that do not. In Figure 4 we show the sentence pair from Figure 2, in which we have underlined those words of each sentence that have a translation in the other sentence, according to our lexicon *LLR-Lex*. The phrases "*to focus on the past year's achievements, which,*" and "*sa se concentreze pe succesele anului trecut, care,*" are mostly underlined (the lexicon is unaware of the fact that "achievements" and "succesele" are in fact translations of each other, because "succesele" is a morphologically inflected form which does not cooccur with "achievements" in our initial parallel corpus). The rest of the sentences are mostly not underlined, although we do have occasional connections, some correct and some wrong. The best we can do in this case is to infer that these two phrases are parallel, and discard the rest. Doing this gains us some new knowledge: the lexicon entry *(achievements, succesele)*.

We need to quantify more precisely the notions of "mostly translated" and "mostly not translated". Our approach is to consider the target sentence as a numeric *signal*, where translated words correspond to positive values (coming from the P^+ distribution described in the previous Section), and the others to negative ones (coming from the P^- distribution). We want to retain the parts of the sentence where the signal is mostly positive. This can be achieved by applying a smoothing filter to the signal, and selecting those fragments of the sentence for which the corresponding filtered values are positive.

The details of the procedure are presented below, and also illustrated in Figure 5. Let the Romanian sentence be the source sentence F, and the English one be the target, E. We compute a word alignment $F \rightarrow E$ by greedily linking each English word with its best translation candidate from the Romanian sentence. For each of the linked target words, the corresponding signal value is the probability of the link (there can be at most one link for each target word). Thus, if target word e is linked to source word f, the signal value corresponding to e is $P^+(e|f)$ (the distribution described in Section 2.2), i.e. the probability that e is the translation of f.

For the remaining target words, the signal value should reflect the probability that they are not

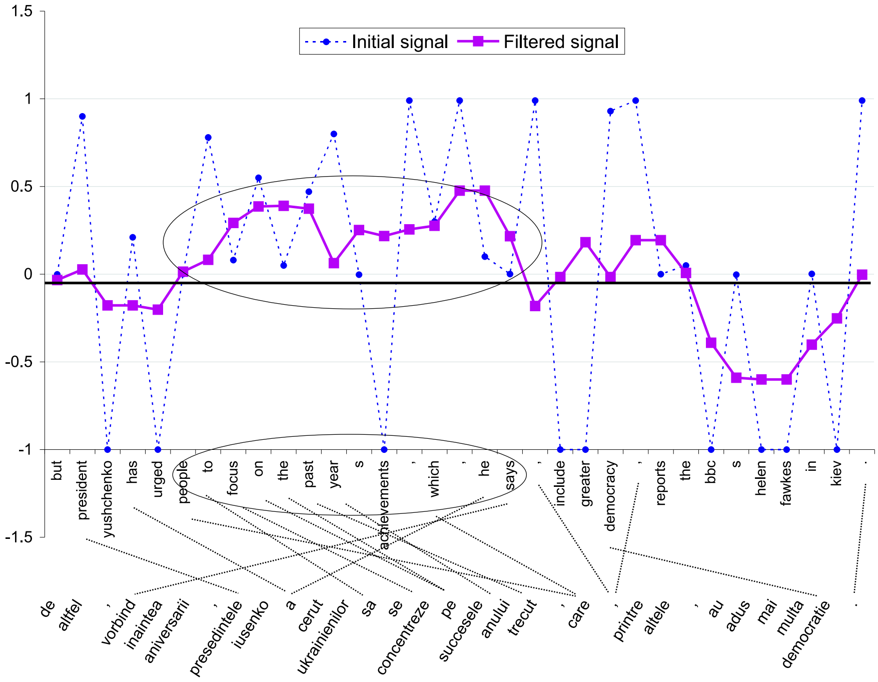


Figure 5: Our approach for detecting parallel fragments. The lower part of the figure shows the source and target sentence together with their alignment. Above are displayed the initial signal and the filtered signal. The circles indicate which fragments of the target sentence are selected by the procedure.

translated; for this, we employ the P^- distribution. Thus, for each non-linked target word e, we look for the source word least likely to be its non-translation: $f_0 = argmin_{f \in F} P^-(e|f)$. If f_0 exists, we set the signal value for e to $-P^-(e|f_0)$; otherwise, we set it to -1. This is the *initial signal*. We obtain the *filtered signal* by applying an averaging filter, which sets the value at each point to be the average of several values surrounding it. In our experiments, we use the surrounding 5 values, which produced good results on a development set. We then simply retain the "positive fragments" of E, i.e. those fragments for which the corresponding filtered signal values are positive.

However, this approach will often produce short "positive fragments" which are not, in fact, translated in the source sentence. An example of this is the fragment ", *reports*" from Figure 5, which although corresponds to positive values of the filtered signal, has no translation in Romanian. In an attempt to avoid such errors, we disregard fragments with less than 3 words.

We repeat the procedure in the other direction ($E \rightarrow F$) to obtain the fragments for f, and consider the resulting two text chunks as parallel.

For the sentence pair from Figure 5, our system will output the pair:

people to focus on the past year's achievements, which, he says

sa se concentreze pe succesele anului trecut, care, printre

3 Experiments

In our experiments, we compare our fragment extraction method (which we call *FragmentExtract*) with the sentence extraction approach of Munteanu and Marcu (2005) (*SentenceExtract*). All extracted datasets are evaluated by using them as additional MT training data and measuring their impact on the performance of the MT system.

3.1 Corpora

We perform experiments in the context of Romanian to English machine translation. We use two initial parallel corpora. One is the training data for the Romanian-English word alignment task from the Workshop on Building and Using Parallel Corpora[3] which has approximately 1M English words. The other contains additional data

[3] http://www.statmt.org/wpt05/

	Romanian		English	
Source	# articles	# tokens	# articles	# tokens
BBC	6k	2.5M	200k	118M
EZZ	183k	91M	14k	8.5M

Table 1: Sizes of our comparable corpora

from the Romanian translations of the European Union's acquis communautaire which we mined from the Web, and has about 10M English words.

We downloaded comparable data from three online news sites: the BBC, and the Romanian newspapers "Evenimentul Zilei" and "Ziua". The *BBC* corpus is precisely the kind of corpus that our method is designed to exploit. It is truly non-parallel; as our example from Figure 1 shows, even closely related documents have few or no parallel sentence pairs. Therefore, we expect that our extraction method should perform best on this corpus.

The other two sources are fairly similar, both in genre and in degree of parallelism, so we group them together and refer to them as the *EZZ* corpus. This corpus exhibits a higher degree of parallelism than the BBC one; in particular, it contains many article pairs which are literal translations of each other. Therefore, although our sub-sentence extraction method should produce useful data from this corpus, we expect the sentence extraction method to be more successful. Using this second corpus should help highlight the strengths and weaknesses of our approach.

Table 1 summarizes the relevant information concerning these corpora.

3.2 Extraction Experiments

On each of our comparable corpora, and using each of our initial parallel corpora, we apply both the fragment extraction and the sentence extraction method of Munteanu and Marcu (2005). In order to evaluate the importance of the *LLR-Lex* lexicon, we also performed fragment extraction experiments that do not use this lexicon, but only *GIZA-Lex*. Thus, for each initial parallel corpus and each comparable corpus, we extract three datasets: *FragmentExtract*, *SentenceExtract*, and *Fragment-noLLR*. The sizes of the extracted datasets, measured in million English tokens, are presented in Table 2.

Initial corpus	Source	FragmentExtract	SentenceExtract	Fragment-noLLR
1M	BBC	0.4M	0.3M	0.8M
1M	EZZ	6M	4M	8.1M
10M	BBC	1.3M	0.9M	2M
10M	EZZ	10M	7.9M	14.3M

Table 2: Sizes of the extracted datasets.

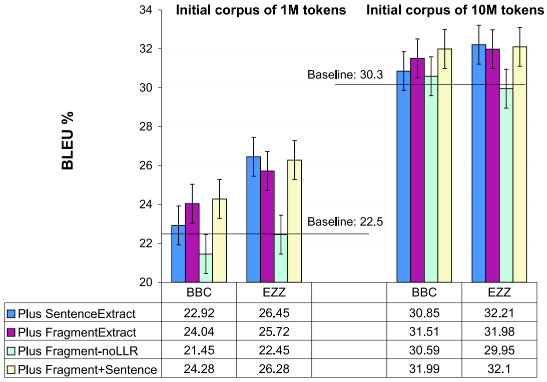
3.3 SMT Performance Results

We evaluate our extracted corpora by measuring their impact on the performance of an SMT system. We use the initial parallel corpora to train *Baseline* systems; and then train comparative systems using the initial corpora plus: the *FragmentExtract* corpora; the *SentenceExtract* corpora; and the *FragmentExtract-noLLR* corpora. In order to verify whether the fragment and sentence detection method complement each other, we also train a *Fragment+Sentence* system, on the initial corpus plus *FragmentExtract* and *SentenceExtract*.

All MT systems are trained using a variant of the alignment template model of Och and Ney (2004). All systems use the same 2 language models: one trained on 800 million English tokens, and one trained on the English side of all our parallel and comparable corpora. This ensures that differences in performance are caused only by differences in the parallel training data.

Our test data consists of news articles from the Time Bank corpus, which were translated into Romanian, and has 1000 sentences. Translation performance is measured using the automatic BLEU (Papineni et al., 2002) metric, on one reference translation. We report BLEU% numbers, i.e. we multiply the original scores by 100. The 95% confidence intervals of our scores, computed by bootstrap resampling (Koehn, 2004), indicate that a score increase of more than 1 BLEU% is statistically significant.

The scores are presented in Figure 6. On the *BBC* corpus, the fragment extraction method produces statistically significant improvements over the baseline, while the sentence extraction method does not. Training on both datasets together brings further improvements. This indicates that this corpus has few parallel sentences, and that by going to the sub-sentence level we make better use of it. On the *EZZ* corpus, although our method brings improvements in the BLEU score, the sen-



	BBC	EZZ		BBC	EZZ
🟦 Plus SentenceExtract	22.92	26.45		30.85	32.21
🟪 Plus FragmentExtract	24.04	25.72		31.51	31.98
🟩 Plus Fragment-noLLR	21.45	22.45		30.59	29.95
🟨 Plus Fragment+Sentence	24.28	26.28		31.99	32.1

Figure 6: SMT performance results

tence extraction method does better. Joining both extracted datasets does not improve performance; since most of the parallel data in this corpus exists at sentence level, the extracted fragments cannot bring much additional knowledge.

The *Fragment-noLLR* datasets bring no translation performance improvements; moreover, when the initial corpus is small (1M words) and the comparable corpus is noisy (BBC), the data has a negative impact on the BLEU score. This indicates that *LLR-Lex* is a higher-quality lexicon than *GIZA-Lex*, and an important component of our method.

4 Previous Work

Much of the work involving comparable corpora has focused on extracting word translations (Fung and Yee, 1998; Rapp, 1999; Diab and Finch, 2000; Koehn and Knight, 2000; Gaussier et al., 2004; Shao and Ng, 2004; Shinyama and Sekine, 2004). Another related research effort is that of Resnik and Smith (2003), whose system is designed to discover parallel document pairs on the Web.

Our work lies between these two directions; we attempt to discover parallelism at the level of fragments, which are longer than one word but shorter than a document. Thus, the previous research most relevant to this paper is that aimed at mining comparable corpora for parallel sentences.

The earliest efforts in this direction are those of Zhao and Vogel (2002) and Utiyama and Isahara (2003). Both methods extend algorithms designed to perform sentence alignment of parallel texts: they use dynamic programming to do sentence alignment of documents hypothesized to be similar. These approaches are only applicable to corpora which are at most “noisy-parallel”, i.e. contain documents which are fairly similar, both in content and in sentence ordering.

Munteanu and Marcu (2005) analyze sentence pairs in isolation from their context, and classify them as parallel or non-parallel. They match each source document with several target ones, and classify all possible sentence pairs from each document pair. This enables them to find sentences from fairly dissimilar documents, and to handle any amount of reordering, which makes the method applicable to truly comparable corpora.

The research reported by Fung and Cheung (2004a; 2004b), Cheung and Fung (2004) and Wu and Fung (2005) is aimed explicitly at “very non-parallel corpora”. They also pair each source document with several target ones and examine all possible sentence pairs; but the list of document pairs is not fixed. After one round of sentence extraction, the list is enriched with additional documents, and the system iterates. Thus, they include in the search document pairs which are dissimilar.

One limitation of all these methods is that they are designed to find only full sentences. Our methodology is the first effort aimed at detecting sub-sentential correspondences. This is a difficult task, requiring the ability to recognize translationally equivalent fragments even in non-parallel sentence pairs.

The work of Deng et. al (2006) also deals with sub-sentential fragments. However, they obtain parallel fragments from parallel sentence pairs (by chunking them and aligning the chunks appropriately), while we obtain them from comparable or non-parallel sentence pairs.

Since our approach can extract parallel data from texts which contain few or no parallel sentences, it greatly expands the range of corpora which can be usefully exploited.

5 Conclusion

We have presented a simple and effective method for extracting sub-sentential fragments from comparable corpora. We also presented a method for computing a probabilistic lexicon based on the LLR statistic, which produces a higher quality lexicon. We showed that using this lexicon helps improve the precision of our extraction method.

Our approach can be improved in several aspects. The signal filtering function is very simple; more advanced filters might work better, and eliminate the need of applying additional heuristics (such as our requirement that the extracted fragments have at least 3 words). The fact that the source and target signal are filtered separately is also a weakness; a joint analysis should produce better results. Despite the better lexicon, the greatest source of errors is still related to false word correspondences, generally involving punctuation and very common, closed-class words. Giving special attention to such cases should help get rid of these errors, and improve the precision of the method.

Acknowledgements

This work was partially supported under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022.

References

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Percy Cheung and Pascale Fung. 2004. Sentence alignment in parallel, comparable, and quasi-comparable corpora. In *LREC2004 Workshop*.

Yonggang Deng, Shankar Kumar, and William Byrne. 2006. Segmentation and alignment of parallel text for statistical machine translation. *Journal of Natural Language Engineering*. to appear.

Mona Diab and Steve Finch. 2000. A statistical word-level translation model for comparable corpora. In *RIAO 2000*.

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

Pascale Fung and Percy Cheung. 2004a. Mining very non-parallel corpora: Parallel sentence and lexicon extraction vie bootstrapping and EM. In *EMNLP 2004*, pages 57–63.

Pascale Fung and Percy Cheung. 2004b. Multi-level bootstrapping for extracting parallel sentences from a quasi-comparable corpus. In *COLING 2004*, pages 1051–1057.

Pascale Fung and Lo Yuen Yee. 1998. An IR approach for translating new words from nonparallel, comparable texts. In *ACL 1998*, pages 414–420.

Eric Gaussier, Jean-Michel Renders, Irina Matveeva, Cyril Goutte, and Herve Dejean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *ACL 2004*, pages 527–534.

Philipp Koehn and Kevin Knight. 2000. Estimating word translation probabilities from unrelated monolingual corpora using the EM algorithm. In *National Conference on Artificial Intelligence*, pages 711–715.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *EMNLP 2004*, pages 388–395.

I. Dan Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.

Robert C. Moore. 2004a. Improving IBM word-alignment model 1. In *ACL 2004*, pages 519–526.

Robert C. Moore. 2004b. On log-likelihood-ratios and the significance of rare events. In *EMNLP 2004*, pages 333–340.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4).

Franz Joseph Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Franz Joseph Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–450.

P. Ogilvie and J. Callan. 2001. Experiments using the Lemur toolkit. In *TREC 2001*, pages 103–108.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL 2002*, pages 311–318.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *ACL 1999*, pages 519–526.

Philip Resnik and Noah A. Smith. 2003. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380.

Philip Resnik, Douglas Oard, and Gina Lewow. 2001. Improved cross-language retrieval using backoff translation. In *HLT 2001*.

Li Shao and Hwee Tou Ng. 2004. Mining new word translations from comparable corpora. In *COLING 2004*, pages 618–624.

Yusuke Shinyama and Satoshi Sekine. 2004. Named entity discovery using comparable news articles. In *COLING 2004*, pages 848–853.

Masao Utiyama and Hitoshi Isahara. 2003. Reliable measures for aligning Japanese-English news articles and sentences. In *ACL 2003*, pages 72–79.

Stephan Vogel. 2003. Using noisy bilingual data for statistical machine translation. In *EACL 2003*, pages 175–178.

Dekai Wu and Pascale Fung. 2005. Inversion transduction grammar constraints for mining parallel sentences from quasi-comparable corpora. In *IJCNLP 2005*, pages 257–268.

Bing Zhao and Stephan Vogel. 2002. Adaptive parallel sentences mining from web bilingual news collection. In *2002 IEEE Int. Conf. on Data Mining*, pages 745–748.

Estimating Class Priors in Domain Adaptation for Word Sense Disambiguation

Yee Seng Chan and Hwee Tou Ng
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
{chanys,nght}@comp.nus.edu.sg

Abstract

Instances of a word drawn from different domains may have different sense priors (the proportions of the different senses of a word). This in turn affects the accuracy of word sense disambiguation (WSD) systems trained and applied on different domains. This paper presents a method to estimate the sense priors of words drawn from a new domain, and highlights the importance of using well calibrated probabilities when performing these estimations. By using well calibrated probabilities, we are able to estimate the sense priors effectively to achieve significant improvements in WSD accuracy.

1 Introduction

Many words have multiple meanings, and the process of identifying the correct meaning, or sense of a word in context, is known as word sense disambiguation (WSD). Among the various approaches to WSD, corpus-based supervised machine learning methods have been the most successful to date. With this approach, one would need to obtain a corpus in which each ambiguous word has been manually annotated with the correct sense, to serve as training data.

However, supervised WSD systems faced an important issue of domain dependence when using such a corpus-based approach. To investigate this, Escudero et al. (2000) conducted experiments using the DSO corpus, which contains sentences drawn from two different corpora, namely Brown Corpus (BC) and Wall Street Journal (WSJ). They found that training a WSD system on one part (BC or WSJ) of the DSO corpus and applying it to the other part can result in an accuracy drop of 12% to 19%. One reason for this is the difference in sense priors (i.e., the proportions of the different senses of a word) between BC and WSJ. For instance, the noun *interest* has these 6 senses in the DSO corpus: sense 1, 2, 3, 4, 5, and 8. In the BC part of the DSO corpus, these senses occur with the proportions: 34%, 9%, 16%, 14%, 12%, and 15%. However, in the WSJ part of the DSO corpus, the proportions are different: 13%, 4%, 3%, 56%, 22%, and 2%. When the authors assumed they knew the sense priors of each word in BC and WSJ, and adjusted these two datasets such that the proportions of the different senses of each word were the same between BC and WSJ, accuracy improved by 9%. In another work, Agirre and Martinez (2004) trained a WSD system on data which was automatically gathered from the Internet. The authors reported a 14% improvement in accuracy if they have an accurate estimate of the sense priors in the evaluation data and sampled their training data according to these sense priors. The work of these researchers showed that when the domain of the training data differs from the domain of the data on which the system is applied, there will be a decrease in WSD accuracy.

To build WSD systems that are portable across different domains, estimation of the sense priors (i.e., determining the proportions of the different senses of a word) occurring in a text corpus drawn from a domain is important. McCarthy et al. (2004) provided a partial solution by describing a method to predict the predominant sense, or the most frequent sense, of a word in a corpus. Using the noun *interest* as an example, their method will try to predict that sense 1 is the predominant sense in the BC part of the DSO corpus, while sense 4 is the predominant sense in the WSJ part of the

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 89–96,
Sydney, July 2006. ©2006 Association for Computational Linguistics

corpus.

In our recent work (Chan and Ng, 2005b), we directly addressed the problem by applying machine learning methods to automatically estimate the sense priors in the target domain. For instance, given the noun *interest* and the WSJ part of the DSO corpus, we attempt to estimate the proportion of each sense of *interest* occurring in WSJ and showed that these estimates help to improve WSD accuracy. In our work, we used naive Bayes as the training algorithm to provide posterior probabilities, or class membership estimates, for the instances in the target domain. These probabilities were then used by the machine learning methods to estimate the sense priors of each word in the target domain.

However, it is known that the posterior probabilities assigned by naive Bayes are not reliable, or not well calibrated (Domingos and Pazzani, 1996). These probabilities are typically too extreme, often being very near 0 or 1. Since these probabilities are used in estimating the sense priors, it is important that they are well calibrated.

In this paper, we explore the estimation of sense priors by first calibrating the probabilities from naive Bayes. We also propose using probabilities from another algorithm (logistic regression, which already gives well calibrated probabilities) to estimate the sense priors. We show that by using well calibrated probabilities, we can estimate the sense priors more effectively. Using these estimates improves WSD accuracy and we achieve results that are significantly better than using our earlier approach described in (Chan and Ng, 2005b).

In the following section, we describe the algorithm to estimate the sense priors. Then, we describe the notion of being well calibrated and discuss why using well calibrated probabilities helps in estimating the sense priors. Next, we describe an algorithm to calibrate the probability estimates from naive Bayes. Then, we discuss the corpora and the set of words we use for our experiments before presenting our experimental results. Next, we propose using the well calibrated probabilities of logistic regression to estimate the sense priors, and perform significance tests to compare our various results before concluding.

2 Estimation of Priors

To estimate the sense priors, or a priori probabilities of the different senses in a new dataset, we used a confusion matrix algorithm (Vucetic and Obradovic, 2001) and an EM based algorithm (Saerens et al., 2002) in (Chan and Ng, 2005b). Our results in (Chan and Ng, 2005b) indicate that the EM based algorithm is effective in estimating the sense priors and achieves greater improvements in WSD accuracy compared to the confusion matrix algorithm. Hence, to estimate the sense priors in our current work, we use the EM based algorithm, which we describe in this section.

2.1 EM Based Algorithm

Most of this section is based on (Saerens et al., 2002). Assume we have a set of labeled data D$_L$ with *n* classes and a set of *N* independent instances $(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ from a new data set. The likelihood of these *N* instances can be defined as:

$$\begin{aligned} L(\mathbf{x}_1, \ldots, \mathbf{x}_N) &= \prod_{k=1}^{N} p(\mathbf{x}_k) \\ &= \prod_{k=1}^{N} \left[\sum_{i=1}^{n} p(\mathbf{x}_k, \omega_i) \right] \\ &= \prod_{k=1}^{N} \left[\sum_{i=1}^{n} p(\mathbf{x}_k|\omega_i) p(\omega_i) \right] \quad (1) \end{aligned}$$

Assuming the within-class densities $p(\mathbf{x}_k|\omega_i)$, i.e., the probabilities of observing $\mathbf{x}_k$ given the class ω_i, do not change from the training set D$_L$ to the new data set, we can define: $p(\mathbf{x}_k|\omega_i) = p_L(\mathbf{x}_k|\omega_i)$. To determine the a priori probability estimates $\widehat{p}(\omega_i)$ of the new data set that will maximize the likelihood of (1) with respect to $p(\omega_i)$, we can apply the iterative procedure of the EM algorithm. In effect, through maximizing the likelihood of (1), we obtain the a priori probability estimates as a by-product.

Let us now define some notations. When we apply a classifier trained on D$_L$ on an instance $\mathbf{x}_k$ drawn from the new data set D$_U$, we get $\widehat{p}_L(\omega_i|\mathbf{x}_k)$, which we define as the probability of instance $\mathbf{x}_k$ being classified as class ω_i by the classifier trained on D$_L$. Further, let us define $\widehat{p}_L(\omega_i)$ as the a priori probabilities of class ω_i in D$_L$. This can be estimated by the class frequency of ω_i in D$_L$. We also define $\widehat{p}^{(s)}(\omega_i)$ and $\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k)$ as estimates of the new a priori and a posteriori probabilities at step *s* of the iterative EM procedure. Assuming we initialize $\widehat{p}^{(0)}(\omega_i) = \widehat{p}_L(\omega_i)$, then for each instance $\mathbf{x}_k$ in D$_U$ and each class ω_i, the EM algorithm provides the following iterative steps:

$$\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k) = \frac{\widehat{p}_L(\omega_i|\mathbf{x}_k)\frac{\widehat{p}^{(s)}(\omega_i)}{\widehat{p}_L(\omega_i)}}{\sum_{j=1}^{n}\widehat{p}_L(\omega_j|\mathbf{x}_k)\frac{\widehat{p}^{(s)}(\omega_j)}{\widehat{p}_L(\omega_j)}} \qquad (2)$$

$$\widehat{p}^{(s+1)}(\omega_i) = \frac{1}{N}\sum_{k=1}^{N}\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k) \qquad (3)$$

where Equation (2) represents the expectation E-step, Equation (3) represents the maximization M-step, and *N* represents the number of instances in D$_U$. Note that the probabilities $\widehat{p}_L(\omega_i|\mathbf{x}_k)$ and $\widehat{p}_L(\omega_i)$ in Equation (2) will stay the same throughout the iterations for each particular instance $\mathbf{x}_k$ and class ω_i. The new a posteriori probabilities $\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k)$ at step *s* in Equation (2) are simply the a posteriori probabilities in the conditions of the labeled data, $\widehat{p}_L(\omega_i|\mathbf{x}_k)$, weighted by the ratio of the new priors $\widehat{p}^{(s)}(\omega_i)$ to the old priors $\widehat{p}_L(\omega_i)$. The denominator in Equation (2) is simply a normalizing factor.

The a posteriori $\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k)$ and a priori probabilities $\widehat{p}^{(s)}(\omega_i)$ are re-estimated sequentially during each iteration *s* for each new instance $\mathbf{x}_k$ and each class ω_i, until the convergence of the estimated probabilities $\widehat{p}^{(s)}(\omega_i)$. This iterative procedure will increase the likelihood of (1) at each step.

2.2 Using A Priori Estimates

If a classifier estimates posterior class probabilities $\widehat{p}_L(\omega_i|\mathbf{x}_k)$ when presented with a new instance $\mathbf{x}_k$ from D$_U$, it can be directly adjusted according to estimated a priori probabilities $\widehat{p}(\omega_i)$ on D$_U$:

$$\widehat{p}_{adjust}(\omega_i|\mathbf{x}_k) = \frac{\widehat{p}_L(\omega_i|\mathbf{x}_k)\frac{\widehat{p}(\omega_i)}{\widehat{p}_L(\omega_i)}}{\sum_j \widehat{p}_L(\omega_j|\mathbf{x}_k)\frac{\widehat{p}(\omega_j)}{\widehat{p}_L(\omega_j)}} \qquad (4)$$

where $\widehat{p}_L(\omega_i)$ denotes the a priori probability of class ω_i from D$_L$ and $\widehat{p}_{adjust}(\omega_i|\mathbf{x}_k)$ denotes the adjusted predictions.

3 Calibration of Probabilities

In our eariler work (Chan and Ng, 2005b), the posterior probabilities assigned by a naive Bayes classifier are used by the EM procedure described in the previous section to estimate the sense priors $\widehat{p}(\omega_i)$ in a new dataset. However, it is known that the posterior probabilities assigned by naive Bayes are not well calibrated (Domingos and Pazzani, 1996).

It is important to use an algorithm which gives well calibrated probabilities, if we are to use the probabilities in estimating the sense priors. In this section, we will first describe the notion of being well calibrated before discussing why having well calibrated probabilities helps in estimating the sense priors. Finally, we will introduce a method used to calibrate the probabilities from naive Bayes.

3.1 Well Calibrated Probabilities

Assume for each instance $\mathbf{x}$, a classifier outputs a probability S$_{\omega_i}(\mathbf{x})$ between 0 and 1, of $\mathbf{x}$ belonging to class ω_i. The classifier is well-calibrated if the empirical class membership probability $p(\omega_i|\text{S}_{\omega_i}(\mathbf{x}) = t)$ converges to the probability value S$_{\omega_i}(\mathbf{x}) = t$ as the number of examples classified goes to infinity (Zadrozny and Elkan, 2002). Intuitively, if we consider all the instances to which the classifier assigns a probability S$_{\omega_i}(\mathbf{x})$ of say 0.6, then 60% of these instances should be members of class ω_i.

3.2 Being Well Calibrated Helps Estimation

To see why using an algorithm which gives well calibrated probabilities helps in estimating the sense priors, let us rewrite Equation (3), the M-step of the EM procedure, as the following:

$$\widehat{p}^{(s+1)}(\omega_i) = \frac{1}{N}\sum_{t\in S_{\omega_i}}\sum_{k\in\{q:S_{\omega_i}(\mathbf{x}_q)=t\}}\widehat{p}^{(s)}(\omega_i|\mathbf{x}_k) \qquad (5)$$

where S$_{\omega_i}$ = $\{t_1,\ldots,t_m\}$ denotes the set of posterior probability values for class ω_i, and S$_{\omega_i}(\mathbf{x}_q)$ denotes the posterior probability of class ω_i assigned by the classifier for instance $\mathbf{x}_q$.

Based on $t_1,\ldots,t_m$, we can imagine that we have m bins, where each bin is associated with a specific t value. Now, distribute all the instances in the new dataset D$_U$ into the m bins according to their posterior probabilities $S_{\omega_i}(\mathbf{x})$. Let B$_l$, for $l = 1, \ldots, m$, denote the set of instances in bin *l*.

Note that |B$_1$| $+\cdots+$ |B$_l$| $+\cdots+$ |B$_m$| = N. Now, let p_l denote the proportion of instances with true class label ω_i in B$_l$. Given a well calibrated algorithm, $p_l = t_l$ by definition and Equation (5) can be rewritten as:

$$\begin{aligned}\widehat{p}^{(s+1)}(\omega_i) &= \frac{1}{N}(t_1|\text{B}_1|+\cdots+t_m|\text{B}_m|)\\ &= \frac{1}{N}(p_1|\text{B}_1|+\cdots+p_m|\text{B}_m|)\\ &= \frac{N_{\omega_i}}{N} \qquad (6)\end{aligned}$$

```
Input: training set $(p_k, y_k)$ sorted in ascending order of $p_k$
Initialize $g_k = y_k$
While $\exists$ $k$ such that $g_j, \ldots, g_{k-1} > g_k, \ldots, g_l$, where $g_j = \cdots = g_{k-1}$ and $g_k = \cdots = g_l$ $(j < k \leq l)$
    Set $m = \frac{\sum_{q=j}^{l} g_q}{l-j+1}$
    Replace $g_j, \ldots, g_l$ with $m$
```

Figure 1: PAV algorithm.

where N_{ω_i} denotes the number of instances in D_U with true class label ω_i. Therefore, $\widehat{p}^{(s+1)}(\omega_i)$ reflects the proportion of instances in D_U with true class label ω_i. Hence, using an algorithm which gives well calibrated probabilities helps in the estimation of sense priors.

3.3 Isotonic Regression

Zadrozny and Elkan (2002) successfully used a method based on isotonic regression (Robertson et al., 1988) to calibrate the probability estimates from naive Bayes. To compute the isotonic regression, they used the pair-adjacent violators (PAV) (Ayer et al., 1955) algorithm, which we show in Figure 1. Briefly, what PAV does is to initially view each data value as a level set. While there are two adjacent sets that are out of order (i.e., the left level set is above the right one) then the sets are combined and the mean of the data values becomes the value of the new level set.

PAV works on binary class problems. In a binary class problem, we have a positive class and a negative class. Now, let $D = (p_k, \mathbf{x}_k), 1 \leq k \leq N$, where $\mathbf{x}_1, \ldots, \mathbf{x}_N$ represent *N* examples and p_k is the probability of $\mathbf{x}_k$ belonging to the positive class, as predicted by a classifier. Further, let y_k represent the true label of $\mathbf{x}_k$. For a binary class problem, we let $y_k = 1$ if $\mathbf{x}_k$ is a positive example and $y_k = 0$ if $\mathbf{x}_k$ is a negative example. The PAV algorithm takes in a set of (p_k, y_k), sorted in ascending order of p_k and returns a series of increasing step-values, where each step-value $g_{j,l}$ (denoted by *m* in Figure 1) is associated with a lowest boundary value p_j and a highest boundary value p_l. We performed 10-fold cross-validation on the training data to assign values to p_k. We then applied the PAV algorithm to obtain values for g_k. To obtain the calibrated probability estimate for a test instance $\mathbf{x}$, we find the boundary values p_j and p_l where $p_j \leq \mathrm{S}_{\omega_i}(\mathbf{x}) \leq p_l$ and assign $g_{j,l}$ as the calibrated probability estimate.

To apply PAV on a multiclass problem, we first reduce the problem into a number of binary class problems. For reducing a multiclass problem into a set of binary class problems, experiments in (Zadrozny and Elkan, 2002) suggest that the one-against-all approach works well. In one-against-all, a separate classifier is trained for each class ω_i, where examples belonging to class ω_i are treated as positive examples and all other examples are treated as negative examples. A separate classifier is then learnt for each binary class problem and the probability estimates from each classifier are calibrated. Finally, the calibrated binary-class probability estimates are combined to obtain multiclass probabilities, computed by a simple normalization of the calibrated estimates from each binary classifier, as suggested by Zadrozny and Elkan (2002).

4 Selection of Dataset

In this section, we discuss the motivations in choosing the particular corpora and the set of words used in our experiments.

4.1 DSO Corpus

The DSO corpus (Ng and Lee, 1996) contains 192,800 annotated examples for 121 nouns and 70 verbs, drawn from BC and WSJ. BC was built as a balanced corpus and contains texts in various categories such as religion, fiction, etc. In contrast, the focus of the WSJ corpus is on financial and business news. Escudero et al. (2000) exploited the difference in coverage between these two corpora to separate the DSO corpus into its BC and WSJ parts for investigating the domain dependence of several WSD algorithms. Following their setup, we also use the DSO corpus in our experiments.

The widely used SEMCOR (SC) corpus (Miller et al., 1994) is one of the few currently available manually sense-annotated corpora for WSD. SEMCOR is a subset of BC. Since BC is a balanced corpus, and training a classifier on a general corpus before applying it to a more specific corpus is a natural scenario, we will use examples from BC as training data, and examples from WSJ as evaluation data, or the target dataset.

4.2 Parallel Texts

Scalability is a problem faced by current supervised WSD systems, as they usually rely on manually annotated data for training. To tackle this problem, in one of our recent work (Ng et al., 2003), we had gathered training data from parallel texts and obtained encouraging results in our evaluation on the nouns of SENSEVAL-2 English lexical sample task (Kilgarriff, 2001). In another recent evaluation on the nouns of SENSEVAL-2 English all-words task (Chan and Ng, 2005a), promising results were also achieved using examples gathered from parallel texts. Due to the potential of parallel texts in addressing the issue of scalability, we also drew training data for our earlier sense priors estimation experiments (Chan and Ng, 2005b) from parallel texts. In addition, our parallel texts training data represents a natural domain difference with the test data of SENSEVAL-2 English lexical sample task, of which 91% is drawn from the British National Corpus (BNC).

As part of our experiments, we followed the experimental setup of our earlier work (Chan and Ng, 2005b), using the same 6 English-Chinese parallel corpora (*Hong Kong Hansards*, *Hong Kong News*, *Hong Kong Laws*, *Sinorama*, *Xinhua News*, and *English translation of Chinese Treebank*), available from Linguistic Data Consortium. To gather training examples from these parallel texts, we used the approach we described in (Ng et al., 2003) and (Chan and Ng, 2005b). We then evaluated our estimation of sense priors on the nouns of SENSEVAL-2 English lexical sample task, similar to the evaluation we conducted in (Chan and Ng, 2005b). Since the test data for the nouns of SENSEVAL-3 English lexical sample task (Mihalcea et al., 2004) were also drawn from BNC and represented a difference in domain from the parallel texts we used, we also expanded our evaluation to these SENSEVAL-3 nouns.

4.3 Choice of Words

Research by (McCarthy et al., 2004) highlighted that the sense priors of a word in a corpus depend on the domain from which the corpus is drawn. A change of predominant sense is often indicative of a change in domain, as different corpora drawn from different domains usually give different predominant senses. For example, the predominant sense of the noun *interest* in the BC part of the DSO corpus has the meaning “a sense of concern with and curiosity about someone or something”. In the WSJ part of the DSO corpus, the noun *interest* has a different predominant sense with the meaning “a fixed charge for borrowing money”, reflecting the business and finance focus of the WSJ corpus.

Estimation of sense priors is important when there is a significant change in sense priors between the training and target dataset, such as when there is a change in domain between the datasets. Hence, in our experiments involving the DSO corpus, we focused on the set of nouns and verbs which had different predominant senses between the BC and WSJ parts of the corpus. This gave us a set of 37 nouns and 28 verbs. For experiments involving the nouns of SENSEVAL-2 and SENSEVAL-3 English lexical sample task, we used the approach we described in (Chan and Ng, 2005b) of sampling training examples from the parallel texts using the natural (empirical) distribution of examples in the parallel texts. Then, we focused on the set of nouns having different predominant senses between the examples gathered from parallel texts and the evaluation data for the two SENSEVAL tasks. This gave a set of 6 nouns for SENSEVAL-2 and 9 nouns for SENSEVAL-3. For each noun, we gathered a maximum of 500 parallel text examples as training data, similar to what we had done in (Chan and Ng, 2005b).

5 Experimental Results

Similar to our previous work (Chan and Ng, 2005b), we used the supervised WSD approach described in (Lee and Ng, 2002) for our experiments, using the naive Bayes algorithm as our classifier. Knowledge sources used include parts-of-speech, surrounding words, and local collocations. This approach achieves state-of-the-art accuracy. All accuracies reported in our experiments are micro-averages over all test examples.

In (Chan and Ng, 2005b), we used a multiclass naive Bayes classifier (denoted by *NB*) for each word. Following this approach, we noted the WSD accuracies achieved without any adjustment, in the column *L* under *NB* in Table 1. The predictions $\widehat{p}_L(\omega_i|\mathbf{x}_k)$ of these naive Bayes classifiers are then used in Equation (2) and (3) to estimate the sense priors $\widehat{p}(\omega_i)$, before being adjusted by these estimated sense priors based on Equation (4). The resulting WSD accuracies after adjustment are listed in the column *EM*$_{NB}$ in Table 1, representing the WSD accuracies achievable by following the approach we described in (Chan and Ng, 2005b).

Next, we used the one-against-all approach to reduce each multiclass problem into a set of binary class problems. We trained a naive Bayes classifier for each binary problem and calibrated the probabilities from these binary classifiers. The WSD

Classifier	NB			NBcal		
Method	L	EM$_{NB}$	EM$_{LogR}$	L	EM$_{NBcal}$	EM$_{LogR}$
DSO nouns	44.5	46.1	46.6	45.8	47.0	51.1
DSO verbs	46.7	48.3	48.7	46.9	49.5	50.8
SE2 nouns	61.7	62.4	63.0	62.3	63.2	63.5
SE3 nouns	53.9	54.9	55.7	55.4	58.8	58.4

Table 1: Micro-averaged WSD accuracies using the various methods. The different naive Bayes classifiers are: multiclass naive Bayes (NB) and naive Bayes with calibrated probabilities (NBcal).

Dataset	True $-$ L	EM$_{NBcal}$ $-$ L	EM$_{LogR}$ $-$ L
DSO nouns	11.6	1.2 (10.3%)	5.3 (45.7%)
DSO verbs	10.3	2.6 (25.2%)	3.9 (37.9%)
SE2 nouns	3.0	0.9 (30.0%)	1.2 (40.0%)
SE3 nouns	3.7	3.4 (91.9%)	3.0 (81.1%)

Table 2: Relative accuracy improvement based on calibrated probabilities.

Dataset	EM$_{NB}$	EM$_{NBcal}$	EM$_{LogR}$
DSO nouns	0.621	0.586	0.293
DSO verbs	0.651	0.602	0.307
SE2 nouns	0.371	0.307	0.214
SE3 nouns	0.693	0.632	0.408

Table 3: KL divergence between the true and estimated sense distributions.

accuracies of these calibrated naive Bayes classifiers (denoted by *NBcal*) are given in the column *L* under *NBcal*.[1] The predictions of these classifiers are then used to estimate the sense priors $\widehat{p}(\omega_i)$, before being adjusted by these estimates based on Equation (4). The resulting WSD accuracies after adjustment are listed in column *EM*$_{NBcal}$ in Table 1.

The results show that calibrating the probabilities improves WSD accuracy. In particular, *EM*$_{NBcal}$ achieves the highest accuracy among the methods described so far. To provide a basis for comparison, we also adjusted the calibrated probabilities by the *true* sense priors $p(\omega_i)$ of the test data. The increase in WSD accuracy thus obtained is given in the column *True* $-$ *L* in Table 2. Note that this represents the maximum possible increase in accuracy achievable provided we know these *true* sense priors $p(\omega_i)$. In the column *EM*$_{NBcal}$ $-$ L in Table 2, we list the increase in WSD accuracy when adjusted by the sense priors $\widehat{p}(\omega_i)$ which were *automatically* estimated using the EM procedure. The relative improvements obtained with using $\widehat{p}(\omega_i)$ (compared against using $p(\omega_i)$) are given as percentages in brackets. As an example, according to Table 1 for the DSO verbs, EM$_{NBcal}$ gives an improvement of 49.5% $-$ 46.9% = 2.6% in WSD accuracy, and the relative improvement compared to using the *true* sense priors is 2.6/10.3 = 25.2%, as shown in Table 2.

6 Discussion

The experimental results show that the sense priors estimated using the calibrated probabilities of naive Bayes are effective in increasing the WSD accuracy. However, using a learning algorithm which already gives well calibrated posterior probabilities may be more effective in estimating the sense priors. One possible algorithm is logistic regression, which directly optimizes for getting approximations of the posterior probabilities. Hence, its probability estimates are already well calibrated (Zhang and Yang, 2004; Niculescu-Mizil and Caruana, 2005).

In the rest of this section, we first conduct experiments to estimate sense priors using the predictions of logistic regression. Then, we perform significance tests to compare the various methods.

6.1 Using Logistic Regression

We trained logistic regression classifiers and evaluated them on the 4 datasets. However, the WSD accuracies of these unadjusted logistic regression classifiers are on average about 4% lower than those of the unadjusted naive Bayes classifiers. One possible reason is that being a discriminative learner, logistic regression requires more training examples for its performance to catch up to, and possibly overtake the generative naive Bayes learner (Ng and Jordan, 2001).

Although the accuracy of logistic regression as a basic classifier is lower than that of naive Bayes, its predictions may still be suitable for estimating

[1] Though not shown, we also calculated the accuracies of these binary classifiers without calibration, and found them to be similar to the accuracies of the multiclass naive Bayes shown in the column *L* under *NB* in Table 1.

Method comparison	DSO nouns	DSO verbs	SE2 nouns	SE3 nouns
NB-EM$_{LogR}$ vs. NB-EM$_{NB}$	$\gg$	$\gg$	$\gg$	$\gg$
NBcal-EM$_{NBcal}$ vs. NB-EM$_{NB}$	$\sim$	$\gg$	$>$	$\gg$
NBcal-EM$_{NBcal}$ vs. NB-EM$_{LogR}$	$\sim$	$\gg$	$\sim$	$\gg$
NBcal-EM$_{LogR}$ vs. NB-EM$_{NB}$	$\gg$	$\gg$	$\gg$	$\gg$
NBcal-EM$_{LogR}$ vs. NB-EM$_{LogR}$	$\gg$	$\gg$	$\sim$	$\gg$
NBcal-EM$_{LogR}$ vs. NBcal-EM$_{NBcal}$	$\gg$	$\gg$	$\sim$	$\sim$

Table 4: Paired t-tests between the various methods for the 4 datasets.

sense priors. To gauge how well the sense priors are estimated, we measure the KL divergence between the true sense priors and the sense priors estimated by using the predictions of (uncalibrated) multiclass naive Bayes, calibrated naive Bayes, and logistic regression. These results are shown in Table 3 and the column *EM*$_{LogR}$ shows that using the predictions of logistic regression to estimate sense priors consistently gives the lowest KL divergence.

Results of the KL divergence test motivate us to use sense priors estimated by logistic regression on the predictions of the naive Bayes classifiers. To elaborate, we first use the probability estimates $\widehat{p}_L(\omega_i|\mathbf{x}_k)$ of logistic regression in Equations (2) and (3) to estimate the sense priors $\widehat{p}(\omega_i)$. These estimates $\widehat{p}(\omega_i)$ and the predictions $\widehat{p}_L(\omega_i|\mathbf{x}_k)$ of the calibrated naive Bayes classifier are then used in Equation (4) to obtain the adjusted predictions. The resulting WSD accuracy is shown in the column *EM*$_{LogR}$ under *NBcal* in Table 1. Corresponding results when the predictions $\widehat{p}_L(\omega_i|\mathbf{x}_k)$ of the multiclass naive Bayes is used in Equation (4), are given in the column *EM*$_{LogR}$ under *NB*. The relative improvements against using the true sense priors, based on the calibrated probabilities, are given in the column *EM*$_{LogR} - L$ in Table 2. The results show that the sense priors provided by logistic regression are in general effective in further improving the results. In the case of DSO nouns, this improvement is especially significant.

6.2 Significance Test

Paired t-tests were conducted to see if one method is significantly better than another. The t statistic of the difference between each test instance pair is computed, giving rise to a p value. The results of significance tests for the various methods on the 4 datasets are given in Table 4, where the symbols “$\sim$”, “$>$”, and “$\gg$” correspond to p-value > 0.05, (0.01, 0.05], and ≤ 0.01 respectively.

The methods in Table 4 are represented in the form *a1-a2*, where *a1* denotes adjusting the predictions of which classifier, and *a2* denotes how the sense priors are estimated. As an example, NBcal-EM$_{LogR}$ specifies that the sense priors estimated by logistic regression is used to adjust the predictions of the calibrated naive Bayes classifier, and corresponds to accuracies in column *EM*$_{LogR}$ under *NBcal* in Table 1. Based on the significance tests, the adjusted accuracies of EM$_{NB}$ and EM$_{NBcal}$ in Table 1 are significantly better than their respective unadjusted *L* accuracies, indicating that estimating the sense priors of a new domain via the EM approach presented in this paper significantly improves WSD accuracy compared to just using the sense priors from the old domain.

NB-EM$_{NB}$ represents our earlier approach in (Chan and Ng, 2005b). The significance tests show that our current approach of using calibrated naive Bayes probabilities to estimate sense priors, and then adjusting the calibrated probabilities by these estimates (NBcal-EM$_{NBcal}$) performs significantly better than NB-EM$_{NB}$ (refer to row 2 of Table 4). For DSO nouns, though the results are similar, the p value is a relatively low 0.06.

Using sense priors estimated by logistic regression further improves performance. For example, row 1 of Table 4 shows that adjusting the predictions of multiclass naive Bayes classifiers by sense priors estimated by logistic regression (NB-EM$_{LogR}$) performs significantly better than using sense priors estimated by multiclass naive Bayes (NB-EM$_{NB}$). Finally, using sense priors estimated by logistic regression to adjust the predictions of calibrated naive Bayes (NBcal-EM$_{LogR}$) in general performs significantly better than most other methods, achieving the best overall performance.

In addition, we implemented the unsupervised method of (McCarthy et al., 2004), which calculates a prevalence score for each sense of a word to predict the predominant sense. As in our earlier work (Chan and Ng, 2005b), we normalized the prevalence score of each sense to obtain estimated sense priors for each word, which we then used to adjust the predictions of our naive Bayes classifiers. We found that the WSD accuracies obtained with the method of (McCarthy et al., 2004) are on average 1.9% *lower* than our NBcal-EM$_{LogR}$ method, and the difference is statistically significant.

7 Conclusion

Differences in sense priors between training and target domain datasets will result in a loss of WSD accuracy. In this paper, we show that using well calibrated probabilities to estimate sense priors is important. By calibrating the probabilities of the naive Bayes algorithm, and using the probabilities given by logistic regression (which is already well calibrated), we achieved significant improvements in WSD accuracy over previous approaches.

References

Eneko Agirre and David Martinez. 2004. Unsupervised WSD based on automatically retrieved examples: The importance of bias. In *Proc. of EMNLP04*.

Miriam Ayer, H. D. Brunk, G. M. Ewing, W. T. Reid, and Edward Silverman. 1955. An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, 26(4).

Yee Seng Chan and Hwee Tou Ng. 2005a. Scaling up word sense disambiguation via parallel texts. In *Proc. of AAAI05*.

Yee Seng Chan and Hwee Tou Ng. 2005b. Word sense disambiguation with distribution estimation. In *Proc. of IJCAI05*.

Pedro Domingos and Michael Pazzani. 1996. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *Proc. of ICML-1996*.

Gerard Escudero, Lluis Marquez, and German Rigau. 2000. An empirical study of the domain dependence of supervised word sense disambiguation systems. In *Proc. of EMNLP/VLC00*.

Adam Kilgarriff. 2001. English lexical sample task description. In *Proc. of SENSEVAL-2*.

Yoong Keok Lee and Hwee Tou Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proc. of EMNLP02*.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant word senses in untagged text. In *Proc. of ACL04*.

Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. 2004. The senseval-3 english lexical sample task. In *Proc. of SENSEVAL-3*.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *Proc. of ARPA Human Language Technology Workshop*.

Andrew Y. Ng and Michael I. Jordan. 2001. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Proc. of NIPS14*.

Hwee Tou Ng and Hian Beng Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Proc. of ACL96*.

Hwee Tou Ng, Bin Wang, and Yee Seng Chan. 2003. Exploiting parallel texts for word sense disambiguation: An empirical study. In *Proc. of ACL03*.

Alexandru Niculescu-Mizil and Rich Caruana. 2005. Predicting good probabilities with supervised learning. In *Proc. of ICML05*.

Tim Robertson, F. T. Wright, and R. L. Dykstra. 1988. Chapter 1. Isotonic Regression. In *Order Restricted Statistical Inference*. John Wiley & Sons.

Marco Saerens, Patrice Latinne, and Christine Decaestecker. 2002. Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure. *Neural Computation*, 14(1).

Slobodan Vucetic and Zoran Obradovic. 2001. Classification on data with biased class distribution. In *Proc. of ECML01*.

Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proc. of KDD02*.

Jian Zhang and Yiming Yang. 2004. Probabilistic score estimation with piecewise logistic regression. In *Proc. of ICML04*.

Ensemble Methods for Unsupervised WSD

Samuel Brody
School of Informatics
University of Edinburgh
s.brody@sms.ed.ac.uk

Roberto Navigli
Dipartimento di Informatica
Università di Roma “La Sapienza”
navigli@di.uniroma1.it

Mirella Lapata
School of Informatics
University of Edinburgh
mlap@inf.ed.ac.uk

Abstract

Combination methods are an effective way of improving system performance. This paper examines the benefits of system combination for unsupervised WSD. We investigate several voting- and arbiter-based combination strategies over a diverse pool of unsupervised WSD systems. Our combination methods rely on predominant senses which are derived automatically from raw text. Experiments using the SemCor and Senseval-3 data sets demonstrate that our ensembles yield significantly better results when compared with state-of-the-art.

1 Introduction

Word sense disambiguation (WSD), the task of identifying the intended meanings (senses) of words in context, holds promise for many NLP applications requiring broad-coverage language understanding. Examples include summarization, question answering, and text simplification. Recent studies have also shown that WSD can benefit machine translation (Vickrey et al., 2005) and information retrieval (Stokoe, 2005).

Given the potential of WSD for many NLP tasks, much work has focused on the computational treatment of sense ambiguity, primarily using data-driven methods. Most accurate WSD systems to date are supervised and rely on the availability of training data, i.e., corpus occurrences of ambiguous words marked up with labels indicating the appropriate sense given the context (see Mihalcea and Edmonds 2004 and the references therein). A classifier automatically learns disambiguation cues from these hand-labeled examples.

Although supervised methods typically achieve better performance than unsupervised alternatives, their applicability is limited to those words for which sense labeled data exists, and their accuracy is strongly correlated with the amount of labeled data available (Yarowsky and Florian, 2002). Furthermore, obtaining manually labeled corpora with word senses is costly and the task must be repeated for new domains, languages, or sense inventories. Ng (1997) estimates that a high accuracy domain independent system for WSD would probably need a corpus of about 3.2 million sense tagged words. At a throughput of one word per minute (Edmonds, 2000), this would require about 27 person-years of human annotation effort.

This paper focuses on unsupervised methods which we argue are useful for broad coverage sense disambiguation. Unsupervised WSD algorithms fall into two general classes: those that perform token-based WSD by exploiting the similarity or relatedness between an ambiguous word and its context (e.g., Lesk 1986); and those that perform type-based WSD, simply by assigning all instances of an ambiguous word its most frequent (i.e., predominant) sense (e.g., McCarthy et al. 2004; Galley and McKeown 2003). The predominant senses are automatically acquired from raw text without recourse to manually annotated data. The motivation for assigning all instances of a word to its most prevalent sense stems from the observation that current supervised approaches rarely outperform the simple heuristic of choosing the most common sense in the training data, despite taking local context into account (Hoste et al., 2002). Furthermore, the approach allows sense inventories to be tailored to specific domains.

The work presented here evaluates and compares the performance of well-established unsupervised WSD algorithms. We show that these algorithms yield sufficiently diverse outputs, thus motivating the use of combination methods for improving WSD performance. While combination approaches have been studied previously for supervised WSD (Florian et al., 2002), their use in an unsupervised setting is, to our knowledge, novel. We examine several existing and novel combination methods and demonstrate that our combined systems consistently outperform the

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 97–104,
Sydney, July 2006. ©2006 Association for Computational Linguistics

state-of-the-art (e.g., McCarthy et al. 2004). Importantly, our WSD algorithms and combination methods do not make use of training material in any way, nor do they use the first sense information available in WordNet.

In the following section, we briefly describe the unsupervised WSD algorithms considered in this paper. Then, we present a detailed comparison of their performance on SemCor (Miller et al., 1993). Next, we introduce our system combination methods and report on our evaluation experiments. We conclude the paper by discussing our results.

2 The Disambiguation Algorithms

In this section we briefly describe the unsupervised WSD algorithms used in our experiments. We selected methods that vary along the following dimensions: (a) the type of WSD performed (i.e., token-based vs. type-based), (b) the representation and size of the context surrounding an ambiguous word (i.e., graph-based vs. word-based, document vs. sentence), and (c) the number and type of semantic relations considered for disambiguation. We base most of our discussion below on the WordNet sense inventory; however, the approaches are not limited to this particular lexicon but could be adapted for other resources with traditional dictionary-like sense definitions and alternative structure.

Extended Gloss Overlap Gloss Overlap was originally introduced by Lesk (1986) for performing token-based WSD. The method assigns a sense to a target word by comparing the dictionary definitions of each of its senses with those of the words in the surrounding context. The sense whose definition has the highest overlap (i.e., words in common) with the context words is assumed to be the correct one. Banerjee and Pedersen (2003) augment the dictionary definition (gloss) of each sense with the glosses of related words and senses. The *extended glosses* increase the information available in estimating the overlap between ambiguous words and their surrounding context.

The range of relationships used to extend the glosses is a parameter, and can be chosen from any combination of WordNet relations. For every sense s_k of the target word we estimate:

$$SenseScore(s_k) = \sum_{Rel \in Relations} Overlap(context, Rel(s_k))$$

where *context* is a simple (space separated) concatenation of all words w_i for $-n \leq i \leq n, i \neq 0$ in a context window of length $\pm n$ around the target word w_0. The overlap scoring mechanism is also parametrized and can be adjusted to take the into account gloss length or to ignore function words.

Distributional and WordNet Similarity McCarthy et al. (2004) propose a method for automatically ranking the senses of ambiguous words from raw text. Key in their approach is the observation that distributionally similar neighbors often provide cues about a word's senses. Assuming that a set of neighbors is available, sense ranking is equivalent to quantifying the degree of similarity among the neighbors and the sense descriptions of the polysemous word.

Let $N(w) = \{n_1, n_2, \dots, n_k\}$ be the k most (distributionally) similar words to an ambiguous target word w and $senses(w) = \{s_1, s_2, \dots s_n\}$ the set of senses for w. For each sense s_i and for each neighbor n_j, the algorithm selects the neighbor's sense which has the highest WordNet similarity score (*wnss*) with regard to s_i. The ranking score of sense s_i is then increased as a function of the WordNet similarity score and the distributional similarity score (*dss*) between the target word and the neighbor:

$$RankScore(s_i) = \sum_{n_j \in N_w} dss(w, n_j) \frac{wnss(s_i, n_j)}{\sum\limits_{s'_i \in senses(w)} wnss(s'_i, n_j)}$$

where $wnss(s_i, n_j) = \max\limits_{ns_x \in senses(n_j)} wnss(s_i, ns_x)$.

The predominant sense is simply the sense with the highest ranking score (*RankScore*) and can be consequently used to perform type-based disambiguation. The method presented above has four parameters: (a) the semantic space model representing the distributional properties of the target words (it is acquired from a large corpus representative of the domain at hand and can be augmented with syntactic relations such as subject or object), (b) the measure of distributional similarity for discovering neighbors (c) the number of neighbors that the ranking score takes into account, and (d) the measure of sense similarity.

Lexical Chains Lexical cohesion is often represented via lexical chains, i.e., sequences of related words spanning a topical text unit (Morris and Hirst, 1991). Algorithms for computing lexical chains often perform WSD before inferring which words are semantically related. Here we describe one such disambiguation algorithm, proposed by Galley and McKeown (2003), while omitting the details of creating the lexical chains themselves.

Galley and McKeown's (2003) method consists of two stages. First, a graph is built representing all possible interpretations of the target words in question. The text is processed sequentially, comparing each word against all words previously read. If a relation exists between the senses of the current word and any possible sense of a previous word, a connection is formed between the appropriate words and senses. The strength of the connection is a function of the type of relationship and of the distance between the words in the text (in terms of words, sentences and paragraphs). Words are represented as nodes in the graph and semantic relations as weighted edges. Again, the set of relations being considered is a parameter that can be tuned experimentally.

In the disambiguation stage, all occurrences of a given word are collected together. For each sense of a target word, the strength of all connections involving that sense are summed, giving that sense a unified score. The sense with the highest unified score is chosen as the correct sense for the target word. In subsequent stages the actual connections comprising the winning unified score are used as a basis for computing the lexical chains.

The algorithm is based on the "one sense per discourse" hypothesis and uses information from every occurrence of the ambiguous target word in order to decide its appropriate sense. It is therefore a type-based algorithm, since it tries to determine the sense of the word in the entire document/discourse at once, and not separately for each instance.

Structural Semantic Interconnections Inspired by lexical chains, Navigli and Velardi (2005) developed Structural Semantic Interconnections (SSI), a WSD algorithm which makes use of an extensive lexical knowledge base. The latter is primarily based on WordNet and its standard relation set (i.e., hypernymy, meronymy, antonymy, similarity, nominalization, pertainymy) but is also enriched with collocation information representing semantic relatedness between sense pairs. Collocations are gathered from existing resources (such as the Oxford Collocations, the Longman Language Activator, and collocation web sites). Each collocation is mapped to the WordNet sense inventory in a semi-automatic manner (Navigli, 2005) and transformed into a *relatedness* edge.

Given a local word context $C = \{w_1, ..., w_n\}$, SSI builds a graph $G = (V, E)$ such that $V = \bigcup_{i=1}^{n} senses(w_i)$ and $(s, s') \in E$ if there is at least one interconnection j between s (a sense of the word) and s' (a sense of its context) in the lexical knowledge base. The set of valid interconnections is determined by a manually-created context-free grammar consisting of a small number of rules. Valid interconnections are computed in advance on the lexical database, not at runtime.

Method	WSD	Context	Relations
LexChains	types	document	first-order
Overlap	tokens	sentence	first-order
Similarity	types	corpus	higher-order
SSI	tokens	sentence	higher-order

Table 1: Properties of the WSD algorithms

Disambiguation is performed in an iterative fashion. At each step, for each sense s of a word in $\mathcal{C}$ (the set of senses of words yet to be disambiguated), SSI determines the degree of connectivity between s and the other senses in $\mathcal{C}$:

$$SSIScore(s) = \frac{\sum\limits_{s' \in \mathcal{C} \setminus \{s\}} \sum\limits_{j \in Interconn(s,s')} \frac{1}{length(j)}}{\sum\limits_{s' \in \mathcal{C} \setminus \{s\}} |Interconn(s,s')|}$$

where $Interconn(s, s')$ is the set of interconnections between senses s and s'. The contribution of a single interconnection is given by the reciprocal of its length, calculated as the number of edges connecting its ends. The overall degree of connectivity is then normalized by the number of contributing interconnections. The highest ranking sense s of word w_i is chosen and the senses of w_i are removed from the context $\mathcal{C}$. The procedure terminates when either $\mathcal{C}$ is the empty set or there is no sense such that its *SSIScore* exceeds a fixed threshold.

Summary The properties of the different WSD algorithms just described are summarized in Table 1. The methods vary in the amount of data they employ for disambiguation. SSI and Extended Gloss Overlap (Overlap) rely on sentence-level information for disambiguation whereas McCarthy et al. (2004) (Similarity) and Galley and McKeown (2003) (LexChains) utilize the entire document or corpus. This enables the accumulation of large amounts of data regarding the ambiguous word, but does not allow separate consideration of each individual occurrence of that word. LexChains and Overlap take into account a restricted set of semantic relations (paths of length one) between any two words in the whole document, whereas SSI and Similarity use a wider set of relations.

3 Experiment 1: Comparison of Unsupervised Algorithms for WSD

3.1 Method

We evaluated the disambiguation algorithms outlined above on two tasks: predominant sense acquisition and token-based WSD. As previously explained, Overlap and SSI were not designed for acquiring predominant senses (see Table 1), but a token-based WSD algorithm can be trivially modified to acquire predominant senses by disambiguating every occurrence of the target word in context and selecting the sense which was chosen most frequently. Type-based WSD algorithms simply tag all occurrences of a target word with its predominant sense, disregarding the surrounding context.

Our first set of experiments was conducted on the SemCor corpus, on the same 2,595 polysemous nouns (53,674 tokens) used as a test set by McCarthy et al. (2004). These nouns were attested in SemCor with a frequency > 2 and occurred in the British National Corpus (BNC) more than 10 times. We used the WordNet 1.7.1 sense inventory.

The following notation describes our evaluation measures: W is the set of all noun types in the SemCor corpus ($|W| = 2,595$), and W_f is the set of noun types with a dominant sense. $senses(w)$ is the set of senses for noun type w, while $f_s(w)$ and $f_m(w)$ refer to w's first sense according to the SemCor gold standard and our algorithms, respectively. Finally, $T(w)$ is the set of tokens of w and $senses_s(t)$ denotes the sense assigned to token t according to SemCor.

We first measure how well our algorithms can identify the predominant sense, if one exists:

$$Acc_{ps} = \frac{|\{w \in W_f \mid f_s(w) = f_m(w)\}|}{|W_f|}$$

A baseline for this task can be easily defined for each word type by selecting a sense at random from its sense inventory and assuming that this is the predominant sense:

$$Baseline_{sr} = \frac{1}{|W_f|} \sum_{w \in W_f} \frac{1}{|senses(w)|}$$

We evaluate the algorithms' disambiguation performance by measuring the ratio of tokens for which our models choose the right sense:

$$Acc_{wsd} = \frac{\sum\limits_{w \in W} |\{t \in T(w) | f_m(w) = senses_s(t)\}|}{\sum\limits_{w \in W} |T(w)|}$$

In the predominant sense detection task, in case of ties in SemCor, any one of the predominant senses was considered correct. Also, all algorithms were designed to randomly choose from among the top scoring options in case of a tie in the calculated scores. This introduces a small amount of randomness (less than 0.5%) in the accuracy calculation, and was done to avoid the pitfall of defaulting to the first sense listed in WordNet, which is usually the actual predominant sense (the order of senses in WordNet is based primarily on the SemCor sense distribution).

3.2 Parameter Settings

We did not specifically tune the parameters of our WSD algorithms on the SemCor corpus, as our goal was to use hand labeled data solely for testing purposes. We selected parameters that have been considered "optimal" in the literature, although admittedly some performance gains could be expected had parameter optimization taken place.

For Overlap, we used the semantic relations proposed by Banerjee and Pedersen (2003), namely hypernyms, hyponyms, meronyms, holonyms, and troponym synsets. We also adopted their overlap scoring mechanism which treats each gloss as a bag of words and assigns an n word overlap the score of n^2. Function words were not considered in the overlap computation. For LexChains, we used the relations reported in Galley and McKeown (2003). These are all first-order WordNet relations, with the addition of the *siblings* – two words are considered siblings if they are both hyponyms of the same hypernym. The relations have different weights, depending on their type and the distance between the words in the text. These weights were imported from Galley and McKeown into our implementation without modification.

Because the SemCor corpus is relatively small (less than 700,00 words), it is not ideal for constructing a neighbor thesaurus appropriate for McCarthy et al.'s (2004) method. The latter requires each word to participate in a large number of cooccurring contexts in order to obtain reliable distributional information. To overcome this problem, we followed McCarthy et al. and extracted the neighbor thesaurus from the entire BNC. We also recreated their semantic space, using a RASP-parsed (Briscoe and Carroll, 2002) version of the BNC and their set of dependencies (i.e., Verb-Object, Verb-Subject, Noun-Noun and Adjective-Noun relations). Similarly to McCarthy et al., we used Lin's (1998) measure of distributional similarity, and considered only the 50 highest ranked

Method	Acc_{ps}	$\text{Acc}_{wsd/dir}$	$\text{Acc}_{wsd/ps}$
Baseline	34.5	–	23.0
LexChains	$48.3^{*\dagger\$}$	–	$40.7^{*\#\dagger\$}$
Overlap	$49.4^{*\dagger\$}$	$36.5^{\$}$	$42.5^{*\dagger\$}$
Similarity	54.9^{*}	–	$46.5^{*\$}$
SSI	53.7^{*}	42.7	47.9^{*}
UpperBnd	100	–	68.4

Table 2: Results of individual disambiguation algorithms on SemCor nouns[2] (*: sig. diff. from Baseline, †: sig. diff. from Similarity, $^\$$: sig diff. from SSI, $^\#$: sig. diff. from Overlap, $p < 0.01$)

neighbors for a given target word. Sense similarity was computed using the Lesk's (Banerjee and Pedersen, 2003) similarity measure[1].

3.3 Results

The performance of the individual algorithms is shown in Table 2. We also include the baseline discussed in Section 3 and the upper bound of defaulting to the first (i.e., most frequent) sense provided by the manually annotated SemCor. We report predominant sense accuracy (Acc_{ps}), and WSD accuracy when using the automatically acquired predominant sense ($\text{Acc}_{wsd/ps}$). For token-based algorithms, we also report their WSD performance in context, i.e., without use of the predominant sense ($\text{Acc}_{wsd/dir}$).

As expected, the accuracy scores in the WSD task are lower than the respective scores in the predominant sense task, since detecting the predominant sense correctly only insures the correct tagging of the instances of the word with that first sense. All methods perform significantly better than the baseline in the predominant sense detection task (using a χ^2-test, as indicated in Table 2). LexChains and Overlap perform significantly worse than Similarity and SSI, whereas LexChains is not significantly different from Overlap. Likewise, the difference in performance between SSI and Similarity is not significant. With respect to WSD, all the differences in performance are statistically significant.

	Overlap	LexChains	Similarity
LexChains	28.05		
Similarity	35.87	33.10	
SSI	30.48	31.67	37.14

Table 3: Algorithms' pairwise agreement in detecting the predominant sense (as % of all words)

Interestingly, using the predominant sense detected by the Gloss Overlap and the SSI algorithm to tag all instances is preferable to tagging each instance individually (compare $\text{Acc}_{wsd/dir}$ and $\text{Acc}_{wsd/ps}$ for Overlap and SSI in Table 2). This means that a large part of the instances which were not tagged individually with the predominant sense were actually that sense.

A close examination of the performance of the individual methods in the predominant-sense detection task shows that while the accuracy of all the methods is within a range of 7%, the actual words for which each algorithm gives the correct predominant sense are very different. Table 3 shows the degree of overlap in assigning the appropriate predominant sense among the four methods. As can be seen, the largest amount of overlap is between Similarity and SSI, and this corresponds approximately to $\frac{2}{3}$ of the words they correctly label. This means that each of these two methods gets more than 350 words right which the other labels incorrectly.

If we had an "oracle" which would tell us which method to choose for each word, we would achieve approximately 82.4% in the predominant sense task, giving us 58% in the WSD task. We see that there is a large amount of complementation between the algorithms, where the successes of one make up for the failures of the others. This suggests that the errors of the individual methods are sufficiently uncorrelated, and that some advantage can be gained by combining their predictions.

4 Combination Methods

An important finding in machine learning is that a set of classifiers whose individual decisions are combined in some way (an *ensemble*) can be more accurate than any of its component classifiers, provided that the individual components are relatively accurate and diverse (Dietterich, 1997). This simple idea has been applied to a variety of classification problems ranging from optical character recognition to medical diagnosis, part-of-speech tagging (see Dietterich 1997 and van Halteren et al. 2001 for overviews), and notably supervised

[1] This measure is identical to the Extended gloss Overlap from Section 2, but instead of searching for overlap between an extended gloss and a word's context, the comparison is done between two extended glosses of two synsets.

[2] The LexChains results presented here are not directly comparable to those reported by Galley and McKeown (2003), since they tested on a subset of SemCor, and included monosemous nouns. They also used the first sense in SemCor in case of ties. The results for the Similarity method are slightly better than those reported by McCarthy et al. (2004) due to minor improvements in implementation.

WSD (Florian et al., 2002).

Since our effort is focused exclusively on unsupervised methods, we cannot use most machine learning approaches for creating an ensemble (e.g., stacking, confidence-based combination), as they require a labeled training set. We therefore examined several basic ensemble combination approaches that do not require parameter estimation from training data.

We define $Score(M_i, s_j)$ as the (normalized) score which a method M_i gives to word sense s_j. The predominant sense calculated by method M_i for word w is then determined by:

$$PS(M_i, w) = \underset{s_j \in senses(w)}{\text{argmax}} \; Score(M_i, s_j)$$

All ensemble methods receive a set $\{M_i\}_{i=1}^k$ of individual methods to combine, so we denote each ensemble method by $MethodName(\{M_i\}_{i=1}^k)$.

Direct Voting Each ensemble component has one vote for the predominant sense, and the sense with the most votes is chosen. The scoring function for the voting ensemble is defined as:

$$Score(Voting(\{M_i\}_{i=1}^k), s)) = \sum_{i=1}^{k} eq[s, PS(M_i, w)]$$

where $eq[s, PS(M_i, w)] = \begin{cases} 1 & \text{if } s = PS(M_i, w) \\ 0 & \text{otherwise} \end{cases}$

Probability Mixture Each method provides a probability distribution over the senses. These probabilities (normalized scores) are summed, and the sense with the highest score is chosen:

$$Score(ProbMix(\{M_i\}_{i=1}^k), s)) = \sum_{i=1}^{k} Score(M_i, s)$$

Rank-Based Combination Each method provides a ranking of the senses for a given target word. For each sense, its placements according to each of the methods are summed and the sense with the lowest total placement (closest to first place) wins.

$$Score(Ranking(\{M_i\}_{i=1}^k), s)) = \sum_{i=1}^{k} (-1) \cdot Place_i(s)$$

where $Place_i(s)$ is the number of distinct scores that are larger or equal to $Score(M_i, s)$.

Arbiter-based Combination One WSD method can act as an arbiter for adjudicating disagreements among component systems. It makes sense for the adjudicator to have reasonable performance on its own. We therefore selected SSI as the arbiter since it had the best accuracy on the WSD task (see Table 2). For each disagreed word w, and for each sense s of w assigned by any of the systems in the ensemble $\{M_i\}_{i=1}^k$, we calculate the following score:

$$Score(Arbiter(\{M_i\}_{i=1}^k), s) = SSIScore^*(s)$$

where $SSIScore^*(s)$ is a modified version of the score introduced in Section 2 which exploits as a context for s the set of agreed senses and the remaining words of each sentence. We exclude from the context used by SSI the senses of w which were not chosen by any of the systems in the ensemble . This effectively reduces the number of senses considered by the arbiter and can positively influence the algorithm's performance, since it eliminates noise coming from senses which are likely to be wrong.

Method	Acc_{ps}	$\text{Acc}_{wsd/ps}$
Similarity	54.9	46.5
SSI	53.5	47.9
Voting	$57.3^{\dagger\$}$	$49.8^{\dagger\$}$
PrMixture	$57.2^{\dagger\$}$	$50.4^{\dagger\$\ddagger}$
Rank-based	$58.1^{\dagger\$}$	$50.3^{\dagger\$\ddagger}$
Arbiter-based	$56.3^{\dagger\$}$	$48.7^{\dagger\$\ddagger}$
UpperBnd	100	68.4

Table 4: Ensemble Combination Results (†: sig. diff. from Similarity, \$: sig. diff. from SSI, ‡: sig. diff. from Voting, $p < 0.01$)

5 Experiment 2: Ensembles for Unsupervised WSD

5.1 Method and Parameter Settings

We assess the performance of the different ensemble systems on the same set of SemCor nouns on which the individual methods were tested. For the best ensemble, we also report results on disambiguating all nouns in the Senseval-3 data set. We focus exclusively on nouns to allow comparisons with the results obtained from SemCor. We used the same parameters as in Experiment 1 for constructing the ensembles. As discussed earlier, token-based methods can disambiguate target words either in context or using the predominant sense. SSI was employed in the predominant sense setting in our arbiter experiment.

5.2 Results

Our results are summarized in Table 4. As can be seen, all ensemble methods perform significantly

Ensemble	Acc_{ps}	$\text{Acc}_{wsd/ps}$
Rank-based	58.1	50.3
Overlap	57.6 (−0.5)	49.7 (−0.6)
LexChains	57.2 (−0.7)	50.2 (−0.1)
Similarity	56.3 (−1.8)	49.4 (−0.9)
SSI	56.3 (−1.8)	48.2 (−2.1)

Table 5: Decrease in accuracy as a result of removal of each method from the rank-based ensemble.

better than the best individual methods, i.e., Similarity and SSI. On the WSD task, the voting, probability mixture, and rank-based ensembles significantly outperform the arbiter-based one. The performances of the probability mixture, and rank-based combinations do not differ significantly but both ensembles are significantly better than voting. One of the factors contributing to the arbiter's worse performance (compared to the other ensembles) is the fact that in many cases (almost 30%), none of the senses suggested by the disagreeing methods is correct. In these cases, there is no way for the arbiter to select the correct sense. We also examined the relative contribution of each component to overall performance. Table 5 displays the drop in performance by eliminating any particular component from the rank-based ensemble (indicated by −). The system that contributes the most to the ensemble is SSI. Interestingly, Overlap and Similarity yield similar improvements in WSD accuracy (0.6 and 0.9, respectively) when added to the ensemble.

Figure 1 shows the WSD accuracy of the best single methods and the ensembles as a function of the noun frequency in SemCor. We can see that there is at least one ensemble outperforming any single method in every frequency band and that the rank-based ensemble consistently outperforms Similarity and SSI in all bands. Although Similarity has an advantage over SSI for low and medium frequency words, it delivers worse performance for high frequency words. This is possibly due to the quality of neighbors obtained for very frequent words, which are not semantically distinct enough to reliably discriminate between different senses.

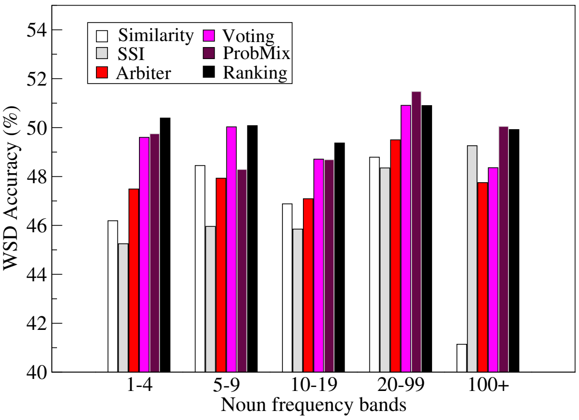


Figure 1: WSD accuracy as a function of noun frequency in SemCor

Method	Precision	Recall	Fscore
Baseline	36.8	36.8	36.8
SSI	62.5	62.5	62.5
IRST-DDD	63.3	62.2	61.2
Rank-based	63.9	63.9	63.9
UpperBnd	68.7	68.7	68.7

Table 6: Results of individual disambiguation algorithms and rank-based ensemble on Senseval-3 nouns

Table 6 lists the performance of the rank-based ensemble on the Senseval-3 (noun) corpus. We also report results for the best individual method, namely SSI, and compare our results with the best unsupervised system that participated in Senseval-3. The latter was developed by Strapparava et al. (2004) and performs domain driven disambiguation (IRST-DDD). Specifically, the approach compares the domain of the context surrounding the target word with the domains of its senses and uses a version of WordNet augmented with domain labels (e.g., economy, geography). Our baseline selects the first sense randomly and uses it to disambiguate all instances of a target word. Our upper bound defaults to the first sense from SemCor. We report precision, recall and Fscore. In cases where precision and recall figures coincide, the algorithm has 100% coverage.

As can be seen the rank-based, ensemble outperforms both SSI and the IRST-DDD system. This is an encouraging result, suggesting that there may be advantages in developing diverse classes of unsupervised WSD algorithms for system combination. The results in Table 6 are higher than those reported for SemCor (see Table 4). This is expected since the Senseval-3 data set contains monosemous nouns as well. Taking solely polysemous nouns into account, SSI's Fscore is 53.39% and the ranked-based ensemble's 55.0%. We further note that not all of the components in our ensemble are optimal. Predominant senses for Lesk and LexChains were estimated from the Senseval-3 data, however a larger corpus would probably yield more reliable estimates.

6 Conclusions and Discussion

In this paper we have presented an evaluation study of four well-known approaches to unsupervised WSD. Our comparison involved type- and token-based disambiguation algorithms relying on different kinds of WordNet relations and different amounts of corpus data. Our experiments revealed two important findings. First, type-based disambiguation yields results superior to a token-based approach. Using predominant senses is preferable to disambiguating instances individually, even for token-based algorithms. Second, the outputs of the different approaches examined here are sufficiently diverse to motivate combination methods for unsupervised WSD. We defined several ensembles on the predominant sense outputs of individual methods and showed that combination systems outperformed their best components both on the SemCor and Senseval-3 data sets.

The work described here could be usefully employed in two tasks: (a) to create preliminary annotations, thus supporting the "annotate automatically, correct manually" methodology used to provide high volume annotation in the Penn Treebank project; and (b) in combination with supervised WSD methods that take context into account; for instance, such methods could default to an unsupervised system for unseen words or words with uninformative contexts.

In the future we plan to integrate more components into our ensembles. These include not only domain driven disambiguation algorithms (Strapparava et al., 2004) but also graph theoretic ones (Mihalcea, 2005) as well as algorithms that quantify the degree of association between senses and their co-occurring contexts (Mohammad and Hirst, 2006). Increasing the number of components would allow us to employ more sophisticated combination methods such as unsupervised rank aggregation algorithms (Tan and Jin, 2004).

Acknowledgements

We are grateful to Diana McCarthy for her help with this work and to Michel Galley for making his code available to us. Thanks to John Carroll and Rob Koeling for insightful comments and suggestions. The authors acknowledge the support of EPSRC (Brody and Lapata; grant EP/C538447/1) and the European Union (Navigli; Interop NoE (508011)).

References

Banerjee, Satanjeev and Ted Pedersen. 2003. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of the 18th IJCAI*. Acapulco, pages 805–810.

Briscoe, Ted and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd LREC*. Las Palmas, Gran Canaria, pages 1499–1504.

Dietterich, T. G. 1997. Machine learning research: Four current directions. *AI Magazine* 18(4):97–136.

Edmonds, Philip. 2000. Designing a task for SENSEVAL-2. Technical note.

Florian, Radu, Silviu Cucerzan, Charles Schafer, and David Yarowsky. 2002. Combining classifiers for word sense disambiguation. *Natural Language Engineering* 1(1):1–14.

Galley, Michel and Kathleen McKeown. 2003. Improving word sense disambiguation in lexical chaining. In *Proceedings of the 18th IJCAI*. Acapulco, pages 1486–1488.

Hoste, Véronique, Iris Hendrickx, Walter Daelemans, and Antal van den Bosch. 2002. Parameter optimization for machine-learning of word sense disambiguation. *Language Engineering* 8(4):311–325.

Lesk, Michael. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th SIGDOC*. New York, NY, pages 24–26.

Lin, Dekang. 1998. An information-theoretic definition of similarity. In *Proceedings of the 15th ICML*. Madison, WI, pages 296–304.

McCarthy, Diana, Rob Koeling, Julie Weeds, and John Carroll. 2004. Finding predominant senses in untagged text. In *Proceedings of the 42th ACL*. Barcelona, Spain, pages 280–287.

Mihalcea, Rada. 2005. Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In *Proceedings of the HLT/EMNLP*. Vancouver, BC, pages 411–418.

Mihalcea, Rada and Phil Edmonds, editors. 2004. *Proceedings of the SENSEVAL-3*. Barcelona, Spain.

Miller, George A., Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proceedings of the ARPA HLT Workshop*. Morgan Kaufman, pages 303–308.

Mohammad, Saif and Graeme Hirst. 2006. Determining word sense dominance using a thesaurus. In *Proceedings of the EACL*. Trento, Italy, pages 121–128.

Morris, Jane and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics* 1(17):21–43.

Navigli, Roberto. 2005. Semi-automatic extension of large-scale linguistic knowledge bases. In *Proceedings of the 18th FLAIRS*. Florida.

Navigli, Roberto and Paola Velardi. 2005. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *PAMI* 27(7):1075–1088.

Ng, Tou Hwee. 1997. Getting serious about word sense disambiguation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*. Washington, DC, pages 1–7.

Stokoe, Christopher. 2005. Differentiating homonymy and polysemy in information retrieval. In *Proceedings of the HLT/EMNLP*. Vancouver, BC, pages 403–410.

Strapparava, Carlo, Alfio Gliozzo, and Claudio Giuliano. 2004. Word-sense disambiguation for machine translation. In *Proceedings of the SENSEVAL-3*. Barcelona, Spain, pages 229–234.

Tan, Pang-Ning and Rong Jin. 2004. Ordering patterns by combining opinions from multiple sources. In *Proceedings of the 10th KDD*. Seattle, WA, pages 22–25.

van Halteren, Hans, Jakub Zavrel, and Walter Daelemans. 2001. Improving accuracy in word class tagging through combination of machine learning systems. *Computational Linguistics* 27(2):199–230.

Vickrey, David, Luke Biewald, Marc Teyssier, and Daphne Koller. 2005. Word-sense disambiguation for machine translation. In *Proceedings of the HLT/EMNLP*. Vancouver, BC, pages 771–778.

Yarowsky, David and Radu Florian. 2002. Evaluating sense disambiguation across diverse parameter spaces. *Natural Language Engineering* 9(4):293–310.

Meaningful Clustering of Senses
Helps Boost Word Sense Disambiguation Performance

Roberto Navigli
Dipartimento di Informatica
Università di Roma “La Sapienza”
Roma, Italy
navigli@di.uniroma1.it

Abstract

Fine-grained sense distinctions are one of the major obstacles to successful Word Sense Disambiguation. In this paper, we present a method for reducing the granularity of the WordNet sense inventory based on the mapping to a manually crafted dictionary encoding sense hierarchies, namely the Oxford Dictionary of English. We assess the quality of the mapping and the induced clustering, and evaluate the performance of coarse WSD systems in the Senseval-3 English all-words task.

1 Introduction

Word Sense Disambiguation (WSD) is undoubtedly one of the hardest tasks in the field of Natural Language Processing. Even though some recent studies report benefits in the use of WSD in specific applications (e.g. Vickrey et al. (2005) and Stokoe (2005)), the present performance of the best ranking WSD systems does not provide a sufficient degree of accuracy to enable real-world, language-aware applications.

Most of the disambiguation approaches adopt the WordNet dictionary (Fellbaum, 1998) as a sense inventory, thanks to its free availability, wide coverage, and existence of a number of standard test sets based on it. Unfortunately, WordNet is a fine-grained resource, encoding sense distinctions that are often difficult to recognize even for human annotators (Edmonds and Kilgariff, 1998).

Recent estimations of the inter-annotator agreement when using the WordNet inventory report figures of 72.5% agreement in the preparation of the English all-words test set at Senseval-3 (Snyder and Palmer, 2004) and 67.3% on the Open Mind Word Expert annotation exercise (Chklovski and Mihalcea, 2002). These numbers lead us to believe that a credible upper bound for unrestricted fine-grained WSD is around 70%, a figure that state-of-the-art automatic systems find it difficult to outperform. Furthermore, even if a system were able to exceed such an upper bound, it would be unclear how to interpret such a result.

It seems therefore that the major obstacle to effective WSD is the fine granularity of the WordNet sense inventory, rather than the performance of the best disambiguation systems. Interestingly, Ng et al. (1999) show that, when a coarse-grained sense inventory is adopted, the increase in inter-annotator agreement is much higher than the reduction of the polysemy degree.

Following these observations, the main question that we tackle in this paper is: *can we produce and evaluate coarse-grained sense distinctions and show that they help boost disambiguation on standard test sets?* We believe that this is a crucial research topic in the field of WSD, that could potentially benefit several application areas.

The contribution of this paper is two-fold. First, we provide a wide-coverage method for clustering WordNet senses via a mapping to a coarse-grained sense inventory, namely the Oxford Dictionary of English (Soanes and Stevenson, 2003) (Section 2). We show that this method is well-founded and accurate with respect to manually-made clusterings (Section 3). Second, we evaluate the performance of WSD systems when using coarse-grained sense inventories (Section 4). We conclude the paper with an account of related work (Section 5), and some final remarks (Section 6).

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 105–112,
Sydney, July 2006. ©2006 Association for Computational Linguistics

2 Producing a Coarse-Grained Sense Inventory

In this section, we present an approach to the automatic construction of a coarse-grained sense inventory based on the mapping of WordNet senses to coarse senses in the Oxford Dictionary of English. In section 2.1, we introduce the two dictionaries, in Section 2.2 we illustrate the creation of sense descriptions from both resources, while in Section 2.3 we describe a lexical and a semantic method for mapping sense descriptions of WordNet senses to ODE coarse entries.

2.1 The Dictionaries

WordNet (Fellbaum, 1998) is a computational lexicon of English which encodes concepts as synonym sets (*synsets*), according to psycholinguistic principles. For each word sense, WordNet provides a gloss (i.e. a textual definition) and a set of relations such as hypernymy (e.g. apple *kind-of* edible fruit), meronymy (e.g. computer *has-part* CPU), etc.

The Oxford Dictionary of English (ODE) (Soanes and Stevenson, 2003)[1] provides a hierarchical structure of senses, distinguishing between homonymy (i.e. completely distinct senses, like race as a competition and race as a taxonomic group) and polysemy (e.g. race as a channel and as a current). Each polysemous sense is further divided into a *core sense* and a set of *subsenses*. For each sense (both core and subsenses), the ODE provides a textual definition, and possibly hypernyms and domain labels. Excluding monosemous senses, the ODE has an average number of 2.56 senses per word compared to the average polysemy of 3.21 in WordNet on the same words (with peaks for verbs of 2.73 and 3.75 senses, respectively).

In Table 1 we show an excerpt of the sense inventories of the noun *race* as provided by both dictionaries[2]. The ODE identifies 3 homonyms and 3 polysemous senses for the first homonym, while WordNet encodes a flat list of 6 senses, some of which strongly related (e.g. $race\#1$ and $race\#3$). Also, the ODE provides a sense (ginger root) which is not taken into account in WordNet.

The structure of the ODE senses is clearly hierarchical: if we were able to map with a high accuracy WordNet senses to ODE entries, then a sense clustering could be trivially induced from the mapping. As a result, the granularity of the WordNet inventory would be drastically reduced. Furthermore, disregarding errors, the clustering would be well-founded, as the ODE sense groupings were manually crafted by expert lexicographers. In the next section we illustrate a general way of constructing sense descriptions that we use for determining a complete, automatic mapping between the two dictionaries.

2.2 Constructing Sense Descriptions

For each word w, and for each sense S of w in a given dictionary $D \in \{\text{WordNet}, \text{ODE}\}$, we construct a sense description $d_{\text{D}}(S)$ as a bag of words:

$$d_{\text{D}}(S) = \mathit{def}_{\text{D}}(S) \cup \mathit{hyper}_{\text{D}}(S) \cup \mathit{domains}_{\text{D}}(S)$$

where:

- $\mathit{def}_{\text{D}}(S)$ is the set of words in the textual definition of S (excluding usage examples), automatically lemmatized and part-of-speech tagged with the RASP statistical parser (Briscoe and Carroll, 2002);
- $\mathit{hyper}_{\text{D}}(S)$ is the set of direct hypernyms of S in the taxonomy hierarchy of D ($\emptyset$ if hypernymy is not available);
- $\mathit{domains}_{\text{D}}(S)$ includes the set of domain labels possibly assigned to sense S ($\emptyset$ when no domain is assigned).

Specifically, in the case of WordNet, we generate $\mathit{def}_{\text{WN}}(S)$ from the gloss of S, $\mathit{hyper}_{\text{WN}}(S)$ from the noun and verb taxonomy, and $\mathit{domains}_{\text{WN}}(S)$ from the subject field codes, i.e. domain labels produced semi-automatically by Magnini and Cavaglià (2000) for each WordNet synset (we exclude the general-purpose label, called FACTOTUM).

For example, for the first WordNet sense of $\mathit{race}\#n$ we obtain the following description:

$$d_{\text{WN}}(\mathit{race}\#n\#1) = \{\mathit{competition}\#n\} \cup \{\mathit{contest}\#n\} \cup \{\text{POLITICS}\#\text{N}, \text{SPORT}\#\text{N}\}$$

In the case of the ODE, $\mathit{def}_{\text{ODE}}(S)$ is generated from the definitions of the core sense and the subsenses of the entry S. Hypernymy (for nouns only) and domain labels, when available, are included in the respective sets $\mathit{hyper}_{\text{ODE}}(S)$

[1] The ODE was kindly made available by Ken Litkowski (CL Research) in the context of a license agreement.

[2] In the following, we denote a WordNet sense with the convention *w#p#i* where w is a word, p a part of speech and i is a sense number; analogously, we denote an ODE sense with the convention $w\#p\#h.k$ where h is the homonym number and k is the k-th polysemous entry under homonym h.

Table 1: The sense inventory of *race#n* in WordNet and ODE (definitions are abridged, bullets (•) indicate a subsense in the ODE, arrows (→) indicate hypernymy, DOMAIN LABELS are in small caps).

race#n (WordNet)	
#1	Any competition (→ contest).
#2	People who are believed to belong to the same genetic stock (→ group).
#3	A contest of speed (→ contest).
#4	The flow of air that is driven backwards by an aircraft propeller (→ flow).
#5	A taxonomic group that is a division of a species; usually arises as a consequence of geographical isolation within a species (→ taxonomic group).
#6	A canal for a current of water (→ canal).

race#n (ODE)	
#1.1	**Core:** SPORT A competition between runners, horses, vehicles, etc. • RACING A series of such competitions for horses or dogs • A situation in which individuals or groups compete (→ contest) • ASTRONOMY The course of the sun or moon through the heavens (→ trajectory).
#1.2	**Core:** NAUTICAL A strong or rapid current (→ flow).
#1.3	**Core:** A groove, channel, or passage. • MECHANICS A water channel • Smooth groove or guide for balls (→ indentation, conduit) • FARMING Fenced passageway in a stockyard (→ route) • TEXTILES The channel along which the shuttle moves.
#2.1	**Core:** ANTHROPOLOGY Division of humankind (→ ethnic group). • The condition of belonging to a racial division or group • A group of people sharing the same culture, history, language • BIOLOGY A group of people descended from a common ancestor.
#3.1	**Core:** BOTANY, FOOD A ginger root (→ plant part).

and $domains_{\text{ODE}}(S)$. For example, the first ODE sense of $race\#n$ is described as follows:

$$d_{\text{ODE}}(race\#n\#1.1) = \{competition\#n, runner\#n, horse\#n, vehicle\#n, \ldots, heavens\#n\} \cup \{contest\#n, trajectory\#n\} \cup \{\text{SPORT\#N, RACING\#N, ASTRONOMY\#N}\}$$

Notice that, for every S, $d_{\text{D}}(S)$ is non-empty as a definition is always provided by both dictionaries. This approach to sense descriptions is general enough to be applicable to any other dictionary with similar characteristics (e.g. the Longman Dictionary of Contemporary English in place of ODE).

2.3 Mapping Word Senses

In order to produce a coarse-grained version of the WordNet inventory, we aim at defining an automatic mapping between WordNet and ODE, i.e. a function $\mu : Senses_{\text{WN}} \to Senses_{\text{ODE}} \cup \{\epsilon\}$, where $Senses_{\text{D}}$ is the set of senses in the dictionary D and ϵ is a special element assigned when no plausible option is available for mapping (e.g. when the ODE encodes no entry corresponding to a WordNet sense).

Given a WordNet sense $S \in Senses_{\text{WN}}(w)$ we define $\hat{m}(S)$, the best matching sense in the ODE, as:

$$\hat{m}(S) = \underset{S' \in Senses_{\text{ODE}}(w)}{\arg\max}\ match(S, S')$$

where $match : Senses_{\text{WN}} \times Senses_{\text{ODE}} \to [0, 1]$ is a function that measures the degree of matching between the sense descriptions of S and S'. We define the mapping μ as:

$$\mu(S) = \begin{cases} \hat{m}(S) & \textit{if } match(S, \hat{m}(S)) \geq \theta \\ \epsilon & otherwise \end{cases}$$

where θ is a threshold below which a matching between sense descriptions is considered unreliable. Finally, we define the clustering of senses $c(w)$ of a word w as:

$$c(w) = \{\mu^{-1}(S') : S' \in Senses_{\text{ODE}}(w), \mu^{-1}(S') \neq \emptyset\} \cup \{\{S\} : S \in Senses_{\text{WN}}(w), \mu(S) = \epsilon\}$$

where $\mu^{-1}(S')$ is the group of WordNet senses mapped to the same sense S' of the ODE, while the second set includes singletons of WordNet senses for which no mapping can be provided according to the definition of μ.

For example, an ideal mapping between entries in Table 1 would be as follows:

μ(race#n#1) = race#n#1.1, μ(race#n#2) = race#n#2.1,
μ(race#n#3) = race#n#1.1, μ(race#n#5) = race#n#2.1,
μ(race#n#4) = race#n#1.2, μ(race#n#6) = race#n#1.3,

resulting in the following clustering:

$$c(race\#n) = \{\{race\#n\#1, race\#n\#3\}, \{race\#n\#2, race\#n\#5\}, \{race\#n\#4\}, \{race\#n\#6\}\}$$

In Sections 2.3.1 and 2.3.2 we describe two different choices for the $match$ function, respectively based on the use of lexical and semantic information.

2.3.1 Lexical matching

As a first approach, we adopted a purely lexical matching function based on the notion of lexical overlap (Lesk, 1986). The function counts the number of lemmas that two sense descriptions of a word have in common (we neglect parts of speech), and is normalized by the minimum of the two description lengths:

$$match_{LESK}(S, S') = \frac{|d_{\text{WN}}(S) \cap d_{\text{ODE}}(S')|}{\min\{|d_{\text{WN}}(S)|, |d_{\text{ODE}}(S')|\}}$$

where $S \in Senses_{\text{WN}}(w)$ and $S' \in Senses_{\text{ODE}}(w)$. For instance:

$$match_{LESK}(race\#n\#1, race\#n\#1.1) = \frac{3}{\min\{4,20\}} = \frac{3}{4} = 0.75$$
$$match_{LESK}(race\#n\#2, race\#n\#1.1) = \frac{1}{8} = 0.125$$

Notice that unrelated senses can get a positive score because of an overlap of the sense descriptions. In the example, $group\#n$, the hypernym of $race\#n\#2$, is also present in the definition of $race\#n\#1.1$.

2.3.2 Semantic matching

Unfortunately, the very same concept can be defined with entirely different words. To match definitions in a semantic manner we adopted a knowledge-based Word Sense Disambiguation algorithm, Structural Semantic Interconnections (SSI, Navigli and Velardi (2004)).

SSI[3] exploits an extensive lexical knowledge base, built upon the WordNet lexicon and enriched with collocation information representing semantic relatedness between sense pairs. Collocations are acquired from existing resources (like the Oxford Collocations, the Longman Language Activator, collocation web sites, etc.). Each collocation is mapped to the WordNet sense inventory in a semi-automatic manner and transformed into a *relatedness* edge (Navigli and Velardi, 2005).

Given a word context $C = \{w_1, ..., w_n\}$, SSI builds a graph $G = (V, E)$ such that $V = \bigcup_{i=1}^{n} Senses_{\text{WN}}(w_i)$ and $(S, S') \in E$ if there is at least one semantic interconnection between S and S' in the lexical knowledge base. A *semantic interconnection pattern* is a relevant sequence of edges selected according to a manually-created context-free grammar, i.e. a path connecting a pair of word senses, possibly including a number of intermediate concepts. The grammar consists of a small number of rules, inspired by the notion of lexical chains (Morris and Hirst, 1991).

SSI performs disambiguation in an iterative fashion, by maintaining a set $\mathcal{C}$ of senses as a semantic context. Initially, $\mathcal{C} = V$ (the entire set of senses of words in C). At each step, for each sense S in C, the algorithm calculates a score of the degree of connectivity between S and the other senses in $\mathcal{C}$:

$$Score_{SSI}(S, \mathcal{C}) = \frac{\sum\limits_{S' \in \mathcal{C} \setminus \{S\}} \sum\limits_{i \in IC(S,S')} \frac{1}{length(i)}}{\sum\limits_{S' \in \mathcal{C} \setminus \{S\}} |IC(S,S')|}$$

where $IC(S, S')$ is the set of interconnections between senses S and S'. The contribution of a single interconnection is given by the reciprocal of its length, calculated as the number of edges connecting its ends. The overall degree of connectivity is then normalized by the number of contributing interconnections. The highest ranking sense S of word w is chosen and the senses of w are removed from the semantic context $\mathcal{C}$. The algorithm terminates when either $\mathcal{C} = \emptyset$ or there is no sense such that its score exceeds a fixed threshold.

Given a word w, semantic matching is performed in two steps. First, for each dictionary $D \in \{\text{WordNet}, \text{ODE}\}$, and for each sense $S \in Senses_{\text{D}}(w)$, the sense description of S is disambiguated by applying SSI to $d_D(S)$. As a result, we obtain a semantic description as a bag of concepts $d_D^{sem}(S)$. Notice that sense descriptions from both dictionaries are disambiguated with respect to the WordNet sense inventory.

Second, given a WordNet sense $S \in Senses_{\text{WN}}(w)$ and an ODE sense $S' \in Senses_{\text{ODE}}(w)$, we define $match_{SSI}(S, S')$ as a function of the direct relations connecting senses in $d_{\text{WN}}^{sem}(S)$ and $d_{\text{ODE}}^{sem}(S')$:

$$match_{SSI}(S, S') = \frac{|c \to c' : c \in d_{\text{WN}}^{sem}(S), c' \in d_{\text{ODE}}^{sem}(S')|}{|d_{\text{WN}}^{sem}(S)| \cdot |d_{\text{ODE}}^{sem}(S')|}$$

where $c \to c'$ denotes the existence of a relation edge in the lexical knowledge base between a concept c in the description of S and a concept c' in the description of S'. Edges include the WordNet relation set (synonymy, hypernymy, meronymy, antonymy, similarity, nominalization, etc.) and the *relatedness* edge mentioned above (we adopt only direct relations to maintain a high precision).

For example, some of the relations found between concepts in $d_{\text{WN}}^{sem}(race\#n\#3)$ and $d_{\text{ODE}}^{sem}(race\#n\#1.1)$ are:

$race\#n\#3$	$relation$	$race\#n\#1.1$
speed#n#1	$\xrightarrow{related-to}$	vehicle#n#1
race#n#3	$\xrightarrow{related-to}$	compete#v#1
racing#n#1	$\xrightarrow{kind-of}$	sport#n#1
race#n#3	$\xrightarrow{kind-of}$	contest#n#1

contributing to the final value of the function on the two senses:

$$match_{SSI}(race\#n\#3, race\#n\#1.1) = 0.41$$

Due to the normalization factor in the denominator, these values are generally low, but unrelated

[3] Available online from: http://lcl.di.uniroma1.it/ssi

Table 2: Performance of the lexical and semantic mapping functions.

Func.	**Prec.**	**Recall**	**F1**	**Acc.**
Lesk	84.74%	65.43%	73.84%	66.08%
SSI	86.87%	79.67%	83.11%	77.94%

senses have values much closer to 0. We chose SSI for the semantic matching function as it has the best performance among untrained systems on unconstrained WSD (cf. Section 4.1).

3 Evaluating the Clustering

We evaluated the accuracy of the mapping produced with the lexical and semantic methods described in Sections 2.3.1 and 2.3.2, respectively. We produced a gold-standard data set by manually mapping 5,077 WordNet senses of 763 randomly-selected words to the respective ODE entries (distributed as follows: 466 nouns, 231 verbs, 50 adjectives, 16 adverbs). The data set was created by two annotators and included only polysemous words. These words had 2,600 senses in the ODE.

Overall, 4,599 out of the 5,077 WordNet senses had a corresponding sense in ODE (i.e. the ODE covered 90.58% of the WordNet senses in the data set), while 2,053 out of the 2,600 ODE senses had an analogous entry in WordNet (i.e. WordNet covered 78.69% of the ODE senses). The WordNet clustering induced by the manual mapping was 49.85% of the original size and the average degree of polysemy decreased from 6.65 to 3.32.

The reliability of our data set is substantiated by a quantitative assessment: 548 WordNet senses of 60 words were mapped to ODE entries by both annotators, with a pairwise mapping agreement of 92.7%. The average Cohen's κ agreement between the two annotators was 0.874.

In Table 2 we report the precision and recall of the lexical and semantic functions in providing the appropriate association for the set of senses having a corresponding entry in ODE (i.e. excluding the cases where a sense ϵ was assigned by the manual annotators, cf. Section 2.3). We also report in the Table the accuracy of the two functions when we view the problem as a classification task: an automatic association is correct if it corresponds to the manual association provided by the annotators or if both assign no answer (equivalently, if both provide an ϵ label). All the differences between Lesk and SSI are statistically significant ($p < 0.01$).

As a second experiment, we used two information-theoretic measures, namely *entropy* and *purity* (Zhao and Karypis, 2004), to compare an automatic clustering $c(w)$ (i.e. the sense groups acquired for word w) with a manual clustering $\hat{c}(w)$. The entropy quantifies the distribution of the senses of a group over manually-defined groups, while the purity measures the extent to which a group contains senses primarily from one manual group.

Given a word w, and a sense group $G \in c(w)$, the entropy of G is defined as:

$$H(G) = -\frac{1}{log\ |\hat{c}(w)|} \sum_{\hat{G} \in \hat{c}(w)} \frac{|\hat{G} \cap G|}{|\hat{G}|} log(\frac{|\hat{G} \cap G|}{|\hat{G}|})$$

i.e., the entropy[4] of the distribution of senses of group G over the groups of the manual clustering $\hat{c}(w)$. The entropy of an entire clustering $c(w)$ is defined as:

$$Entropy(c(w)) = \sum_{G \in c(w)} \frac{|G|}{|Senses_{\mathrm{WN}}(w)|} H(G)$$

that is, the entropy of each group weighted by its size. The purity of a sense group $G \in c(w)$ is defined as:

$$Pu(G) = \frac{1}{|G|} \max_{\hat{G} \in \hat{c}(w)} |\hat{G} \cap G|$$

i.e., the normalized size of the largest subset of G contained in a single group $\hat{G}$ of the manual clustering. The overall purity of a clustering is obtained as a weighted sum of the individual cluster purities:

$$Purity(c(w)) = \sum_{G \in c(w)} \frac{|G|}{|Senses_{\mathrm{WN}}(w)|} Pu(G)$$

We calculated the entropy and purity of the clustering produced automatically with the lexical and the semantic method, when compared to the grouping induced by our manual mapping (ODE), and to the grouping manually produced for the English all-words task at Senseval-2 (3,499 senses of 403 nouns). We excluded from both gold standards words having a single cluster. The figures are shown in Table 3 (good entropy and purity values should be close to 0 and 1 respectively).

Table 3 shows that the quality of the clustering induced with a semantic function outperforms both lexical overlap and a random baseline. The baseline was computed averaging among 200 random clustering solutions for each word. Random

[4] Notice that we are comparing clusterings against the manual clustering (rather than viceversa), as otherwise a completely unclustered solution would result in 1.0 entropy and 0.0 purity.

Table 3: Comparison with gold standards.

Gold standard	Method	Entropy	Purity
ODE	Lesk	0.15	0.87
	SSI	0.11	0.87
	Baseline	0.28	0.67
Senseval	Lesk	0.17	0.71
	SSI	0.16	0.69
	Baseline	0.27	0.57

clusterings were the result of a random mapping function between WordNet and ODE senses. As expected, the automatic clusterings have a lower purity when compared to the Senseval-2 noun grouping as the granularity of the latter is much finer than ODE (entropy is only partially affected by this difference, indicating that we are producing larger groups). Indeed, our gold standard (ODE), when compared to the Senseval groupings, obtains a low purity as well (0.75) and an entropy of 0.13.

4 Evaluating Coarse-Grained WSD

The main reason for building a clustering of WordNet senses is to make Word Sense Disambiguation a feasible task, thus overcoming the obstacles that even humans encounter when annotating sentences with excessively fine-grained word senses.

As the semantic method outperformed the lexical overlap in the evaluations of previous Section, we decided to acquire a clustering on the entire WordNet sense inventory using this approach. As a result, we obtained a reduction of 33.54% in the number of entries (from 60,302 to 40,079 senses) and a decrease of the polysemy degree from 3.14 to 2.09. These figures exclude monosemous senses and derivatives in WordNet. As we are experimenting on an automatically-acquired clustering, all the figures are affected by the 22.06% error rate resulting from Table 2.

4.1 Experiments on Senseval-3

As a first experiment, we assessed the effect of the automatic sense clustering on the English all-words task at Senseval-3 (Snyder and Palmer, 2004). This task required WSD systems to provide a sense choice for 2,081 content words in a set of 301 sentences from the fiction, news story, and editorial domains.

We considered the three best-ranking WSD systems – GAMBL (Decadt et al., 2004), SenseLearner (Mihalcea and Faruque, 2004), and Koc University (Yuret, 2004) – and the best unsupervised system, namely IRST-DDD (Strapparava et al., 2004). We also included SSI as it outperforms all the untrained systems (Navigli and Velardi, 2005). To evaluate the performance of the five systems on our coarse clustering, we considered a fine-grained answer to be correct if it belongs to the same cluster as that of the correct answer. Table 4 reports the performance of the systems, together with the first sense and the random baseline (in the last column we report the performance on the original fine-grained test set).

Table 4: Performance of WSD systems at Senseval-3 on coarse-grained sense inventories.

System	Prec.	Rec.	F1	$\mathbf{F1}_{fine}$
Gambl	0.779	0.779	0.779	0.652
SenseLearner	0.769	0.769	0.769	0.646
KOC Univ.	0.768	0.768	0.768	0.641
SSI	0.758	0.758	0.758	0.612
IRST-DDD	0.721	0.719	0.720	0.583
FS baseline	0.769	0.769	0.769	0.624
Random BL	0.497	0.497	0.497	0.340

The best system, Gambl, obtains almost 78% precision and recall, an interesting figure compared to 65% performance in the fine-grained WSD task. An interesting aspect is that the ranking across systems was maintained when moving from a fine-grained to a coarse-grained sense inventory, although two systems (SSI and IRST-DDD) show the best improvement.

In order to show that the general improvement is the result of an appropriate clustering, we assessed the performance of Gambl by averaging its results when using 100 randomly-generated different clusterings. We excluded monosemous clusters from the test set (i.e. words with all the senses mapped to the same ODE entry), so as to clarify the real impact of properly grouped clusters. As a result, the random setting obtained 64.56% average accuracy, while the performance when adopting our automatic clustering was 70.84% (1,025/1,447 items).

To make it clear that the performance improvement is not only due to polysemy reduction, we considered a subset of the Senseval-3 test set including only the incorrect answers given by the fine-grained version of Gambl (623 items). In other words, on this data set Gambl performs with 0% accuracy. We compared the performance of

Table 5: Performance of SSI on coarse inventories (SSI* uses a coarse-grained knowledge base).

System	**Prec.**	**Recall**	**F1**
SSI + baseline	0.758	0.758	0.758
SSI	0.717	0.576	0.639
SSI*	0.748	0.674	0.709

Gambl when adopting our automatic clustering with the accuracy of the random baseline. The results were respectively 34% and 15.32% accuracy.

These experiments prove that the performance in Table 4 is not due to chance, but to an effective way of clustering word senses. Furthermore, the systems in the Table are not taking advantage of the information given by the clustering (trained systems could be retrained on the coarse clustering). To assess this aspect, we performed a further experiment. We modified the sense inventory of the SSI lexical knowledge base by adopting the coarse inventory acquired automatically. To this end, we merged the semantic interconnections belonging to the same cluster. We also disabled the first sense baseline heuristic, that most of the systems use as a back-off when they have no information about the word at hand. We call this new setting SSI* (as opposed to SSI used in Table 4).

In Table 5 we report the results. The algorithm obtains an improvement of 9.8% recall and 3.1% precision (both statistically significant, $p < 0.05$). The increase in recall is mostly due to the fact that different senses belonging to the same cluster now contribute together to the choice of that cluster (rather than individually to the choice of a fine-grained sense).

5 Related Work

Dolan (1994) describes a method for clustering word senses with the use of information provided in the electronic version of LDOCE (textual definitions, semantic relations, domain labels, etc.). Unfortunately, the approach is not described in detail and no evaluation is provided.

Most of the approaches in the literature make use of the WordNet structure to cluster its senses. Peters et al. (1998) exploit specific patterns in the WordNet hierarchy (e.g. sisters, autohyponymy, twins, etc.) to group word senses. They study semantic regularities or generalizations obtained and analyze the effect of clustering on the compatibility of language-specific wordnets. Mihalcea and Moldovan (2001) study the structure of WordNet for the identification of sense regularities: to this end, they provide a set of semantic and probabilistic rules. An evaluation of the heuristics provided leads to a polysemy reduction of 39% and an error rate of 5.6%. A different principle for clustering WordNet senses, based on the Minimum Description Length, is described by Tomuro (2001). The clustering is evaluated against WordNet cousins and used for the study of inter-annotator disagreement. Another approach exploits the (dis)agreements of human annotators to derive coarse-grained sense clusters (Chklovski and Mihalcea, 2003), where sense similarity is computed from confusion matrices.

Agirre and Lopez (2003) analyze a set of methods to cluster WordNet senses based on the use of confusion matrices from the results of WSD systems, translation equivalences, and topic signatures (word co-occurrences extracted from the web). They assess the acquired clusterings against 20 words from the Senseval-2 sense groupings.

Finally, McCarthy (2006) proposes the use of ranked lists, based on distributionally nearest neighbours, to relate word senses. This softer notion of sense relatedness allows to adopt the most appropriate granularity for a specific application.

Compared to our approach, most of these methods do not evaluate the clustering produced with respect to a gold-standard clustering. Indeed, such an evaluation would be difficult and time-consuming without a coarse sense inventory like that of ODE. A limited assessment of coarse WSD is performed by Fellbaum et al. (2001), who obtain a large improvement in the accuracy of a maximum-entropy system on clustered verbs.

6 Conclusions

In this paper, we presented a study on the construction of a coarse sense inventory for the WordNet lexicon and its effects on unrestricted WSD.

A key feature in our approach is the use of a well-established dictionary encoding sense hierarchies. As remarked in Section 2.2, the method can employ any dictionary with a sufficiently structured inventory of senses, and can thus be applied to reduce the granularity of, e.g., wordnets of other languages. One could argue that the adoption of the ODE as a sense inventory for WSD would be a better solution. While we are not against this possibility, there are problems that cannot be solved at present: the ODE does not encode semantic relations and is not freely available. Also, most of the present research and standard data sets focus on WordNet.

The fine granularity of the WordNet sense inventory is unsuitable for most applications, thus constituting an obstacle that must be overcome. We believe that the research topic analyzed in this paper is a first step towards making WSD a feasible task and enabling language-aware applications, like information retrieval, question answering, machine translation, etc. In a future work, we plan to investigate the contribution of coarse disambiguation to such real-world applications. To this end, we aim to set up an Open Mind-like experiment for the validation of the entire mapping from WordNet to ODE, so that only a minimal error rate would affect the experiments to come.

Finally, the method presented here could be useful for lexicographers in the comparison of the quality of dictionaries, and in the detection of missing word senses.

Acknowledgments

This work is partially funded by the Interop NoE (508011), 6^{th} European Union FP. We wish to thank Paola Velardi, Mirella Lapata and Samuel Brody for their useful comments.

References

Eneko Agirre and Oier Lopez. 2003. Clustering wordnet word senses. In *Proc. of Conf. on Recent Advances on Natural Language (RANLP)*. Borovets, Bulgary.

Ted Briscoe and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proc. of* 3^{rd} *Conference on Language Resources and Evaluation*. Las Palmas, Gran Canaria.

Tim Chklovski and Rada Mihalcea. 2002. Building a sense tagged corpus with open mind word expert. In *Proc. of ACL 2002 Workshop on WSD: Recent Successes and Future Directions*. Philadelphia, PA.

Tim Chklovski and Rada Mihalcea. 2003. Exploiting agreement and disagreement of human annotators for word sense disambiguation. In *Proc. of Recent Advances In NLP (RANLP 2003)*. Borovetz, Bulgaria.

Bart Decadt, Véronique Hoste, Walter Daelemans, and Antal van den Bosch. 2004. Gambl, genetic algorithm optimization of memory-based wsd. In *Proc. of ACL/SIGLEX Senseval-3*. Barcelona, Spain.

William B. Dolan. 1994. Word sense ambiguation: Clustering related senses. In *Proc. of 15th Conference on Computational Linguistics (COLING)*. Morristown, N.J.

Philip Edmonds and Adam Kilgariff. 1998. Introduction to the special issue on evaluating word sense disambiguation systems. *Journal of Natural Language Engineering*, 8(4).

Christiane Fellbaum, Martha Palmer, Hoa Trang Dang, Lauren Delfs, and Susanne Wolf. 2001. Manual and automatic semantic annotation with wordnet. In *Proc. of NAACL Workshop on WordNet and Other Lexical Resources*. Pittsburgh, PA.

Christiane Fellbaum, editor. 1998. *WordNet: an Electronic Lexical Database*. MIT Press.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine code from an ice cream cone. In *Proc. of* 5^{th} *Conf. on Systems Documentation*. ACM Press.

Bernardo Magnini and Gabriela Cavaglià. 2000. Integrating subject field codes into wordnet. In *Proc. of the* 2^{nd} *Conference on Language Resources and Evaluation (LREC)*. Athens, Greece.

Diana McCarthy. 2006. Relating wordnet senses for word sense disambiguation. In *Proc. of ACL Workshop on Making Sense of Sense*. Trento, Italy.

Rada Mihalcea and Ehsanul Faruque. 2004. Senselearner: Minimally supervised word sense disambiguation for all words in open text. In *Proc. of ACL/SIGLEX Senseval-3*. Barcelona, Spain.

Rada Mihalcea and Dan Moldovan. 2001. Automatic generation of a coarse grained wordnet. In *Proc. of NAACL Workshop on WordNet and Other Lexical Resources*. Pittsburgh, PA.

Jane Morris and Graeme Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1).

Roberto Navigli and Paola Velardi. 2004. Learning domain ontologies from document warehouses and dedicated websites. *Computational Linguistics*, 30(2).

Roberto Navigli and Paola Velardi. 2005. Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(7).

Hwee T. Ng, Chung Y. Lim, and Shou K. Foo. 1999. A case study on the inter-annotator agreement for word sense disambiguation. In *Proc. of ACL Workshop: Standardizing Lexical Resources*. College Park, Maryland.

Wim Peters, Ivonne Peters, and Piek Vossen. 1998. Automatic sense clustering in eurowordnet. In *Proc. of the* 1^{st} *Conference on Language Resources and Evaluation (LREC)*. Granada, Spain.

Benjamin Snyder and Martha Palmer. 2004. The english all-words task. In *Proc. of ACL 2004 SENSEVAL-3 Workshop*. Barcelona, Spain.

Catherine Soanes and Angus Stevenson, editors. 2003. *Oxford Dictionary of English*. Oxford University Press.

Christopher Stokoe. 2005. Differentiating homonymy and polysemy in information retrieval. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*. Vancouver, Canada.

Carlo Strapparava, Alfio Gliozzo, and Claudio Giuliano. 2004. Pattern abstraction and term similarity for word sense disambiguation. In *Proc. of ACL/SIGLEX Senseval-3*. Barcelona, Spain.

Noriko Tomuro. 2001. Tree-cut and a lexicon based on systematic polysemy. In *Proc. of the Meeting of the NAACL*. Pittsburgh, USA.

David Vickrey, Luke Biewald, Marc Teyssier, and Daphne Koller. 2005. Word sense disambiguation vs. statistical machine translation. In *Proc. of Conference on Empirical Methods in Natural Language Processing*. Vancouver, Canada.

Deniz Yuret. 2004. Some experiments with a naive bayes wsd system. In *Proc. of ACL/SIGLEX Senseval-3*. Barcelona, Spain.

Ying Zhao and George Karypis. 2004. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3).

Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations

Patrick Pantel
Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
pantel@isi.edu

Marco Pennacchiotti
ART Group - DISP
University of Rome “Tor Vergata”
Viale del Politecnico 1
Rome, Italy
pennacchiotti@info.uniroma2.it

Abstract

In this paper, we present *Espresso*, a weakly-supervised, general-purpose, and accurate algorithm for harvesting semantic relations. The main contributions are: i) a method for exploiting generic patterns by filtering incorrect instances using the Web; and ii) a principled measure of pattern and instance reliability enabling the filtering algorithm. We present an empirical comparison of *Espresso* with various state of the art systems, on different size and genre corpora, on extracting various general and specific relations. Experimental results show that our exploitation of generic patterns substantially increases system recall with small effect on overall precision.

1 Introduction

Recent attention to knowledge-rich problems such as question answering (Pasca and Harabagiu 2001) and textual entailment (Geffet and Dagan 2005) has encouraged natural language processing researchers to develop algorithms for automatically harvesting shallow semantic resources. With seemingly endless amounts of textual data at our disposal, we have a tremendous opportunity to automatically grow semantic term banks and ontological resources.

To date, researchers have harvested, with varying success, several resources, including concept lists (Lin and Pantel 2002), topic signatures (Lin and Hovy 2000), facts (Etzioni et al. 2005), and word similarity lists (Hindle 1990). Many recent efforts have also focused on extracting semantic relations between entities, such as entailments (Szpektor et al. 2004), *is-a* (Ravichandran and Hovy 2002), *part-of* (Girju et al. 2006), and other relations.

The following desiderata outline the properties of an ideal relation harvesting algorithm:

- *Performance*: it must generate both high precision and high recall relation instances;
- *Minimal supervision*: it must require little or no human annotation;
- *Breadth*: it must be applicable to varying corpus sizes and domains; and
- *Generality*: it must be applicable to a wide variety of relations (i.e., not just *is-a* or *part-of*).

To our knowledge, no previous harvesting algorithm addresses all these properties concurrently.

In this paper, we present *Espresso*, a general-purpose, broad, and accurate corpus harvesting algorithm requiring minimal supervision. The main algorithmic contribution is a novel method for exploiting *generic patterns*, which are broad coverage noisy patterns – i.e., patterns with high recall and low precision. Insofar, difficulties in using these patterns have been a major impediment for minimally supervised algorithms resulting in either very low precision or recall. We propose a method to automatically detect generic patterns and to separate their correct and incorrect instances. The key intuition behind the algorithm is that given a set of *reliable* (high precision) patterns on a corpus, correct instances of a generic pattern will fire more with reliable patterns on a very large corpus, like the Web, than incorrect ones. Below is a summary of the main contributions of this paper:

- ***Algorithm for exploiting generic patterns***: Unlike previous algorithms that require significant manual work to make use of generic patterns, we propose an unsupervised Web-filtering method for using generic patterns; and
- ***Principled reliability measure***: We propose a new measure of pattern and instance reliability which enables the use of generic patterns.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 113–120,
Sydney, July 2006.

Espresso addresses the desiderata as follows:

- *Performance*: *Espresso* generates balanced precision and recall relation instances by exploiting generic patterns;
- *Minimal supervision*: *Espresso* requires as input only a small number of seed instances;
- *Breadth*: *Espresso* works on both small and large corpora – it uses Web and syntactic expansions to compensate for lacks of redundancy in small corpora;
- *Generality*: *Espresso* is amenable to a wide variety of binary relations, from classical *is-a* and *part-of* to specific ones such as *reaction* and *succession*.

Previous work like (Girju et al. 2006) that has made use of generic patterns through filtering has shown both high precision and high recall, at the expensive cost of much manual semantic annotation. Minimally supervised algorithms, like (Hearst 1992; Pantel et al. 2004), typically ignore generic patterns since system precision dramatically decreases from the introduced noise and bootstrapping quickly spins out of control.

2 Relevant Work

To date, most research on relation harvesting has focused on *is-a* and *part-of*. Approaches fall into two categories: pattern- and clustering-based.

Most common are *pattern-based approaches*. Hearst (1992) pioneered using patterns to extract hyponym (*is-a*) relations. Manually building three lexico-syntactic patterns, Hearst sketched a bootstrapping algorithm to learn more patterns from instances, which has served as the model for most subsequent pattern-based algorithms.

Berland and Charniak (1999) proposed a system for *part-of* relation extraction, based on the (Hearst 1992) approach. Seed instances are used to infer linguistic patterns that are used to extract new instances. While this study introduces statistical measures to evaluate instance quality, it remains vulnerable to data sparseness and has the limitation of considering only one-word terms.

Improving upon (Berland and Charniak 1999), Girju et al. (2006) employ machine learning algorithms and WordNet (Fellbaum 1998) to disambiguate *part-of* generic patterns like "*X's Y*" and "*X of Y*". This study is the first extensive attempt to make use of generic patterns. In order to discard incorrect instances, they learn WordNet-based selectional restrictions, like "*X(scene#4)'s Y(movie#1)*". While making huge grounds on improving precision/recall, heavy supervision is required through manual semantic annotations.

Ravichandran and Hovy (2002) focus on scaling relation extraction to the Web. A simple and effective algorithm is proposed to infer surface patterns from a small set of instance seeds by extracting substrings relating seeds in corpus sentences. The approach gives good results on specific relations such as *birthdates*, however it has low precision on generic ones like *is-a* and *part-of*. Pantel et al. (2004) proposed a similar, highly scalable approach, based on an edit-distance technique, to learn lexico-POS patterns, showing both good performance and efficiency. *Espresso* uses a similar approach to infer patterns, but we make use of generic patterns and apply refining techniques to deal with wide variety of relations.

Other pattern-based algorithms include (Riloff and Shepherd 1997), who used a semi-automatic method for discovering similar words using a few seed examples, KnowItAll (Etzioni et al. 2005) that performs large-scale extraction of facts from the Web, Mann (2002) who used part of speech patterns to extract a subset of *is-a* relations involving proper nouns, and (Downey et al. 2005) who formalized the problem of relation extraction in a coherent and effective combinatorial model that is shown to outperform previous probabilistic frameworks.

Clustering approaches have so far been applied only to *is-a* extraction. These methods use clustering algorithms to group words according to their meanings in text, label the clusters using its members' lexical or syntactic dependencies, and then extract an *is-a* relation between each cluster member and the cluster label. Caraballo (1999) proposed the first attempt, which used conjunction and apposition features to build noun clusters. Recently, Pantel and Ravichandran (2004) extended this approach by making use of all syntactic dependency features for each noun. The advantage of clustering approaches is that they permit algorithms to identify *is-a* relations that do not explicitly appear in text, however they generally fail to produce coherent clusters from fewer than 100 million words; hence they are unreliable for small corpora.

3 The *Espresso* Algorithm

Espresso is based on the framework adopted in (Hearst 1992). It is a minimally supervised bootstrapping algorithm that takes as input a few seed instances of a particular relation and iteratively learns surface patterns to extract more instances. The key to *Espresso* lies in its use of *generic patters*, i.e., those broad coverage noisy patterns that extract both many correct and incorrect relation instances. For example, for *part-of* relations, the pattern "*X of Y*" extracts many correct relation instances like "*wheel of the car*" but also many incorrect ones like "*house of representatives*".

The key assumption behind *Espresso* is that in very large corpora, like the Web, correct instances generated by a generic pattern will be instantiated by some *reliable patterns*, where reliable patterns are patterns that have high precision but often very low recall (e.g., "*X consists of Y*" for *part-of* relations). In this section, we describe the overall architecture of *Espresso*, propose a principled measure of reliability, and give an algorithm for exploiting generic patterns.

3.1 System Architecture

Espresso iterates between the following three phases: *pattern induction*, *pattern ranking/selection*, and *instance extraction*.

The algorithm begins with seed instances of a particular binary relation (e.g., *is-a*) and then iterates through the phases until it extracts τ_1 patterns or the average pattern score decreases by more than τ_2 from the previous iteration. In our experiments, we set $\tau_1 = 5$ and $\tau_2 = 50\%$.

For our tokenization, in order to harvest multi-word terms as relation instances, we adopt a slightly modified version of the term definition given in (Justeson 1995), as it is one of the most commonly used in the NLP literature:

((Adj|Noun)+|((Adj|Noun)(NounPrep)?)(Adj|Noun)*)Noun*

Pattern Induction

In the *pattern induction* phase, *Espresso* infers a set of surface patterns *P* that connects as many of the seed instances as possible in a given corpus. Any pattern learning algorithm would do. We chose the state of the art algorithm described in (Ravichandran and Hovy 2002) with the following slight modification. For each input instance {*x*, *y*}, we first retrieve all sentences containing the two terms *x* and *y*. The sentences are then generalized into a set of new sentences $S_{x,y}$ by replacing all terminological expressions by a terminological label, *TR*. For example:

*"Because/IN HF/NNP is/VBZ a/DT weak/JJ acid/NN and/CC **x** is/VBZ a/DT **y**"*

is generalized as:

*"Because/IN **TR** is/VBZ a/DT **TR** and/CC **x** is/VBZ a/DT **y**"*

Term generalization is useful for small corpora to ease data sparseness. Generalized patterns are naturally less precise, but this is ameliorated by our filtering step described in Section 3.3.

As in the original algorithm, all substrings linking terms *x* and *y* are then extracted from $S_{x,y}$, and overall frequencies are computed to form *P*.

Pattern Ranking/Selection

In (Ravichandran and Hovy 2002), a frequency threshold on the patterns in *P* is set to select the final patterns. However, low frequency patterns may in fact be very good. In this paper, instead of frequency, we propose a novel measure of pattern reliability, r_π, which is described in detail in Section 3.2.

Espresso ranks all patterns in *P* according to reliability r_π and discards all but the top-*k*, where *k* is set to the number of patterns from the previous iteration plus one. In general, we expect that the set of patterns is formed by those of the previous iteration plus a new one. Yet, new statistical evidence can lead the algorithm to discard a pattern that was previously discovered.

Instance Extraction

In this phase, *Espresso* retrieves from the corpus the set of instances *I* that match any of the patterns in *P*. In Section 3.2, we propose a principled measure of instance reliability, r_ι, for ranking instances. Next, *Espresso* filters incorrect instances using the algorithm proposed in Section 3.3 and then selects the highest scoring *m* instances, according to r_ι, as input for the subsequent iteration. We experimentally set *m*=200.

In small corpora, the number of extracted instances can be too low to guarantee sufficient statistical evidence for the pattern discovery phase of the next iteration. In such cases, the system enters an *expansion phase*, where instances are expanded as follows:

Web expansion: New instances of the patterns in *P* are retrieved from the Web, using the Google search engine. Specifically, for each instance $\{x, y\} \in I$, the system creates a set of queries, using each pattern in *P* instantiated with *y*. For example, given the instance *"Italy, country"* and the pattern "*Y such as X*", the resulting Google query will be *"country such as *"*. New instances are then created from the retrieved Web results (e.g. *"Canada, country"*) and added to *I*. The noise generated from this expansion is attenuated by the filtering algorithm described in Section 3.3.

Syntactic expansion: New instances are created from each instance $\{x, y\} \in I$ by extracting sub-terminological expressions from *x* corresponding to the syntactic head of terms. For example, the relation *"new record of a criminal conviction part-of FBI report"* expands to: *"new record part-of FBI report"*, and *"record part-of FBI report"*.

3.2 Pattern and Instance Reliability

Intuitively, a reliable pattern is one that is both highly precise and one that extracts many instances. The recall of a pattern *p* can be approximated by the fraction of input instances that are extracted by *p*. Since it is non-trivial to estimate automatically the precision of a pattern, we are wary of keeping patterns that generate many instances (i.e., patterns that generate high recall but potentially disastrous precision). Hence, we desire patterns that are highly associated with the input instances. Pointwise mutual information (Cover and Thomas 1991) is a commonly used metric for measuring this strength of association between two events *x* and *y*:

$$pmi(x, y) = \log \frac{P(x, y)}{P(x)P(y)}$$

We define the reliability of a pattern *p*, $r_\pi(p)$, as its average strength of association across each input instance *i* in *I*, weighted by the reliability of each instance *i*:

$$r_\pi(p) = \frac{\sum_{i \in I} \left(\frac{pmi(i, p)}{\max_{pmi}} * r_\iota(i) \right)}{|I|}$$

where $r_\iota(i)$ is the reliability of instance *i* (defined below) and max_{pmi} is the maximum pointwise mutual information between all patterns and all instances. $r_\pi(p)$ ranges from [0,1]. The reliability of the manually supplied seed instances are $r_\iota(i) = 1$. The pointwise mutual information between instance $i = \{x, y\}$ and pattern *p* is estimated using the following formula:

$$pmi(i, p) = \log \frac{|x, p, y|}{|x, *, y||*, p, *|}$$

where $|x, p, y|$ is the frequency of pattern *p* instantiated with terms *x* and *y* and where the asterisk (*) represents a wildcard. A well-known problem is that pointwise mutual information is biased towards infrequent events. We thus multiply *pmi*(*i*, *p*) with the discounting factor suggested in (Pantel and Ravichandran 2004).

Estimating the reliability of an instance is similar to estimating the reliability of a pattern. Intuitively, a reliable instance is one that is highly associated with as many reliable patterns as possible (i.e., we have more confidence in an instance when multiple reliable patterns instantiate it.) Hence, analogous to our pattern reliability measure, we define the reliability of an instance *i*, $r_\iota(i)$, as:

$$r_\iota(i) = \frac{\sum_{p \in P'} \frac{pmi(i, p)}{\max_{pmi}} * r_\pi(p)}{|P|}$$

where $r_\pi(p)$ is the reliability of pattern *p* (defined earlier) and max_{pmi} is as before. Note that $r_\iota(i)$ and $r_\pi(p)$ are recursively defined, where $r_\iota(i) = 1$ for the manually supplied seed instances.

3.3 Exploiting Generic Patterns

Generic patterns are high recall / low precision patterns (e.g, the pattern "*X of Y*" can ambiguously refer to a *part-of*, *is-a* and *possession* relations). Using them blindly increases system recall while dramatically reducing precision. Minimally supervised algorithms have typically ignored them for this reason. Only heavily supervised approaches, like (Girju et al. 2006) have successfully exploited them.

Espresso's recall can be significantly increased by automatically separating correct instances extracted by generic patterns from incorrect ones. The challenge is to harness the expressive power of the generic patterns while remaining minimally supervised.

The intuition behind our method is that in a very large corpus, like the Web, correct instances of a generic pattern will be instantiated by many of *Espresso*'s reliable patterns accepted in *P*. Recall that, by definition, *Espresso*'s reliable patterns extract instances with high precision (yet often low recall). In a very large corpus, like the Web, we assume that a correct instance will occur in at least one of *Espresso*'s reliable pattern even though the patterns' recall is low. Intuitively, our confidence in a correct instance increases when, i) the instance is associated with many reliable patterns; and ii) its association with the reliable patterns is high. At a given *Espresso* iteration, where P_R represents the set of previously selected reliable patterns, this intuition is captured by the following measure of confidence in an instance $i = \{x, y\}$:

$$S(i) = \sum_{p \in P_R} S_p(i) \times \frac{r_\pi(p)}{T}$$

where *T* is the sum of the reliability scores $r_\pi(p)$ for each pattern $p \in P_R$, and

$$S_p(i) = pmi(i, p) = \log \frac{|x, p, y|}{|x, *, y| \times |*, p, *|}$$

Table 1. Sample seeds used for each semantic relation and sample outputs from *Espresso*. The number in the parentheses for each relation denotes the total number of seeds used as input for the system.

	Is-a (12)	*Part-Of (12)*	*Succession (12)*	*Reaction (13)*	*Production (14)*
Seeds	wheat :: crop	leader :: panel	Khrushchev :: Stalin	magnesium :: oxygen	bright flame :: flares
	George Wendt :: star	city :: region	Carla Hills :: Yeutter	hydrazine :: water	hydrogen :: metal hydrides
	nitrogen :: element	ion :: matter	Bush :: Reagan	aluminum metal :: oxygen	ammonia :: nitric oxide
	diborane :: substance	oxygen :: water	Julio Barbosa :: Mendes	lithium metal :: fluorine gas	copper :: brown gas
Es-presso	Picasso :: artist	trees :: land	Ford :: Nixon	hydrogen :: oxygen	electron :: ions
	tax :: charge	material :: FBI report	Setrakian :: John Griesemer	Ni :: HCl	glycerin :: nitroglycerin
	protein :: biopolymer	oxygen :: air	Camero Cardiel :: Camacho	carbon dioxide :: methane	kidneys :: kidney stones
	HCl :: strong acid	atom :: molecule	Susan Weiss :: editor	boron :: fluorine	ions :: charge

where pointwise mutual information between instance *i* and pattern *p* is estimated with Google as follows:

$$S_p(i) \approx \frac{|x, p, y|}{|x| \times |y| \times |p|}$$

An instance *i* is rejected if *S*(*i*) is smaller than some threshold τ.

Although this filtering may also be applied to reliable patterns, we found this to be detrimental in our experiments since most instances generated by reliable patterns are correct. In *Espresso*, we classify a pattern as generic when it generates more than 10 times the instances of previously accepted reliable patterns.

4 Experimental Results

In this section, we present an empirical comparison of *Espresso* with three state of the art systems on the task of extracting various semantic relations.

4.1 Experimental Setup

We perform our experiments using the following two datasets:

- **TREC**: This dataset consists of a sample of articles from the Aquaint (TREC-9) newswire text collection. The sample consists of 5,951,432 words extracted from the following data files: AP890101 – AP890131, AP890201 – AP890228, and AP890310 – AP890319.
- **CHEM**: This small dataset of 313,590 words consists of a college level textbook of introductory chemistry (Brown et al. 2003).

Each corpus is pre-processed using the Alembic Workbench POS-tagger (Day et al. 1997).

Below we describe the systems used in our empirical evaluation of *Espresso*.

- **RH02**: The algorithm by Ravichandran and Hovy (2002) described in Section 2.
- **GI03**: The algorithm by Girju et al. (2006) described in Section 2.
- **PR04**: The algorithm by Pantel and Ravichandran (2004) described in Section 2.
- **ESP-**: The *Espresso* algorithm using the pattern and instance reliability measures, but without using generic patterns.
- **ESP+**: The full *Espresso* algorithm described in this paper exploiting generic patterns.

For *ESP+*, we experimentally set τ from Section 3.3 to τ = 0.4 for TREC and τ = 0.3 for CHEM by manually inspecting a small set of instances.

Espresso is designed to extract various semantic relations exemplified by a given small set of seed instances. We consider the standard *is-a* and *part-of* relations as well as the following more specific relations:

- *succession*: This relation indicates that a person succeeds another in a position or title. For example, *George Bush* succeeded *Bill Clinton* and *Pope Benedict XVI* succeeded *Pope John Paul II*. We evaluate this relation on the TREC-9 corpus.
- *reaction*: This relation occurs between chemical elements/molecules that can be combined in a chemical reaction. For example, *hydrogen gas* reacts-with *oxygen gas* and *zinc* reacts-with *hydrochloric acid*. We evaluate this relation on the CHEM corpus.
- *production*: This relation occurs when a process or element/object produces a result[1]. For example, *ammonia* produces *nitric oxide*. We evaluate this relation on the CHEM corpus.

For each semantic relation, we manually extracted a small set of seed examples. The seeds were used for both *Espresso* as well as RH02. Table 1 lists a sample of the seeds as well as sample outputs from *Espresso*.

4.2 Precision and Recall

We implemented the systems outlined in Section 4.1, except for GI03, and applied them to the

[1] *Production* is an ambiguous relation; it is intended to be a *causation* relation in the context of chemical reactions.

Table 2. System performance: TREC/*is-a*.

SYSTEM	INSTANCES	PRECISION*	REL RECALL†
RH02	57,525	28.0%	5.31
PR04	1,504	47.0%	0.23
ESP-	4,154	**73.0%**	1.00
ESP+	69,156	36.2%	**8.26**

Table 3. System performance: CHEM/*is-a*.

SYSTEM	INSTANCES	PRECISION*	REL RECALL†
RH02	2556	25.0%	3.76
PR04	108	40.0%	0.25
ESP-	200	**85.0%**	1.00
ESP+	1490	76.0%	**6.66**

Table 4. System performance: TREC/*part-of*.

SYSTEM	INSTANCES	PRECISION*	REL RECALL†
RH02	12,828	35.0%	42.52
ESP-	132	**80.0%**	1.00
ESP+	87,203	69.9%	**577.22**

Table 5. System performance: CHEM/*part-of*.

SYSTEM	INSTANCES	PRECISION*	REL RECALL†
RH02	11,582	33.8%	**58.78**
ESP-	111	**60.0%**	1.00
ESP+	5973	50.7%	45.47

Table 6. System performance: TREC/*succession*.

SYSTEM	INSTANCES	PRECISION*	REL RECALL†
RH02	49,798	2.0%	**36.96**
ESP-	55	**49.0%**	1.00
ESP+	55	**49.0%**	1.00

Table 7. System performance: CHEM/*reaction*.

SYSTEM	INSTANCES	PRECISION*	REL RECALL†
RH02	6,083	30%	53.67
ESP-	40	85%	1.00
ESP+	3102	**91.4%**	**89.39**

Table 8. System performance: CHEM/*production*.

SYSTEM	INSTANCES	PRECISION*	REL RECALL†
RH02	197	57.5%	0.80
ESP-	196	**72.5%**	1.00
ESP+	1676	55.8%	**6.58**

TREC and CHEM datasets. For each output set, per relation, we evaluate the precision of the system by extracting a random sample of instances (50 for the TREC corpus and 20 for the CHEM corpus) and evaluating their quality manually using two human judges (a total of 680 instances were annotated per judge). For each instance, judges may assign a score of 1 for correct, 0 for incorrect, and ½ for partially correct. Example instances that were judged partially correct include "*analyst is-a manager*" and "*pilot is-a teacher*". The kappa statistic (Siegel and Castellan Jr. 1988) on this task was K = 0.69[2]. The precision for a given set of instances is the sum of the judges' scores divided by the total instances.

Although knowing the total number of correct instances of a particular relation in any non-trivial corpus is impossible, it is possible to compute the recall of a system relative to another system's recall. Following (Pantel et al. 2004), we define the relative recall of system *A* given system *B*, $R_{A|B}$, as:

$$R_{A|B} = \frac{R_A}{R_B} = \frac{\frac{C_A}{C}}{\frac{C_B}{C}} = \frac{C_A}{C_B} = \frac{P_A \times |A|}{P_B \times |B|}$$

where R_A is the recall of *A*, C_A is the number of correct instances extracted by *A*, *C* is the (unknown) total number of correct instances in the corpus, P_A is *A*'s precision in our experiments, and |*A*| is the total number of instances discovered by *A*.

Tables 2 – 8 report the total number of instances, precision, and relative recall of each system on the TREC-9 and CHEM corpora . The relative recall is always given in relation to the *ESP-* system. For example, in Table 2, *RH02* has a relative recall of 5.31 with *ESP-*, which means that the *RH02* system outputs 5.31 times more correct relations than *ESP-* (at a cost of much lower precision). Similarly, *PR04* has a relative recall of 0.23 with *ESP-*, which means that *PR04* outputs 4.35 fewer correct relations than *ESP-* (also with a smaller precision). We did not include the results from GI03 in the tables since the system is only applicable to *part-of* relations and we did not reproduce it. However, the authors evaluated their system on a sample of the TREC-9 dataset and reported 83% precision and 72% recall (this algorithm is heavily supervised.)

[2] The kappa statistic jumps to K = 0.79 if we treat partially correct classifications as correct.

[*] Because of the small evaluation sets, we estimate the 95% confidence intervals using bootstrap resampling to be in the order of ± 10-15% (absolute numbers).

[†] Relative recall is given in relation to ESP-.

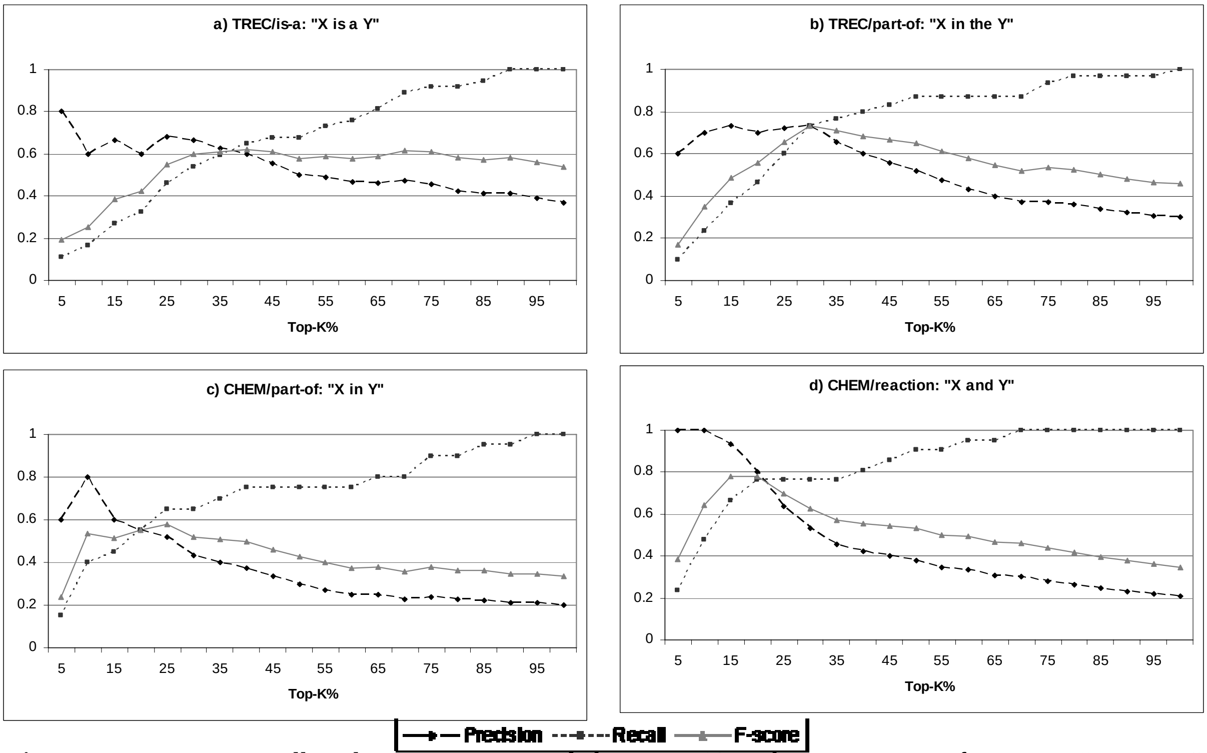


Figure 1. Precision, recall and *F*-score curves of the Top-*K*% ranking instances of patterns "*X* is a *Y*" (TREC/*is-a*), "*X* in *Y*" (TREC/*part-of*), "*X* in the *Y*" (CHEM/*part-of*), and "*X* and *Y*" (CHEM/*reaction*).

In all tables, RH02 extracts many more relations than ESP-, but with a much lower precision, because it uses generic patterns without filtering. The high precision of ESP- is due to the effective reliability measures presented in Section 3.2.

4.3 Effect of Generic Patterns

Experimental results, for all relations and the two different corpus sizes, show that *ESP-* greatly outperforms the other methods on precision. However, without the use of generic patterns, the *ESP-* system shows lower recall in all but the *production* relation.

As hypothesized, exploiting generic patterns using the algorithm from Section 3.3 substantially improves recall without much deterioration in precision. *ESP+* shows one to two orders of magnitude improvement on recall while losing on average below 10% precision. The *succession* relation in Table 6 was the only relation where *Espresso* found no generic pattern. For other relations, *Espresso* found from one to five generic patterns. Table 4 shows the power of generic patterns where system recall increases by 577 times with only a 10% drop in precision. In Table 7, we see a case where the combination of filtering with a large increase in retrieved instances resulted in both higher precision and recall.

In order to better analyze our use of generic patterns, we performed the following experiment. For each relation, we randomly sampled 100 instances for each generic pattern and built a gold standard for them (by manually tagging each instance as correct or incorrect). We then sorted the 100 instances according to the scoring formula $S(i)$ derived in Section 3.3 and computed the average precision, recall, and *F*-score of each top-*K* ranked instances for each pattern[5]. Due to lack of space, we only present the graphs for four of the 22 generic patterns: "*X* is a *Y*" for the *is-a* relation of Table 2, "*X* in the *Y*" for the *part-of* relation of Table 4, "*X* in *Y*" for the *part-of* relation of Table 5, and "*X* and *Y*" for the *reaction* relation of Table 7. Figure 1 illustrates the results.

In each figure, notice that recall climbs at a much faster rate than precision decreases. This indicates that the scoring function of Section 3.3 effectively separates correct and incorrect instances. In Figure 1a), there is a big initial drop in precision that accounts for the poor precision reported in Table 1.

Recall that the cutoff points on $S(i)$ were set to $\tau = 0.4$ for TREC and $\tau = 0.3$ for CHEM. The figures show that this cutoff is far from the maximum *F*-score. An interesting avenue of future work would be to automatically determine the proper threshold for each individual generic pattern instead of setting a uniform threshold.

[5] We can directly compute recall here since we built a gold standard for each set of 100 samples.

5 Conclusions

We proposed a weakly-supervised, general-purpose, and accurate algorithm, called *Espresso*, for harvesting binary semantic relations from raw text. The main contributions are: i) a method for exploiting generic patterns by filtering incorrect instances using the Web; and ii) a principled measure of pattern and instance reliability enabling the filtering algorithm.

We have empirically compared *Espresso*'s precision and recall with other systems on both a small domain-specific textbook and on a larger corpus of general news, and have extracted several standard and specific semantic relations: *is-a*, *part-of*, *succession*, *reaction*, and *production*. *Espresso* achieves higher and more balanced performance than other state of the art systems. By exploiting generic patterns, system recall substantially increases with little effect on precision.

There are many avenues of future work both in improving system performance and making use of the relations in applications like question answering. For the former, we plan to investigate the use of WordNet to automatically learn selectional constraints on generic patterns, as proposed by (Girju et al. 2006). We expect here that negative instances will play a key role in determining the selectional restrictions.

Espresso is the first system, to our knowledge, to emphasize concurrently *performance*, *minimal supervision*, *breadth*, and *generality*. It remains to be seen whether one could enrich existing ontologies with relations harvested by *Espresso*, and it is our hope that these relations will benefit NLP applications.

References

Berland, M. and E. Charniak, 1999. Finding parts in very large corpora. In *Proceedings of ACL-1999*. pp. 57-64. College Park, MD.

Brown, T.L.; LeMay, H.E.; Bursten, B.E.; and Burdge, J.R. 2003. *Chemistry: The Central Science*, Ninth Edition. Prentice Hall.

Caraballo, S. 1999. Automatic acquisition of a hypernym-labeled noun hierarchy from text. In *Proceedings of ACL-99*. pp 120-126, Baltimore, MD.

Cover, T.M. and Thomas, J.A. 1991. *Elements of Information Theory*. John Wiley & Sons.

Day, D.; Aberdeen, J.; Hirschman, L.; Kozierok, R.; Robinson, P.; and Vilain, M. 1997. Mixed-initiative development of language processing systems. In *Proceedings of ANLP-97*. Washington D.C.

Downey, D.; Etzioni, O.; and Soderland, S. 2005. A Probabilistic model of redundancy in information extraction. In *Proceedings of IJCAI-05*. pp. 1034-1041. Edinburgh, Scotland.

Etzioni, O.; Cafarella, M.J.; Downey, D.; Popescu, A.-M.; Shaked, T.; Soderland, S.; Weld, D.S.; and Yates, A. 2005. Unsupervised named-entity extraction from the Web: An experimental study. *Artificial Intelligence*, 165(1): 91-134.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Geffet, M. and Dagan, I. 2005. The Distributional Inclusion Hypotheses and Lexical Entailment. In *Proceedings of ACL-2005*. Ann Arbor, MI.

Girju, R.; Badulescu, A.; and Moldovan, D. 2006. Automatic Discovery of Part-Whole Relations. *Computational Linguistics*, 32(1): 83-135.

Hearst, M. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING-92*. pp. 539-545. Nantes, France.

Hindle, D. 1990. Noun classification from predicate-argument structures. In *Proceedings of ACL-90*. pp. 268–275. Pittsburgh, PA.

Justeson J.S. and Katz S.M. 1995. Technical Terminology: some linguistic properties and algorithms for identification in text. In *Proceedings of ICCL-95*. pp.539-545. Nantes, France.

Lin, C.-Y. and Hovy, E.H.. 2000. The Automated acquisition of topic signatures for text summarization. In *Proceedings of COLING-00*. pp. 495-501. Saarbrücken, Germany.

Lin, D. and Pantel, P. 2002. Concept discovery from text. In *Proceedings of COLING-02*. pp. 577-583. Taipei, Taiwan.

Mann, G. S. 2002. Fine-Grained Proper Noun Ontologies for Question Answering. In *Proceedings of SemaNet' 02: Building and Using Semantic Networks*, Taipei, Taiwan.

Pantel, P. and Ravichandran, D. 2004. Automatically labeling semantic classes. In *Proceedings of HLT/NAACL-04*. pp. 321-328. Boston, MA.

Pantel, P.; Ravichandran, D.; Hovy, E.H. 2004. Towards terascale knowledge acquisition. In *Proceedings of COLING-04*. pp. 771-777. Geneva, Switzerland.

Pasca, M. and Harabagiu, S. 2001. The informative role of WordNet in Open-Domain Question Answering. In *Proceedings of NAACL-01 Workshop on WordNet and Other Lexical Resources*. pp. 138-143. Pittsburgh, PA.

Ravichandran, D. and Hovy, E.H. 2002. Learning surface text patterns for a question answering system. In *Proceedings of ACL-2002*. pp. 41-47. Philadelphia, PA.

Riloff, E. and Shepherd, J. 1997. A corpus-based approach for building semantic lexicons. In *Proceedings of EMNLP-97*.

Siegel, S. and Castellan Jr., N. J. 1988. Nonparametric Statistics for the Behavioral Sciences. McGraw-Hill.

Szpektor, I.; Tanev, H.; Dagan, I.; and Coppola, B. 2004. Scaling web-based acquisition of entailment relations. In *Proceedings of EMNLP-04*. Barcelona, Spain.

Modeling Commonality among Related Classes in Relation Extraction

Zhou GuoDong Su Jian Zhang Min
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
Email: {zhougd, sujian, mzhang}@i2r.a-star.edu.sg

Abstract

This paper proposes a novel hierarchical learning strategy to deal with the data sparseness problem in relation extraction by modeling the commonality among related classes. For each class in the hierarchy either manually predefined or automatically clustered, a linear discriminative function is determined in a top-down way using a perceptron algorithm with the lower-level weight vector derived from the upper-level weight vector. As the upper-level class normally has much more positive training examples than the lower-level class, the corresponding linear discriminative function can be determined more reliably. The upper-level discriminative function then can effectively guide the discriminative function learning in the lower-level, which otherwise might suffer from limited training data. Evaluation on the ACE RDC 2003 corpus shows that the hierarchical strategy much improves the performance by 5.6 and 5.1 in F-measure on least- and medium- frequent relations respectively. It also shows that our system outperforms the previous best-reported system by 2.7 in F-measure on the 24 subtypes using the same feature set.

1 Introduction

With the dramatic increase in the amount of textual information available in digital archives and the WWW, there has been growing interest in techniques for automatically extracting information from text. Information Extraction (IE) is such a technology that IE systems are expected to identify relevant information (usually of predefined types) from text documents in a certain domain and put them in a structured format.

According to the scope of the ACE program (ACE 2000-2005), current research in IE has three main objectives: Entity Detection and Tracking (EDT), Relation Detection and Characterization (RDC), and Event Detection and Characterization (EDC). This paper will focus on the ACE RDC task, which detects and classifies various semantic relations between two entities. For example, we want to determine whether a person is at a location, based on the evidence in the context. Extraction of semantic relationships between entities can be very useful for applications such as question answering, e.g. to answer the query "Who is the president of the United States?".

One major challenge in relation extraction is due to the data sparseness problem (Zhou et al 2005). As the largest annotated corpus in relation extraction, the ACE RDC 2003 corpus shows that different subtypes/types of relations are much unevenly distributed and a few relation subtypes, such as the subtype "Founder" under the type "ROLE", suffers from a small amount of annotated data. Further experimentation in this paper (please see Figure 2) shows that most relation subtypes suffer from the lack of the training data and fail to achieve steady performance given the current corpus size. Given the relative large size of this corpus, it will be time-consuming and very expensive to further expand the corpus with a reasonable gain in performance. Even if we can somehow expend the corpus and achieve steady performance on major relation subtypes, it will be still far beyond practice for those minor subtypes given the much unevenly distribution among different relation subtypes. While various machine learning approaches, such as generative modeling (Miller et al 2000), maximum entropy (Kambhatla 2004) and support vector machines (Zhao and Grisman 2005; Zhou et al 2005), have been applied in the relation extraction task, no explicit learning strategy is proposed to deal with the inherent data sparseness problem caused by the much uneven distribution among different relations.

This paper proposes a novel hierarchical learning strategy to deal with the data sparseness problem by modeling the commonality among related classes. Through organizing various classes hierarchically, a linear discriminative function is determined for each class in a top-down way using a perceptron algorithm with the lower-level weight vector derived from the upper-level weight vector. Evaluation on the ACE RDC 2003 corpus shows that the hierarchical

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 121–128,
Sydney, July 2006. ©2006 Association for Computational Linguistics

strategy achieves much better performance than the flat strategy on least- and medium-frequent relations. It also shows that our system based on the hierarchical strategy outperforms the previous best-reported system.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 describes the hierarchical learning strategy using the perceptron algorithm. Finally, we present experimentation in Section 4 and conclude this paper in Section 5.

2 Related Work

The relation extraction task was formulated at MUC-7(1998). With the increasing popularity of ACE, this task is starting to attract more and more researchers within the natural language processing and machine learning communities. Typical works include Miller et al (2000), Zelenko et al (2003), Culotta and Sorensen (2004), Bunescu and Mooney (2005a), Bunescu and Mooney (2005b), Zhang et al (2005), Roth and Yih (2002), Kambhatla (2004), Zhao and Grisman (2005) and Zhou et al (2005).

Miller et al (2000) augmented syntactic full parse trees with semantic information of entities and relations, and built generative models to integrate various tasks such as POS tagging, named entity recognition, template element extraction and relation extraction. The problem is that such integration may impose big challenges, e.g. the need of a large annotated corpus. To overcome the data sparseness problem, generative models typically applied some smoothing techniques to integrate different scales of contexts in parameter estimation, e.g. the back-off approach in Miller et al (2000).

Zelenko et al (2003) proposed extracting relations by computing kernel functions between parse trees. Culotta and Sorensen (2004) extended this work to estimate kernel functions between augmented dependency trees and achieved F-measure of 45.8 on the 5 relation types in the ACE RDC 2003 corpus[1]. Bunescu and Mooney (2005a) proposed a shortest path dependency kernel. They argued that the information to model a relationship between two entities can be typically captured by the shortest path between them in the dependency graph. It achieved the F-measure of 52.5 on the 5 relation types in the ACE RDC 2003 corpus. Bunescu and Mooney (2005b) proposed a subsequence kernel and applied it in protein interaction and ACE relation extraction tasks. Zhang et al (2005) adopted clustering algorithms in unsupervised relation extraction using tree kernels. To overcome the data sparseness problem, various scales of sub-trees are applied in the tree kernel computation. Although tree kernel-based approaches are able to explore the huge implicit feature space without much feature engineering, further research work is necessary to make them effective and efficient.

Comparably, feature-based approaches achieved much success recently. Roth and Yih (2002) used the SNoW classifier to incorporate various features such as word, part-of-speech and semantic information from WordNet, and proposed a probabilistic reasoning approach to integrate named entity recognition and relation extraction. Kambhatla (2004) employed maximum entropy models with features derived from word, entity type, mention level, overlap, dependency tree, parse tree and achieved F-measure of 52.8 on the 24 relation subtypes in the ACE RDC 2003 corpus. Zhao and Grisman (2005)[2] combined various kinds of knowledge from tokenization, sentence parsing and deep dependency analysis through support vector machines and achieved F-measure of 70.1 on the 7 relation types of the ACE RDC 2004 corpus[3]. Zhou et al (2005) further systematically explored diverse lexical, syntactic and semantic features through support vector machines and achieved F-measure of 68.1 and 55.5 on the 5 relation types and the 24 relation subtypes in the ACE RDC 2003 corpus respectively. To overcome the data sparseness problem, feature-based approaches normally incorporate various scales of contexts into the feature vector extensively. These approaches then depend on adopted learning algorithms to weight and combine each feature effectively. For example, an exponential model and a linear model are applied in the maximum entropy models and support vector machines respectively to combine each feature via the learned weight vector.

In summary, although various approaches have been employed in relation extraction, they implicitly attack the data sparseness problem by using features of different contexts in feature-based approaches or including different sub-

[1] The ACE RDC 2003 corpus defines 5/24 relation types/subtypes between 4 entity types.

[2] Here, we classify this paper into feature-based approaches since the feature space in the kernels of Zhao and Grisman (2005) can be easily represented by an explicit feature vector.

[3] The ACE RDC 2004 corpus defines 7/27 relation types/subtypes between 7 entity types.

structures in kernel-based approaches. Until now, there are no explicit ways to capture the hierarchical topology in relation extraction. Currently, all the current approaches apply the flat learning strategy which equally treats training examples in different relations independently and ignore the commonality among different relations. This paper proposes a novel hierarchical learning strategy to resolve this problem by considering the relatedness among different relations and capturing the commonality among related relations. By doing so, the data sparseness problem can be well dealt with and much better performance can be achieved, especially for those relations with small amounts of annotated examples.

3 Hierarchical Learning Strategy

Traditional classifier learning approaches apply the flat learning strategy. That is, they equally treat training examples in different classes independently and ignore the commonality among related classes. The flat strategy will not cause any problem when there are a large amount of training examples for each class, since, in this case, a classifier learning approach can always learn a nearly optimal discriminative function for each class against the remaining classes. However, such flat strategy may cause big problems when there is only a small amount of training examples for some of the classes. In this case, a classifier learning approach may fail to learn a reliable (or nearly optimal) discriminative function for a class with a small amount of training examples, and, as a result, may significantly affect the performance of the class or even the overall performance.

To overcome the inherent problems in the flat strategy, this paper proposes a hierarchical learning strategy which explores the inherent commonality among related classes through a class hierarchy. In this way, the training examples of related classes can help in learning a reliable discriminative function for a class with only a small amount of training examples. To reduce computation time and memory requirements, we will only consider linear classifiers and apply the simple and widely-used perceptron algorithm for this purpose with more options open for future research. In the following, we will first introduce the perceptron algorithm in linear classifier learning, followed by the hierarchical learning strategy using the perceptron algorithm. Finally, we will consider several ways in building the class hierarchy.

3.1 Perceptron Algorithm

```
Input: the initial weight vector $w$, the training example sequence
    $(x_t, y_t) \in X \times Y, t = 1,2...,T$ and the number of
    the maximal iterations N (e.g. 10 in this paper) of the training sequence[4]
Output: the weight vector $w$ for the linear discriminative function $f = w \cdot x$
BEGIN
  $w_1 = w$
  REPEAT for t=1,2,…,T*N
    1. Receive the instance $x_t \in R^n$
    2. Compute the output $o_t = w_t \cdot x_t$
    3. Give the prediction $\hat{y}_t = sign(o_t)$
    4. Receive the desired label $y_t \in \{-1,+1\}$
    5. Update the hypothesis according to
       $w_{t+1} = w_t + \delta_t y_t x_t$    (1)
       where $\delta_t = 0$ if the margin of $w_t$ at the
         given example $(x_t, y_t)$  $y_t w_t \cdot x_t > 0$
         and $\delta_t = 1$ otherwise
  END REPEAT
  Return $w = \sum_{i=N-4}^{N} w_{T*i+1} / 5$
END BEGIN
```

Figure 1: the perceptron algorithm

This section first deals with binary classification using linear classifiers. Assume an instance space $X = R^n$ and a binary label space $Y = \{-1,+1\}$. With any weight vector $w \in R^n$ and a given instance $x \in R^n$, we associate a linear classifier h_w with a linear discriminative function [5] $f(x) = w \cdot x$ by $h_w(x) = sign(w \cdot x)$, where $sign(w \cdot x) = -1$ if $w \cdot x < 0$ and $sign(w \cdot x) = +1$ otherwise. Here, the margin of w at (x_t, y_t) is defined as $y_t w \cdot x_t$. Then if the margin is positive, we have a correct prediction with $h_w(x) = y_t$, and if the margin is negative, we have an error with $h_w(x) \neq y_t$. Therefore, given a sequence of training examples $(x_t, y_t) \in X \times Y, t = 1,2...,T$, linear classifier learning attemps to find a weight vector w that achieves a positive margin on as many examples as possible.

[4] The training example sequence is feed N times for better performance. Moreover, this number can control the maximal affect a training example can pose. This is similar to the regulation parameter C in SVM, which affects the trade-off between complexity and proportion of non-separable examples. As a result, it can be used to control over-fitting and robustness.

[5] $(w \cdot x)$ denotes the dot product of the weight vector $w \in R^n$ and a given instance $x \in R^n$.

The well-known perceptron algorithm, as shown in Figure 1, belongs to online learning of linear classifiers, where the learning algorithm represents its t -th hyposthesis by a weight vector $w_t \in R^n$. At trial t, an online algorithm receives an instance $x_t \in R^n$, makes its prediction $\hat{y}_t = sign(w_t \cdot x_t)$ and receives the desired label $y_t \in \{-1,+1\}$. What distinguishes different online algorithms is how they update w_t into w_{t+1} based on the example (x_t, y_t) received at trial t . In particular, the perceptron algorithm updates the hypothesis by adding a scalar multiple of the instance, as shown in Equation 1 of Figure 1, when there is an error. Normally, the tradictional perceptron algorithm initializes the hypothesis as the zero vector $w_1 = 0$. This is usually the most natural choice, lacking any other preference.

Smoothing

In order to further improve the performance, we iteratively feed the training examples for a possible better discriminative function. In this paper, we have set the maximal iteration number to 10 for both efficiency and stable performance and the final weight vector in the discriminative function is averaged over those of the discriminative functions in the last few iterations (e.g. 5 in this paper).

Bagging

One more problem with any online classifier learning algorithm, including the perceptron algorithm, is that the learned discriminative function somewhat depends on the feeding order of the training examples. In order to eliminate such dependence and further improve the performance, an ensemble technique, called bagging (Breiman 1996), is applied in this paper. In bagging, the bootstrap technique is first used to build M (e.g. 10 in this paper) replicate sample sets by randomly re-sampling with replacement from the given training set repeatedly. Then, each training sample set is used to train a certain discriminative function. Finally, the final weight vector in the discriminative function is averaged over those of the M discriminative functions in the ensemble.

Multi-Class Classification

Basically, the perceptron algorithm is only for binary classification. Therefore, we must extend the perceptron algorithms to multi-class classification, such as the ACE RDC task. For efficiency, we apply the *one vs. others* strategy, which builds K classifiers so as to separate one class from all others. However, the outputs for the perceptron algorithms of different classes may be not directly comparable since any positive scalar multiple of the weight vector will not affect the actual prediction of a perceptron algorithm. For comparability, we map the perceptron algorithm output into the probability by using an additional sigmoid model:

$$p(y=1 \mid f) = \frac{1}{1+\exp(Af+B)} \qquad (2)$$

where $f = w \cdot x$ is the output of a perceptron algorithm and the coefficients A & B are to be trained using the model trust alorithm as described in Platt (1999). The final decision of an instance in multi-class classification is determined by the class which has the maximal probability from the corresponding perceptron algorithm.

3.2 Hierarchical Learning Strategy using the Perceptron Algorithm

Assume we have a class hierarchy for a task, e.g. the one in the ACE RDC 2003 corpus as shown in Table 1 of Section 4.1. The hierarchical learning strategy explores the inherent commonality among related classes in a top-down way. For each class in the hierarchy, a linear discriminative function is determined in a top-down way with the lower-level weight vector derived from the upper-level weight vector iteratively. This is done by initializing the weight vector in training the linear discriminative function for the lower-level class as that of the upper-level class. That is, the lower-level discriminative function has the preference toward the discriminative function of its upper-level class. For an example, let’s look at the training of the “Located” relation subtype in the class hierarchy as shown in Table 1:

1) Train the weight vector of the linear discriminative function for the “YES” relation vs. the “NON” relation with the weight vector initialized as the zero vector.
2) Train the weight vector of the linear discriminative function for the “AT” relation type vs. all the remaining relation types (including the “NON” relation) with the weight vector initialized as the weight vector of the linear discriminative function for the “YES” relation vs. the “NON” relation.
3) Train the weight vector of the linear discriminative function for the “Located” relation subtype vs. all the remaining relation subtypes under all the relation types (including the “NON” relation) with the

weight vector initialized as the weight vector of the linear discriminative function for the "AT" relation type vs. all the remaining relation types.

4) Return the above trained weight vector as the discriminatie function for the "Located" relation subtype.

In this way, the training examples in different classes are not treated independently any more, and the commonality among related classes can be captured via the hierarchical learning strategy. The intuition behind this strategy is that the upper-level class normally has more positive training examples than the lower-level class so that the corresponding linear discriminative function can be determined more reliably. In this way, the training examples of related classes can help in learning a reliable discriminative function for a class with only a small amount of training examples in a top-down way and thus alleviate its data sparseness problem.

3.3 Building the Class Hierarchy

We have just described the hierarchical learning strategy using a given class hierarchy. Normally, a rough class hierarchy can be given manually according to human intuition, such as the one in the ACE RDC 2003 corpus. In order to explore more commonality among sibling classes, we make use of binary hierarchical clustering for sibling classes at both lowest and all levels. This can be done by first using the flat learning strategy to learn the discriminative functions for individual classes and then iteratively combining the two most related classes using the cosine similarity function between their weight vectors in a bottom-up way. The intuition is that related classes should have similar hyper-planes to separate from other classes and thus have similar weight vectors.

- **Lowest-level hybrid**: Binary hierarchical clustering is only done at the lowest level while keeping the upper-level class hierarchy. That is, only sibling classes at the lowest level are hierarchically clustered.
- **All-level hybrid**: Binary hierarchical clustering is done at all levels in a bottom-up way. That is, sibling classes at the lowest level are hierarchically clustered first and then sibling classes at the upper-level. In this way, the binary class hierarchy can be built iteratively in a bottom-up way.

4 Experimentation

This paper uses the ACE RDC 2003 corpus provided by LDC to train and evaluate the hierarchical learning strategy. Same as Zhou et al (2005), we only model explicit relations and explicitly model the argument order of the two mentions involved.

4.1 Experimental Setting

Type	Subtype	Freq	Bin Type
AT	Based-In	347	Medium
	Located	2126	Large
	Residence	308	Medium
NEAR	Relative-Location	201	Medium
PART	Part-Of	947	Large
	Subsidiary	355	Medium
	Other	6	Small
ROLE	Affiliate-Partner	204	Medium
	Citizen-Of	328	Medium
	Client	144	Small
	Founder	26	Small
	General-Staff	1331	Large
	Management	1242	Large
	Member	1091	Large
	Owner	232	Medium
	Other	158	Small
SOCIAL	Associate	91	Small
	Grandparent	12	Small
	Other-Personal	85	Small
	Other-Professional	339	Medium
	Other-Relative	78	Small
	Parent	127	Small
	Sibling	18	Small
	Spouse	77	Small

Table 1: Statistics of relation types and subtypes in the training data of the ACE RDC 2003 corpus (Note: According to frequency, all the subtypes are divided into three bins: large/ middle/ small, with 400 as the lower threshold for the large bin and 200 as the upper threshold for the small bin).

The training data consists of 674 documents (~300k words) with 9683 relation examples while the held-out testing data consists of 97 documents (~50k words) with 1386 relation examples. All the experiments are done five times on the 24 relation subtypes in the ACE corpus, except otherwise specified, with the final performance averaged using the same re-sampling with replacement strategy as the one in the bagging technique. Table 1 lists various types and subtypes of relations for the ACE RDC 2003 corpus, along with their occurrence frequency in the training data. It shows that this corpus suffers from a small amount of annotated data for a few subtypes such as the subtype "Founder" under the type "ROLE".

For comparison, we also adopt the same feature set as Zhou et al (2005): word, entity type, mention level, overlap, base phrase chunking, dependency tree, parse tree and semantic information.

4.2 Experimental Results

Table 2 shows the performance of the hierarchical learning strategy using the existing class hierarchy in the given ACE corpus and its comparison with the flat learning strategy, using the perceptron algorithm. It shows that the pure hierarchical strategy outperforms the pure flat strategy by 1.5 (56.9 vs. 55.4) in F-measure. It also shows that further smoothing and bagging improve the performance of the hierarchical and flat strategies by 0.6 and 0.9 in F-measure respectively. As a result, the final hierarchical strategy achieves F-measure of 57.8 and outperforms the final flat strategy by 1.8 in F-measure.

Strategies	P	R	F
Flat	58.2	52.8	55.4
Flat+Smoothing	58.9	53.1	55.9
Flat+Bagging	59.0	53.1	55.9
Flat+Both	**59.1**	**53.2**	**56.0**
Hierarchical	61.9	52.6	56.9
Hierarchical+Smoothing	62.7	53.1	57.5
Hierarchical+Bagging	62.9	53.1	57.6
Hierarchical+Both	**63.0**	**53.4**	**57.8**

Table 2: Performance of the hierarchical learning strategy using the existing class hierarchy and its comparison with the flat learning strategy

Class Hierarchies	P	R	F
Existing	63.0	53.4	57.8
Entirely Automatic	63.4	53.1	57.8
Lowest-level Hybrid	63.6	53.5	58.1
All-level Hybrid	**63.6**	**53.6**	**58.2**

Table 3: Performance of the hierarchical learning strategy using different class hierarchies

Table 3 compares the performance of the hierarchical learning strategy using different class hierarchies. It shows that, the lowest-level hybrid approach, which only automatically updates the existing class hierarchy at the lowest level, improves the performance by 0.3 in F-measure while further updating the class hierarchy at upper levels in the all-level hybrid approach only has very slight effect. This is largely due to the fact that the major data sparseness problem occurs at the lowest level, i.e. the relation subtype level in the ACE corpus. As a result, the final hierarchical learning strategy using the class hierarchy built with the all-level hybrid approach achieves F-measure of 58.2 in F-measure, which outperforms the final flat strategy by 2.2 in F-measure. In order to justify the usefulness of our hierarchical learning strategy when a rough class hierarchy is not available and difficult to determine manually, we also experiment using entirely automatically built class hierarchy (using the traditional binary hierarchical clustering algorithm and the cosine similarity measurement) without considering the existing class hierarchy. Table 3 shows that using automatically built class hierarchy performs comparably with using only the existing one.

With the major goal of resolving the data sparseness problem for the classes with a small amount of training examples, Table 4 compares the best-performed hierarchical and flat learning strategies on the relation subtypes of different training data sizes. Here, we divide various relation subtypes into three bins: large/middle/small, according to their available training data sizes. For the ACE RDC 2003 corpus, we use 400 as the lower threshold for the large bin[6] and 200 as the upper threshold for the small bin[7]. As a result, the large/medium/small bin includes 5/8/11 relation subtypes, respectively. Please see Table 1 for details. Table 4 shows that the hierarchical strategy outperforms the flat strategy by 1.0/5.1/5.6 in F-measure on the large/middle/small bin respectively. This indicates that the hierarchical strategy performs much better than the flat strategy for those classes with a small or medium amount of annotated examples although the hierarchical strategy only performs slightly better by 1.0 and 2.2 in F-measure than the flat strategy on those classes with a large size of annotated corpus and on all classes as a whole respectively. This suggests that the proposed hierarchical strategy can well deal with the data sparseness problem in the ACE RDC 2003 corpus.

An interesting question is about the similarity between the linear discriminative functions learned using the hierarchical and flat learning strategies. Table 4 compares the cosine similarities between the weight vectors of the linear discriminative functions using the two strategies for different bins, weighted by the training data sizes

[6] The reason to choose this threshold is that no relation subtype in the ACE RC 2003 corpus has training examples in between 400 and 900.

[7] A few minor relation subtypes only have very few examples in the testing set. The reason to choose this threshold is to guarantee a reasonable number of testing examples in the small bin. For the ACE RC 2003 corpus, using 200 as the upper threshold will fill the small bin with about 100 testing examples while using 100 will include too few testing examples for reasonable performance evaluation.

of different relation subtypes. It shows that the linear discriminative functions learned using the two strategies are very similar (with the cosine similarity 0.98) for the relation subtypes belonging to the large bin while the linear discriminative functions learned using the two strategies are not for the relation subtypes belonging to the medium/small bin with the cosine similarity 0.92/0.81 respectively. This means that the use of the hierarchical strategy over the flat strategy only has very slight change on the linear discriminative functions for those classes with a large amount of annotated examples while its effect on those with a small amount of annotated examples is obvious. This contributes to and explains (the degree of) the performance difference between the two strategies on the different training data sizes as shown in Table 4.

Due to the difficulty of building a large annotated corpus, another interesting question is about the learning curve of the hierarchical learning strategy and its comparison with the flat learning strategy. Figure 2 shows the effect of different training data sizes for some major relation subtypes while keeping all the training examples of remaining relation subtypes. It shows that the hierarchical strategy performs much better than the flat strategy when only a small amount of training examples is available. It also shows that the hierarchical strategy can achieve stable performance much faster than the flat strategy. Finally, it shows that the ACE RDC 2003 task suffers from the lack of training examples. Among the three major relation subtypes, only the subtype "Located" achieves steady performance.

Finally, we also compare our system with the previous best-reported systems, such as Kambhatla (2004) and Zhou et al (2005). Table 5 shows that our system outperforms the previous best-reported system by 2.7 (58.2 vs. 55.5) in F-measure, largely due to the gain in recall. It indicates that, although support vector machines and maximum entropy models always perform better than the simple perceptron algorithm in most (if not all) applications, the hierarchical learning strategy using the perceptron algorithm can easily overcome the difference and outperforms the flat learning strategy using the overwhelming support vector machines and maximum entropy models in relation extraction, at least on the ACE RDC 2003 corpus.

Bin Type(cosine similarity)	Large Bin (0.98)			Middle Bin (0.92)			Small Bin (0.81)		
	P	R	F	P	R	F	P	R	F
Flat Strategy	62.3	61.9	62.1	60.8	38.7	47.3	33.0	21.7	26.2
Hierarchical Strategy	66.4	60.2	63.1	67.6	42.7	52.4	40.2	26.3	31.8

Table 4: Comparison of the hierarchical and flat learning strategies on the relation subtypes of different training data sizes. Notes: the figures in the parentheses indicate the cosine similarities between the weight vectors of the linear discriminative functions learned using the two strategies.

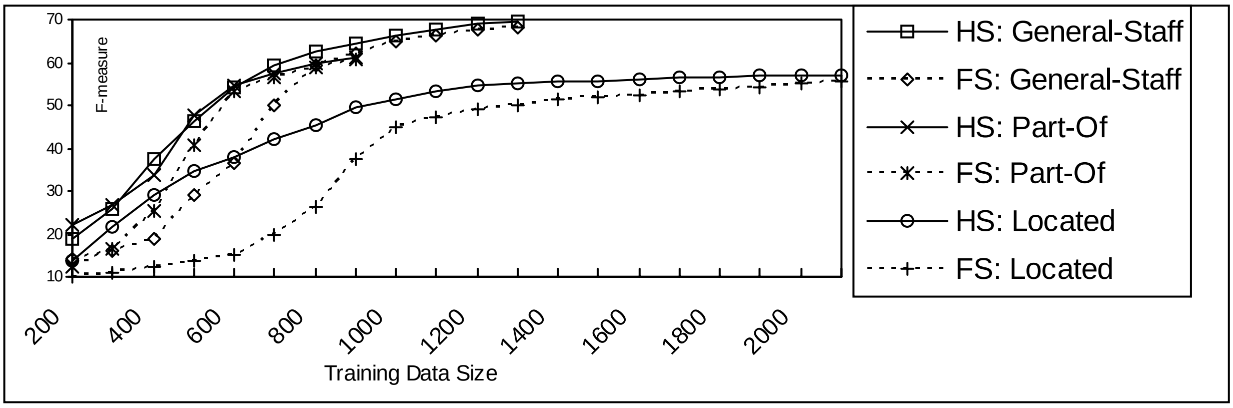


Figure 2: Learning curve of the hierarchical strategy and its comparison with the flat strategy for some major relation subtypes (Note: FS for the flat strategy and HS for the hierarchical strategy)

System	Performance		
	P	R	F
Our: Perceptron Algorithm + Hierarchical Strategy	63.6	53.6	58.2
Zhou et al (2005): SVM + Flat Strategy	63.1	49.5	55.5
Kambhatla (2004): Maximum Entropy + Flat Strategy	63.5	45.2	52.8

Table 5: Comparison of our system with other best-reported systems

5 Conclusion

This paper proposes a novel hierarchical learning strategy to deal with the data sparseness problem in relation extraction by modeling the commonality among related classes. For each class in a class hierarchy, a linear discriminative function is determined in a top-down way using the perceptron algorithm with the lower-level weight vector derived from the upper-level weight vector. In this way, the upper-level discriminative function can effectively guide the lower-level discriminative function learning. Evaluation on the ACE RDC 2003 corpus shows that the hierarchical strategy performs much better than the flat strategy in resolving the critical data sparseness problem in relation extraction.

In the future work, we will explore the hierarchical learning strategy using other machine learning approaches besides online classifier learning approaches such as the simple perceptron algorithm applied in this paper. Moreover, just as indicated in Figure 2, most relation subtypes in the ACE RDC 2003 corpus (arguably the largest annotated corpus in relation extraction) suffer from the lack of training examples. Therefore, a critical research in relation extraction is how to rely on semi-supervised learning approaches (e.g. bootstrap) to alleviate its dependency on a large amount of annotated training examples and achieve better and steadier performance. Finally, our current work is done when NER has been perfectly done. Therefore, it would be interesting to see how imperfect NER affects the performance in relation extraction. This will be done by integrating the relation extraction system with our previously developed NER system as described in Zhou and Su (2002).

References

ACE. (2000-2005). Automatic Content Extraction. http://www.ldc.upenn.edu/Projects/ACE/

Bunescu R. & Mooney R.J. (2005a). A shortest path dependency kernel for relation extraction. *HLT/EMNLP'2005*: 724-731. 6-8 Oct 2005. Vancover, B.C.

Bunescu R. & Mooney R.J. (2005b). Subsequence Kernels for Relation Extraction NIPS'2005. Vancouver, BC, December 2005

Breiman L. (1996) Bagging Predictors. *Machine Learning*, 24(2): 123-140.

Collins M. (1999). Head-driven statistical models for natural language parsing. *Ph.D. Dissertation*, University of Pennsylvania.

Culotta A. and Sorensen J. (2004). Dependency tree kernels for relation extraction. *ACL'2004*. 423-429. 21-26 July 2004. Barcelona, Spain.

Kambhatla N. (2004). Combining lexical, syntactic and semantic features with Maximum Entropy models for extracting relations. *ACL'2004(Poster)*. 178-181. 21-26 July 2004. Barcelona, Spain.

Miller G.A. (1990). WordNet: An online lexical database. *International Journal of Lexicography*. 3(4):235-312.

Miller S., Fox H., Ramshaw L. and Weischedel R. (2000). A novel use of statistical parsing to extract information from text. *ANLP'2000*. 226-233. 29 April - 4 May 2000, Seattle, USA

MUC-7. (1998). *Proceedings of the 7th Message Understanding Conference (MUC-7)*. Morgan Kaufmann, San Mateo, CA.

Platt J. 1999. Probabilistic Outputs for Support Vector Machines and Comparisions to regularized Likelihood Methods. In *Advances in Large Margin Classifiers.* Edited by Smola .J., Bartlett P., Scholkopf B. and Schuurmans D. MIT Press.

Roth D. and Yih W.T. (2002). Probabilistic reasoning for entities and relation recognition. *CoLING'2002*. 835-841.26-30 Aug 2002. Taiwan.

Zelenko D., Aone C. and Richardella. (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research*. 3(Feb):1083-1106.

Zhang M., Su J., Wang D.M., Zhou G.D. and Tan C.L. (2005). Discovering Relations from a Large Raw Corpus Using Tree Similarity-based Clustering, *IJCNLP'2005, Lecture Notes in Computer Science (LNCS 3651)*. 378-389. 11-16 Oct 2005. Jeju Island, South Korea.

Zhao S.B. and Grisman R. 2005. Extracting relations with integrated information using kernel methods. ACL'2005: 419-426. Univ of Michigan-Ann Arbor, USA, 25-30 June 2005**.**

Zhou G.D. and Su Jian. Named Entity Recognition Using a HMM-based Chunk Tagger, *ACL'2002*. pp473-480. Philadelphia. July 2002.

Zhou G.D., Su J. Zhang J. and Zhang M. (2005). Exploring various knowledge in relation extraction. *ACL'2005*. 427-434. 25-30 June, Ann Arbor, Michgan, USA.

Relation Extraction Using Label Propagation Based Semi-supervised Learning

Jinxiu Chen[1] **Donghong Ji**[1] **Chew Lim Tan**[2] **Zhengyu Niu**[1]

[1]Institute for Infocomm Research
21 Heng Mui Keng Terrace
119613 Singapore
{jinxiu,dhji,zniu}@i2r.a-star.edu.sg

[2]Department of Computer Science
National University of Singapore
117543 Singapore
tancl@comp.nus.edu.sg

Abstract

Shortage of manually labeled data is an obstacle to supervised relation extraction methods. In this paper we investigate a graph based semi-supervised learning algorithm, a label propagation (LP) algorithm, for relation extraction. It represents labeled and unlabeled examples and their distances as the nodes and the weights of edges of a graph, and tries to obtain a labeling function to satisfy two constraints: 1) it should be fixed on the labeled nodes, 2) it should be smooth on the whole graph. Experiment results on the ACE corpus showed that this LP algorithm achieves better performance than SVM when only very few labeled examples are available, and it also performs better than bootstrapping for the relation extraction task.

1 Introduction

Relation extraction is the task of detecting and classifying relationships between two entities from text. Many machine learning methods have been proposed to address this problem, e.g., supervised learning algorithms (Miller et al., 2000; Zelenko et al., 2002; Culotta and Soresen, 2004; Kambhatla, 2004; Zhou et al., 2005), semi-supervised learning algorithms (Brin, 1998; Agichtein and Gravano, 2000; Zhang, 2004), and unsupervised learning algorithms (Hasegawa et al., 2004).

Supervised methods for relation extraction perform well on the ACE Data, but they require a large amount of manually labeled relation instances. Unsupervised methods do not need the definition of relation types and manually labeled data, but they cannot detect relations between entity pairs and its result cannot be directly used in many NLP tasks since there is no relation type label attached to each instance in clustering result. Considering both the availability of a large amount of untagged corpora and direct usage of extracted relations, semi-supervised learning methods has received great attention.

DIPRE (Dual Iterative Pattern Relation Expansion) (Brin, 1998) is a bootstrapping-based system that used a pattern matching system as classifier to exploit the duality between sets of patterns and relations. Snowball (Agichtein and Gravano, 2000) is another system that used bootstrapping techniques for extracting relations from unstructured text. Snowball shares much in common with DIPRE, including the employment of the bootstrapping framework as well as the use of pattern matching to extract new candidate relations. The third system approaches relation classification problem with bootstrapping on top of SVM, proposed by Zhang (2004). This system focuses on the ACE subproblem, RDC, and extracts various lexical and syntactic features for the classification task. However, Zhang (2004)'s method doesn't actually "detect" relaitons but only performs relation classification between two entities given that they are known to be related.

Bootstrapping works by iteratively classifying unlabeled examples and adding confidently classified examples into labeled data using a model learned from augmented labeled data in previous iteration. It

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 129–136,
Sydney, July 2006. ©2006 Association for Computational Linguistics

can be found that the affinity information among unlabeled examples is not fully explored in this bootstrapping process.

Recently a promising family of semi-supervised learning algorithm is introduced, which can effectively combine unlabeled data with labeled data in learning process by exploiting manifold structure (cluster structure) in data (Belkin and Niyogi, 2002; Blum and Chawla, 2001; Blum et al., 2004; Zhu and Ghahramani, 2002; Zhu et al., 2003). These graph-based semi-supervised methods usually define a graph where the nodes represent labeled and unlabeled examples in a dataset, and edges (may be weighted) reflect the similarity of examples. Then one wants a labeling function to satisfy two constraints at the same time: 1) it should be close to the given labels on the labeled nodes, and 2) it should be smooth on the whole graph. This can be expressed in a regularization framework where the first term is a loss function, and the second term is a regularizer. These methods differ from traditional semi-supervised learning methods in that they use graph structure to smooth the labeling function.

To the best of our knowledge, no work has been done on using graph based semi-supervised learning algorithms for relation extraction. Here we investigate a label propagation algorithm (LP) (Zhu and Ghahramani, 2002) for relation extraction task. This algorithm works by representing labeled and unlabeled examples as vertices in a connected graph, then propagating the label information from any vertex to nearby vertices through weighted edges iteratively, finally inferring the labels of unlabeled examples after the propagation process converges. In this paper we focus on the ACE RDC task[1].

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 formulates relation extraction problem in the context of semi-supervised learning and describes our proposed approach. Then we provide experimental results of our proposed method and compare with a popular supervised learning algorithm (SVM) and bootstrapping algorithm in Section 4. Finally we conclude our work in section 5.

2 The Proposed Method

2.1 Problem Definition

The problem of relation extraction is to assign an appropriate relation type to an occurrence of two entity pairs in a given context. It can be represented as follows:

$$R \rightarrow (C_{pre}, e_1, C_{mid}, e_2, C_{post}) \qquad (1)$$

where e_1 and e_2 denote the entity mentions, and C_{pre},C_{mid},and C_{post} are the contexts before, between and after the entity mention pairs. In this paper, we set the mid-context window as the words between the two entity mentions and the pre- and post-context as up to two words before and after the corresponding entity mention.

Let $X = \{x_i\}_{i=1}^n$ be a set of contexts of occurrences of all the entity mention pairs, where x_i represents the contexts of the i-th occurrence, and n is the total number of occurrences. The first l examples (or contexts) are labeled as y_g ($y_g \in \{r_j\}_{j=1}^R$, r_j denotes relation type and R is the total number of relation types). The remaining $u (u = n - l)$ examples are unlabeled.

Intuitively, if two occurrences of entity mention pairs have the similarity context, they tend to hold the same relation type. Based on the assumption, we define a graph where the vertices represent the contexts of labeled and unlabeled occurrences of entity mention pairs, and the edge between any two vertices x_i and x_j is weighted so that the closer the vertices in some distance measure, the larger the weight associated with this edge. Hence, the weights are defined as follows:

$$W_{ij} = exp(-\frac{s_{ij}^2}{\alpha^2}) \qquad (2)$$

where s_{ij} is the similarity between x_i and x_j calculated by some similarity measures, e.g., cosine similarity, and α is used to scale the weights. In this paper, we set α as the average similarity between labeled examples from different classes.

2.2 A Label Propagation Algorithm

In the LP algorithm, the label information of any vertex in a graph is propagated to nearby vertices through weighted edges until a global stable stage is achieved. Larger edge weights allow labels to travel

[1] http://www.ldc.upenn.edu/Projects/ACE/, Three tasks of ACE program: Entity Detection and Tracking (EDT), Relation Detection and Characterization (RDC), and Event Detection and Characterization (EDC)

through easier. Thus the closer the examples are, the more likely they have similar labels.

We define soft label as a vector that is a probabilistic distribution over all the classes. In the label propagation process, the soft label of each initial labeled example is clamped in each iteration to replenish label sources from these labeled data. Thus the labeled data act like sources to push out labels through unlabeled data. With this push from labeled examples, the class boundaries will be pushed through edges with large weights and settle in gaps along edges with small weights. Hopefully, the values of W_{ij} across different classes would be as small as possible and the values of W_{ij} within the same class would be as large as possible. This will make label propagation to stay within the same class. This label propagation process will make the labeling function smooth on the graph.

Define an $n \times n$ probabilistic transition matrix T

$$T_{ij} = P(j \rightarrow i) = \frac{w_{ij}}{\sum_{k=1}^{n} w_{kj}} \quad (3)$$

where T_{ij} is the probability to jump from vertex x_j to vertex x_i. We define a $n \times R$ label matrix Y, where Y_{ij} representing the probabilities of vertex y_i to have the label r_j.

Then the label propagation algorithm consists the following main steps:

Step1 : Initialization

- Set the iteration index $t = 0$;
- Let Y^0 be the initial soft labels attached to each vertex, where $Y^0_{ij} = 1$ if y_i is label r_j and 0 otherwise.
- Let Y^0_L be the top l rows of Y^0 and Y^0_U be the remaining u rows. Y^0_L is consistent with the labeling in labeled data and the initialization of Y^0_U can be arbitrary.

Step 2 : Propagate the labels of any vertex to nearby vertices by $Y^{t+1} = \overline{T}Y^t$, where $\overline{T}$ is the row-normalized matrix of T, i.e. $\overline{T_{ij}} = T_{ij}/\sum_k T_{ik}$, which can maintain the class probability interpretation.

Step 3 : Clamp the labeled data, that is, replace the top l row of Y^{t+1} with Y^0_L.

Step 4 : Repeat from step 2 until Y converges.

Step 5 : Assign $x_h(l + 1 \leq h \leq n)$ with a label: $y_h = argmax_j Y_{hj}$.

The above algorithm can ensure that the labeled data Y_L never changes since it is clamped in Step 3. Actually we are interested in only Y_U. This algorithm has been shown to converge to a unique solution $\hat{Y}_U = \lim_{t\rightarrow\infty} Y^t_U = (I - \bar{T}_{uu})^{-1}\bar{T}_{ul}Y^0_L$ (Zhu and Ghahramani, 2002). Here, $\bar{T}_{uu}$ and $\bar{T}_{ul}$ are acquired by splitting matrix $\bar{T}$ after the l-th row and the l-th column into 4 sub-matrices. And I is $u \times u$ identity matrix. We can see that the initialization of Y^0_U in this solution is not important, since Y^0_U does not affect the estimation of $\hat{Y}_U$.

3 Experiments and Results

3.1 Feature Set

Following (Zhang, 2004), we used lexical and syntactic features in the contexts of entity pairs, which are extracted and computed from the parse trees derived from Charniak Parser (Charniak, 1999) and the Chunklink script [2] written by Sabine Buchholz from Tilburg University.

Words: Surface tokens of the two entities and words in the three contexts.

Entity Type: the entity type of both entity mentions, which can be PERSON, ORGANIZATION, FACILITY, LOCATION and GPE.

POS features: Part-Of-Speech tags corresponding to all tokens in the two entities and words in the three contexts.

Chunking features: This category of features are extracted from the chunklink representation, which includes:

- **Chunk tag information** of the two entities and words in the three contexts. The “0” tag means that the word is not in any chunk. The “I-XP” tag means that this word is inside an XP chunk. The “B-XP” by default means that the word is at the beginning of an XP chunk.
- **Grammatical function** of the two entities and words in the three contexts. The

[2] Software available at http://ilk.uvt.nl/∼sabine/chunklink/

last word in each chunk is its head, and the function of the head is the function of the whole chunk. "NP-SBJ" means a NP chunk as the subject of the sentence. The other words in a chunk that are not the head have "NOFUNC" as their function.

- **IOB-chains** of the heads of the two entities. So-called IOB-chain, noting the syntactic categories of all the constituents on the path from the root node to this leaf node of tree.

The position information is also specified in the description of each feature above. For example, word features with position information include:

1) WE1 (WE2): all words in e_1 (e_2)

2) WHE1 (WHE2): head word of e_1 (e_2)

3) WMNULL: no words in C_{mid}

4) WMFL: the only word in C_{mid}

5) WMF, WML, WM2, WM3, ...: first word, last word, second word, third word, ...in C_{mid} when at least two words in C_{mid}

6) WEL1, WEL2, ...: first word, second word, ... before e_1

7) WER1, WER2, ...: first word, second word, ... after e_2

We combine the above lexical and syntactic features with their position information in the contexts to form context vectors. Before that, we filter out low frequency features which appeared only once in the dataset.

3.2 Similarity Measures

The similarity s_{ij} between two occurrences of entity pairs is important to the performance of the LP algorithm. In this paper, we investigated two similarity measures, cosine similarity measure and Jensen-Shannon (JS) divergence (Lin, 1991). Cosine similarity is commonly used semantic distance, which measures the angle between two feature vectors. JS divergence has ever been used as distance measure for document clustering, which outperforms cosine similarity based document clustering (Slonim et al., 2002). JS divergence measures the distance between two probability distributions if feature vector is considered as probability distribution over features. JS divergence is defined as follows:

Table 1: Frequency of Relation SubTypes in the ACE training and devtest corpus.

Type	SubType	Training	Devtest
ROLE	General-Staff	550	149
	Management	677	122
	Citizen-Of	127	24
	Founder	11	5
	Owner	146	15
	Affiliate-Partner	111	15
	Member	460	145
	Client	67	13
	Other	15	7
PART	Part-Of	490	103
	Subsidiary	85	19
	Other	2	1
AT	Located	975	192
	Based-In	187	64
	Residence	154	54
SOC	Other-Professional	195	25
	Other-Personal	60	10
	Parent	68	24
	Spouse	21	4
	Associate	49	7
	Other-Relative	23	10
	Sibling	7	4
	GrandParent	6	1
NEAR	Relative-Location	88	32

$$JS(q,r) = \frac{1}{2}[D_{KL}(q\|\bar{p}) + D_{KL}(r\|\bar{p})] \tag{4}$$

$$D_{KL}(q\|\bar{p}) = \sum_y q(y)(\log\frac{q(y)}{\bar{p}(y)}) \tag{5}$$

$$D_{KL}(r\|\bar{p}) = \sum_y r(y)(\log\frac{r(y)}{\bar{p}(y)}) \tag{6}$$

where $\bar{p} = \frac{1}{2}(q + r)$ and $JS(q,r)$ represents JS divergence between probability distribution q(y) and r(y) (y is a random variable), which is defined in terms of KL-divergence.

3.3 Experimental Evaluation

3.3.1 Experiment Setup

We evaluated this label propagation based relation extraction method for relation subtype detection and characterization task on the official ACE 2003 corpus. It contains 519 files from sources including broadcast, newswire, and newspaper. We dealt with only intra-sentence explicit relations and assumed that all entities have been detected beforehand in the EDT sub-task of ACE. Table 1 lists the types and subtypes of relations for the ACE Relation Detection and Characterization (RDC) task, along with their

Table 2: The Performance of SVM and LP algorithm with different sizes of labeled data for relation detection on relation subtypes. The LP algorithm is run with two similarity measures: cosine similarity and JS divergence.

	SVM			LP_{Cosine}			LP_{JS}		
Percentage	P	R	F	P	R	F	P	R	F
1%	35.9	32.6	34.4	58.3	56.1	57.1	58.5	58.7	58.5
10%	51.3	41.5	45.9	64.5	57.5	60.7	64.6	62.0	63.2
25%	67.1	52.9	59.1	68.7	59.0	63.4	68.9	63.7	66.1
50%	74.0	57.8	64.9	69.9	61.8	65.6	70.1	64.1	66.9
75%	77.6	59.4	67.2	71.8	63.4	67.3	72.4	64.8	68.3
100%	79.8	62.9	70.3	73.9	66.9	70.2	74.2	68.2	71.1

Table 3: The performance of SVM and LP algorithm with different sizes of labeled data for relation detection and classification on relation subtypes. The LP algorithm is run with two similarity measures: cosine similarity and JS divergence.

	SVM			LP_{Cosine}			LP_{JS}		
Percentage	P	R	F	P	R	F	P	R	F
1%	31.6	26.1	28.6	39.6	37.5	38.5	40.1	38.0	39.0
10%	39.1	32.7	35.6	45.9	39.6	42.5	46.2	41.6	43.7
25%	49.8	35.0	41.1	51.0	44.5	47.3	52.3	46.0	48.9
50%	52.5	41.3	46.2	54.1	48.6	51.2	54.9	50.8	52.7
75%	58.7	46.7	52.0	56.0	52.0	53.9	56.1	52.6	54.3
100%	60.8	48.9	54.2	56.2	52.3	54.1	56.3	52.9	54.6

frequency of occurrence in the ACE training set and test set. We constructed labeled data by randomly sampling some examples from ACE training data and additionally sampling examples with the same size from the pool of unrelated entity pairs for the "NONE" class. We used the remaining examples in the ACE training set and the whole ACE test set as unlabeled data. The testing set was used for final evaluation.

3.3.2 LP vs. SVM

Support Vector Machine (SVM) is a state of the art technique for relation extraction task. In this experiment, we use LIBSVM tool [3] with linear kernel function.

For comparison between SVM and LP, we ran SVM and LP with different sizes of labeled data and evaluate their performance on unlabeled data using precision, recall and F-measure. Firstly, we ran SVM or LP algorithm to detect possible relations from unlabeled data. If an entity mention pair is classified not to the "NONE" class but to the other 24 subtype classes, then it has a relation. Then construct labeled datasets with different sampling set size *l*, including $1\% \times N_{train}$, $10\% \times N_{train}$, $25\% \times N_{train}$, $50\% \times N_{train}$, $75\% \times N_{train}$, $100\% \times N_{train}$ (N_{train} is the number of examples in the ACE training set). If any relation subtype was absent from the sampled labeled set, we redid the sampling. For each size, we performed 20 trials and calculated average scores on test set over these 20 random trials.

Table 2 reports the performance of SVM and LP with different sizes of labled data for relation detection task. We used the same sampled labeled data in LP as the training data for SVM model.

From Table 2, we see that both LP_{Cosine} and LP_{JS} achieve higher *Recall* than SVM. Specifically, with small labeled dataset (percentage of labeled data $\leq 25\%$), the performance improvement by LP is significant. When the percentage of labeled data increases from 50% to 100%, LP_{Cosine} is still comparable to SVM in *F-measure* while LP_{JS} achieves slightly better *F-measure* than SVM. On the other hand, LP_{JS} consistently outperforms LP_{Cosine}.

Table 3 reports the performance of relation classification by using SVM and LP with different sizes of labled data. And the performance describes the average values of *Precision*, *Recall* and *F-measure* over major relation subtypes.

From Table 3, we see that LP_{Cosine} and LP_{JS} outperform SVM by *F-measure* in almost all settings of labeled data, which is due to the increase of *Recall*. With smaller labeled dataset (percentage of labeled data $\leq 50\%$), the gap between LP and SVM is larger. When the percentage of labeled data in-

[3] $LIBSVM$: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

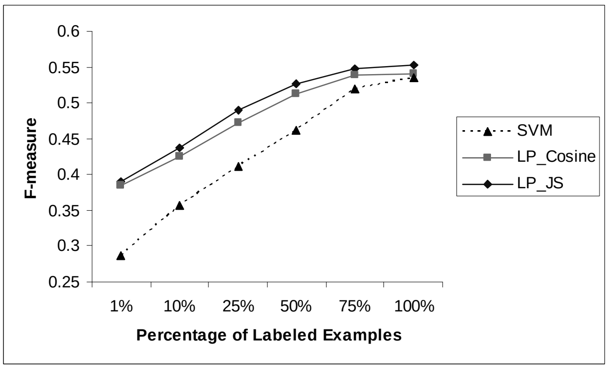


Figure 1: Comparison of the performance of SVM and LP with different sizes of labeled data

creases from 75% to 100%, the performance of LP algorithm is still comparable to SVM. On the other hand, the LP algorithm based on JS divergence consistently outperforms the LP algorithm based on Cosine similarity. Figure 1 visualizes the accuracy of three algorithms.

As shown in Figure 1, the gap between SVM curve and LP$_{JS}$ curves is large when the percentage of labeled data is relatively low.

3.3.3 An Example

In Figure 2, we selected 25 instances in training set and 15 instances in test set from the ACE corpus,which covered five relation types. Using $Isomap$ tool [4], the 40 instances with 229 feature dimensions are visualized in a two-dimensional space as the figure. We randomly sampled only one labeled example for each relation type from the 25 training examples as labeled data. Figure 2(a) and 2(b) show the initial state and ground truth result respectively. Figure 2(c) reports the classification result on test set by SVM ($accuracy = \frac{4}{15} = 26.7\%$), and Figure 2(d) gives the classification result on both training set and test set by LP ($accuracy = \frac{11}{15} = 73.3\%$).

Comparing Figure 2(b) and Figure 2(c), we find that many examples are misclassified from class $\diamond$ to other class symbols. This may be caused that SVMs method ignores the intrinsic structure in data. For Figure 2(d), the labels of unlabeled examples are determined not only by nearby labeled examples, but also by nearby unlabeled examples, so using LP

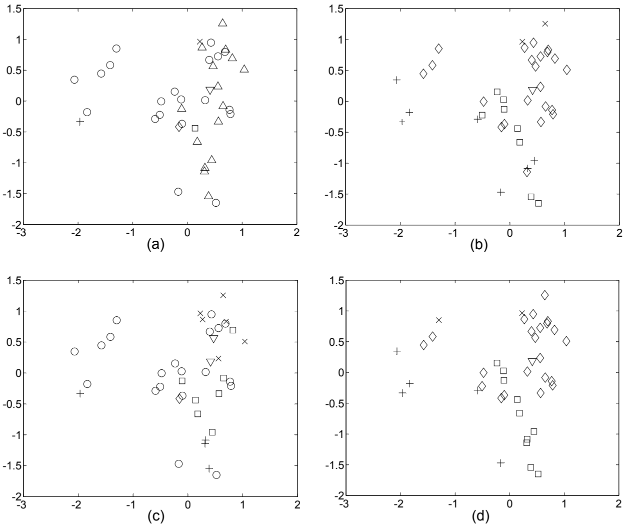


Figure 2: An example: comparison of SVM and LP algorithm on a data set from ACE corpus. $\circ$ and $\triangle$ denote the unlabeled examples in training set and test set respectively, and other symbols ($\diamond, \times, \Box, +$ and $\bigtriangledown$) represent the labeled examples with respective relation type sampled from training set.

strategy achieves better performance than the local consistency based SVM strategy when the size of labeled data is quite small.

3.3.4 LP vs. Bootstrapping

In (Zhang, 2004), they perform relation classification on ACE corpus with bootstrapping on top of SVM. To compare with their proposed Bootstrapped SVM algorithm, we use the same feature stream setting and randomly selected 100 instances from the training data as the size of initial labeled data.

Table 4 lists the performance of the bootstrapped SVM method from (Zhang, 2004) and LP method with 100 seed labeled examples for relation type classification task. We can see that LP algorithm outperforms the bootstrapped SVM algorithm on four relation type classification tasks, and perform comparably on the relation ”SOC” classification task.

4 Discussion

In this paper,we have investigated a graph-based semi-supervised learning approach for relation extraction problem. Experimental results showed that the LP algorithm performs better than SVM and

[4] The tool is available at http://isomap.stanford.edu/.

Table 4: Comparison of the performance of the bootstrapped SVM method from (Zhang, 2004) and LP method with 100 seed labeled examples for relation type classification task.

	Bootstrapping			LP$_{JS}$		
Relation type	P	R	F	P	R	F
ROLE	78.5	69.7	73.8	81.0	74.7	77.7
PART	65.6	34.1	44.9	70.1	41.6	52.2
AT	61.0	84.8	70.9	74.2	79.1	76.6
SOC	47.0	57.4	51.7	45.0	59.1	51.0
NEAR	–	–	–	13.7	12.5	13.0

Table 5: Comparison of the performance of previous methods on ACE RDC task.

		Relation Dectection			Relation Detection and Classification					
					on Types			on Subtypes		
	Method	P	R	F	P	R	F	P	R	F
Culotta and Soresen (2004)	Tree kernel based	81.2	51.8	63.2	67.1	35.0	45.8	-	-	-
Kambhatla (2004)	Feature based, Maximum Entropy	-	-	-	-	-	-	63.5	45.2	52.8
Zhou et al. (2005)	Feature based,SVM	84.8	66.7	74.7	77.2	60.7	68.0	63.1	49.5	55.5

bootstrapping. We have some findings from these results:

The LP based relation extraction method can use the graph structure to smooth the labels of unlabeled examples. Therefore, the labels of unlabeled examples are determined not only by the nearby labeled examples, but also by nearby unlabeled examples. For supervised methods, e.g., SVM, very few labeled examples are not enough to reveal the structure of each class. Therefore they can not perform well, since the classification hyperplane was learned only from few labeled data and the coherent structure in unlabeled data was not explored when inferring class boundary. Hence, our LP-based semi-supervised method achieves better performance on both relation detection and classification when only few labeled data is available. Bootstrapping

Currently most of works on the RDC task of ACE focused on supervised learning methods Culotta and Soresen (2004; Kambhatla (2004; Zhou et al. (2005). Table 5 lists a comparison on relation detection and classification of these methods. Zhou et al. (2005) reported the best result as 63.1%/49.5%/55.5% in *Precision/Recall/F-measure* on the relation subtype classification using feature based method, which outperforms tree kernel based method by Culotta and Soresen (2004). Compared with Zhou et al.'s method, the performance of LP algorithm is slightly lower. It may be due to that we used a much simpler feature set. Our work in this paper focuses on the investigation of a graph based semi-supervised learning algorithm for relation extraction. In the future, we would like to use more effective feature sets Zhou et al. (2005) or kernel based similarity measure with LP for relation extraction.

5 Conclusion and Future Work

This paper approaches the problem of semi-supervised relation extraction using a label propagation algorithm. It represents labeled and unlabeled examples and their distances as the nodes and the weights of edges of a graph, and tries to obtain a labeling function to satisfy two constraints: 1) it should be fixed on the labeled nodes, 2) it should be smooth on the whole graph. In the classification process, the labels of unlabeled examples are determined not only by nearby labeled examples, but also by nearby unlabeled examples. Our experimental results demonstrated that this graph based algorithm can achieve better performance than SVM when only very few labeled examples are available, and also outperforms the bootstrapping method for relation extraction task.

In the future, we would like to investigate more effective feature set or use feature selection to improve the performance of this graph-based semi-supervised relation extraction method.

References

Agichtein E. and Gravano L.. 2000. *Snowball: Extracting Relations from large Plain-Text Collections, In Proceedings of the 5^{th} ACM International Conference on Digital Libraries (ACMDL'00)*.

Belkin M. and Niyogi P.. 2002. *Using Manifold Structure for Partially Labeled Classification. Advances in Neural Infomation Processing Systems 15*.

Blum A. and Chawla S. 2001. *Learning from Labeled and Unlabeled Data Using Graph Mincuts. In Proceedings of the 18th International Conference on Machine Learning*.

Blum A., Lafferty J., Rwebangira R. and Reddy R. 2004. *Semi-Supervised Learning Using Randomized Mincuts. In Proceedings of the 21th International Conference on Machine Learning.*.

Brin Sergey. 1998. *Extracting patterns and relations from world wide web. In Proceedings of WebDB Workshop at 6th International Conference on Extending Database Technology (WebDB'98)*. pages 172-183.

Charniak E. 1999. *A Maximum-entropy-inspired parser. Technical Report CS-99-12*. Computer Science Department, Brown University.

Culotta A. and Soresen J. 2004. *Dependency tree kernels for relation extraction, In Proceedings of 42th Annual Meeting of the Association for Computational Linguistics*. 21-26 July 2004. Barcelona, Spain.

Hasegawa T., Sekine S. and Grishman R. 2004. *Discovering Relations among Named Entities from Large Corpora, In Proceeding of Conference ACL2004*. Barcelona, Spain.

Kambhatla N. 2004. *Combining lexical, syntactic and semantic features with Maximum Entropy Models for extracting relations, In Proceedings of 42th Annual Meeting of the Association for Computational Linguistics.*. 21-26 July 2004. Barcelona, Spain.

Lin J. 1991. *Divergence Measures Based on the Shannon Entropy. IEEE Transactions on Information Theory*. Vol 37, No.1, 145-150.

Miller S.,Fox H.,Ramshaw L. and Weischedel R. 2000. *A novel use of statistical parsing to extract information from text. In Proceedings of 6th Applied Natural Language Processing Conference* 29 April-4 may 2000, Seattle USA.

Slonim, N., Friedman, N., and Tishby, N. 2002. *Unsupervised Document Classification Using Sequential Information Maximization. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Yarowsky D. 1995. *Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*. pp.189-196.

Zelenko D., Aone C. and Richardella A. 2002. *Kernel Methods for Relation Extraction, Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Philadelphia.

Zhang Zhu. 2004. *Weakly-supervised relation classification for Information Extraction, In Proceedings of ACM 13th conference on Information and Knowledge Management (CIKM'2004)*. 8-13 Nov 2004. Washington D.C.,USA.

Zhou GuoDong, Su Jian, Zhang Jie and Zhang min. 2005. *Exploring Various Knowledge in Relation Extraction. In Proceedings of 43th Annual Meeting of the Association for Computational Linguistics*. USA.

Zhu Xiaojin and Ghahramani Zoubin. 2002. *Learning from Labeled and Unlabeled Data with Label Propagation. CMU CALD tech report CMU-CALD-02-107*.

Zhu Xiaojin, Ghahramani Zoubin, and Lafferty J. 2003. *Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In Proceedings of the 20th International Conference on Machine Learning*.

Polarized Unification Grammars

Sylvain Kahane
Modyco, Université Paris 10
sk@ccr.jussieu.fr

Abstract

This paper proposes a generic mathematical formalism for the combination of various structures: strings, trees, dags, graphs and products of them. The polarization of the objects of the elementary structures controls the saturation of the final structure. This formalism is both elementary and powerful enough to strongly simulate many grammar formalisms, such as rewriting systems, dependency grammars, TAG, HPSG and LFG.

1 Introduction

Our aim is to propose a generic formalism as simple as possible but powerful enough to write real grammars for natural language and to handle complex linguistic structures. The formalism we propose can strongly simulate most rule-based formalisms used in linguistics.[1]

Language utterances are both strongly structured and compositional and the structure of a complex utterance can be obtained by combining elementary structures associated to the elementary units of language.[2] The most simple way to combine two structures A and B is unification, that is, to build a new structure C by partially superimposing A and B and identifying a part of the objects of A with those of B. This idea recalls an old idea, used by Jespersen (1924), Tesnière (1934) or Ajduckiewicz (1935): the sentence is like a molecule whose words are atoms, each word bearing a valence (a linguistic term explicitly borrowed from chemistry) that forces or allows it to meet some other words. Nevertheless, unification grammars cannot directly take into account the fact that some linguistic units are unsaturated in a sense that they must absolutely combine with other structures to form a stable unit. Saturation is ensured by additional mechanisms, such as the distinction of terminal and non-terminal symbols in rewriting systems or by requiring some features to have an empty list as a value in HPSG.

This paper presents a new family of formalisms, *Polarized Unification Grammars* (PUGs). PUGs extend Unification Grammars with an explicit control of the saturation of structures by attributing a *polarity* to each object. Using polarities allows integrating the treatment of saturation in the formalism of the rules. Thus the processing of saturation will pilot the combination of structures during the generation processing. Some polarities are neutral, others are not, but a final structure must be completely neutral. Two non-neutral objects can unify (that is, identify) and form a neutral object (that is, neutralizing each other). Proper unification holds no equivalent.

Polarization takes its source in categorial grammar and subsequent works on resource-sensitive logic (see Lambek's, Girard's or van Benthem's works). Nasr (1995) is among the first to introduce a rule-based formalism using an explicit polarization of structures. Duchier & Thater (1999) propose a formalism for tree description where they put forward the notion of polarity (and they uses the terms of *polarity* and *neutralization*). Perrier (2000) is probably the

[1] A formalism A *strongly simulates* a formalism B if A has a better strong generative capacity than B, that is, if A can generate the languages generated by B with the same structures associated to the utterances of these languages.

[2] Whether a natural language utterance contains one or several structures depends on our point of view. On the one hand it is clear that a sentence can receive various structures according to the semantic, syntactic, morphological or phonological point of view. On the other hand these different structures are not independent from each other and even if they are not structures on the same objects (for instance the semantic units do not correspond one to one to the syntactic units, that is the words) there are links between the different objects of these structures. In other words, considering separately the different simple structures of the sentence does not take into account the whole structure of the sentence, because we lost the interrelation between structures of different levels.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 137–144,
Sydney, July 2006. ©2006 Association for Computational Linguistics

first to develop a linguistic formalism entirely based on these ideas, the *Interaction Grammar*.

PUG is both an elementary formalism (structures simply combine by identifying objects) and a powerful formalism, equivalent to Turing machines and capable of handling strings, trees, dags, *n*-graphs and products of such structures (such as ordered trees).[3] But, above all, PUG is a well-adapted formalism for writing grammars and it is capable of strongly simulating many classic formalisms.

Part 2 presents the general framework of PUG and its system of polarities. Part 3 proposes several examples of PUG and the translation in PUG of rewriting grammars, TAG, HPSG and LFG. We hope that these translations shed light on some common features of these formalisms.

2 Polarities and unification

2.1 Polarized Unification Grammars

Polarized Unification Grammars generate sets of finite structures. A structure is based on *objects*. For instance, for a (directed) graph, objects are *nodes* and *edges*. These two *types* of objects are linked, giving us the proper structure: if X is the set of nodes and U, the set of edges, the graph is defined by two *maps* π_1 and π_2 from U into X which associate an edge with its *source* and its *target*.

Our structures are *polarized*, that is, objects are associated to polarities. The set P of polarities is provided with an operation noted "**.**" and called *product*. The product is commutative and generally associative; (P, **.**) is called the *system of polarities*. A non-empty strict subset N of P contains the *neutral* polarities. A polarized structure is *neutral* if all its polarities are neutral.

Structures are defined on a collection of objects of various types (syntactic nodes, semantic nodes, syntactic edges …) and a collection of maps*: structural maps* linking objects to objects (such as *source* and *target* for edges), *label maps* linking objects to labels and *polarity maps* linking objects to polarities.

Structures combine by *unification*. The unification of two structures A and B gives a new structure A⊕B obtained by "pasting" together these structures and identifying a part of the objects of the first structure with objects of the second structure. When two polarized structures A and B are unified, the polarity of an object of A⊕B obtained by identifying two objects of A and B is the product of their polarities; if the two objects bear the same map, these maps must be identified and their values, unified. (For instance identifying two edges forces us to identify their sources and targets.)

A Polarized Unification Grammar (PUG) is defined by a finite family T of types of objects, a set of maps attached to the objects of each type, a system (P,**.**) of polarities, a subset N of P of neutral polarities, and a finite subset of elementary polarized structures, whose objects are described by T; one elementary structure is marked as the initial one and must be used exactly once. The structures *generated* by the grammar are the neutral structures obtained by combining the initial structure and a finite number of elementary structures. In the derivation process, elementary structures combine successively, each new elementary structure combining with at least one object of the previous result; this ensures that the derived structure is continuous. Polarities are only necessary to control the saturation and are not considered when the strong generative capacity of the grammar is estimated. Polarities belong to the declarative part of the grammar, but they rather play a role in the processing of the grammar.

2.2 The system of polarities

In this paper we will use the system of polarities P = {▨,□,–,+,■} (which are called like this: ■ = *black* = *saturated*, + = *positive*, – = *negative*, □ = *white* = *obligatory context* and ▨ = *grey* = *absolutely neutral*), with N = {▨,■}, and a product defined by the following array (where ⊥ represents the impossibility to combine). Note that ▨ is the neutral element of the product. The symbol – can be interpreted as a *need* and + as the corresponding *resource*.

.	▨	□	–	+	■
▨	▨	□	–	+	■
□	□	□	–	+	■
–	–	–	⊥	■	⊥
+	+	+	■	⊥	⊥
■	■	■	⊥	⊥	⊥

The system {□,■} is used by Nasr (1995), while the system {▨,■,–,+}, noted {=,↔,←,→}, is considered by Bonfante *et al.* (2004), who show advantages of negative and positive polarities for prefiltration in parsing (a set of structures bearing negative and positive polarities can only

[3] A *dag* is a directed acyclic graph. An *n*-graph is a graph whose nodes are edges of a (*n*-1)-graph and a 1-graph is a standard graph.

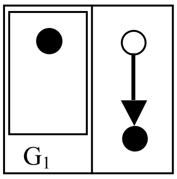
be reduced into a neutral structure if the sum of negative polarities of each object type is equal the sum of positive polarities).

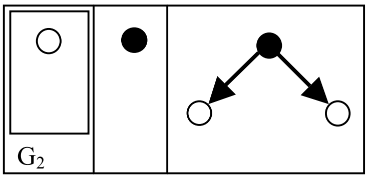
The system (P, **.**) we have presented is commutative and associative. Commutativity implies that the combination of two structures is not procedurally oriented (and we can begin a derivation by any elementary structure, provided we use only once the initial structure). Associativity implies that the combination of structures is unordered: if an object B must combine with A and C, there is no precedence order between the combination of A and B and the one of B and C, that is, A⊕(B⊕C) = (A⊕B)⊕C. If we leave polarities aside, our formalism is trivially monotonic: the combination of two structures A and B by a PUG gives us a structure A⊕B that contains A and B as substructures. We can add a (partial) order on P in order to make the formalism monotonic.[4] Let ≤ be this order. In order to give us a monotonic formalism, ≤ must verify the following *monotonicity property*: $\forall x,y \in \mathrm{P}\ x.y \geq x$. This provides us with the following order: ▨ < □ < +/– < ■. A PUG built with an ordered system of polarities (P, **.** ,≤) verifying the monotonicity property is monotonic. Monotonicity implies good computational properties; for instance it allows translating the parsing with PUG into a problem of constraint resolution (Duchier & Thater, 1999).

3 Examples of PUGs

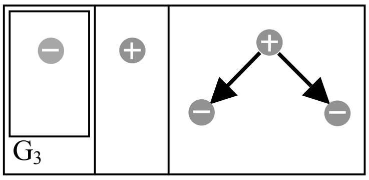
3.1 Tree grammars

The first tree grammars belonging to the paradigm of PUGs was proposed by Nasr 1995. The following grammar G_1 allows generating all finite *trees* (a tree is a connected directed graph such that every node except one is the target of at most one edge); objects are nodes and edges; the initial structure (the box on the left) is reduced to a black node; the grammar has only one other elementary structure, which is composed of a black edge linking a white node to a black node. Each white node must unify with a black node in order to be neutralized and each black node can unify with whatever number of white nodes. It can easily be verified that the structures generated by the grammar are trees, because every node has one and only one governor, except the node introduced by the initial structure, which is the root of the tree.

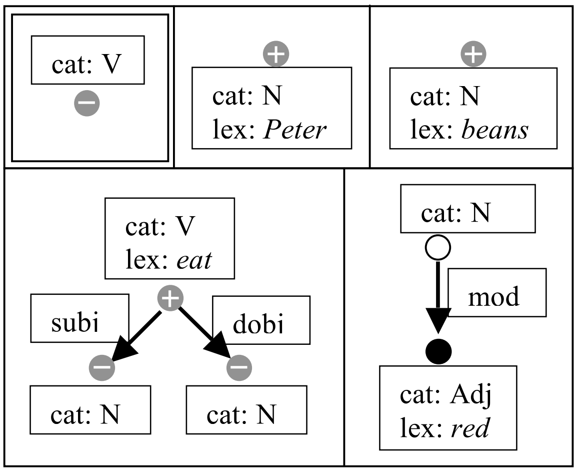




The grammar G_1 does not control the number of dependents of nodes. A grammar like G_2 allows controlling the valence of each node, but it does not ensure that generated structures are trees, because two white nodes can unify and a node can have more than one governor.[5] The grammar G_3 solves the problem. In fact, G_3 can be viewed as the superimposition of G_1 and G_2. Indeed, if P_0 = {□,■}, $P_1 = P_0 \times P_0$ = {(□,□),(□,■),(■,□),(■,■)} is equivalent to {□,+,–,■}. The first polarity controls the tree structure as G_1 does, while the second polarity controls the valence as G_2 does.



With the same principles, one can build a *dependency grammar* generating the syntactic dependency trees of a fragment of natural language. Grammar G_4, directly inspired from Nasr 1995, proposes a fragment of grammar for English generating the syntactic tree of *Peter eats red beans*. Nodes of this grammar are labeled by two label maps, /cat/ and /lex/. Note that the root of



G_4 (Dependency grammar for English)

[4] I was suggested this idea by Guy Perrier.

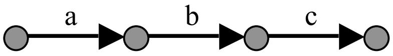
[5] Nasr 1995 proposes such a grammar in order to generate trees. He uses an external requirement, which forces, when two structures are combined, the root of one to combine with a node of the other one.

the elementary structure of an adjective is a white node, allowing an unlimited number of such structures to adjoin to a noun.

3.2 Rewriting systems and ordered trees

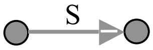
PUG can simulate any rewriting system and have the weak generative capacity of Turing machines. We follow ideas developed by Burroni 1993 or Dymetman 1999, themselves following van Kampen 1933's ideas.

A sequence *abc* is represented by a string of labeled edges *a, b* and *c*:

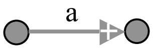


Intuitively, edges are intervals and nodes model their extremities. This is the simplest way to model linear order and precedence rules: X precedes Y iff the end of X is the beginning of Y.

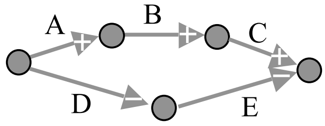
The initial category S of the grammar gives us the initial structure:



A terminal symbol *a* corresponds to a positive edge:

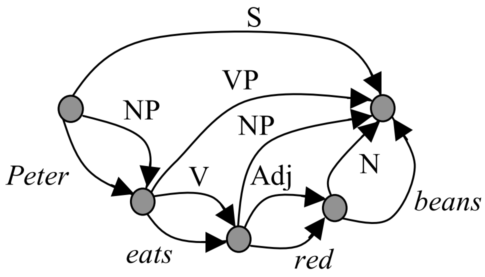


A rewriting rule ABC → DE gives us the elementary structure:

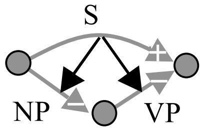


This elementary structure is a "cell" whose upper frontier is a string of positive edges corresponding to the left part of the rule, while the lower frontier is a string of negative edges corresponding to the right part of the rule. Each positive edge must unify with a negative edge and vice versa, in order to give a black edge. Nodes are grey (= absolutely neutral) and their unification is entirely driven by the unification of edges.

Cells will unify with each other to give a final structure representing the derivation structure of a sequence, which is the lower edge of this structure. The next figure shows the derivation structure of the sequence *Peter eats red beans* with a standard phrase structure grammar, which can be reconstructed by the reader. In such a representation, edges represent phrases and correspond to intervals in the cutting of the sequence, while nodes are bounds of these intervals.



For a *context-free rewriting system*, the grammar generates the derivation tree, which can be represented in a more traditional way by adding the branches of the tree (giving us a 2-graph).



Let us recall that a derivation tree for a context-free grammar is an ordered tree. An *ordered tree* combines two structures on the same set of nodes: a structure of tree and a precedence relation on the node of the tree. Here the precedence relation is explicitly represented (a "node" of the tree precedes another "node" if the target of the first one is the source of the second one). It is not possible, with a PUG, to generate the derivation tree, including the precedence relation, in a simpler way.[6] Note that the usual representation of ordered trees (where the precedence relation is not explicit, but only deductible from the planarity of the representation) is very misleading from the computational viewpoint. When they calculate the precedence relation, parsers (of the CKY type for instance) in fact calculate a data structure like the one we present here, where beginnings and ends of phrase are explicitly considered as objects.

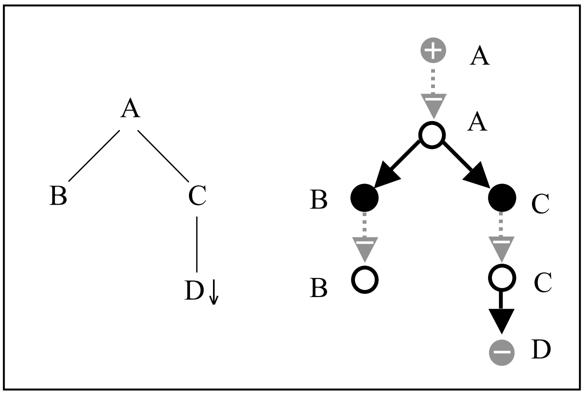
3.3 TAG (Tree Adjoining Grammar)

PUG has a clear kinship with TAG, which is the first formalism based on combination of structures to be studied at length. TAGs are generally presented as grammars combining (ordered) trees. In fact, as a tree grammar, TAG is not

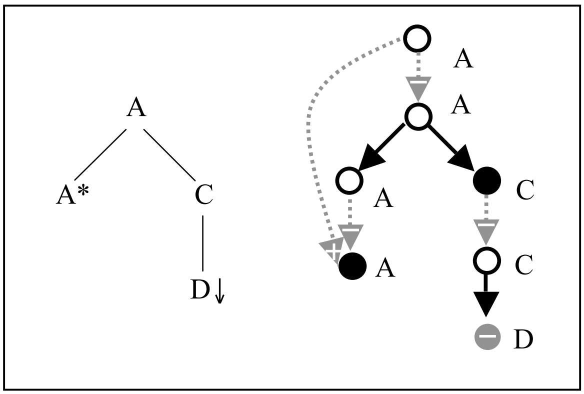
[6] The most natural idea would be to encode a rewriting rule with a tree of depth 1 and the precedence relation with edges from a node to its successor. The difficulty is then to propagate the order relation to the descendants of two sister nodes when we apply a rewriting rule by substituting a tree of depth 1. The simplest solution is undeniably the one presented here, consisting to introduce objects representing the beginning and the end of phrases (our grey nodes) and to indicate the relation between a phrase, its beginning and its end by representing the phrase with an edge from the beginning to the end.

monotonic and cannot be simulated with PUG. As shown by Vijay-Shanker 1992, to obtain a monotonic formalism, TAG must be viewed as a grammar combining quasi-trees. Intuitively, a *quasi-tree* is a tree whose nodes has been split in two parts and have each one an upper part and a lower part, between which another quasi-tree can be inserted (this is the famous adjoining operation of TAG). Formally, a quasi-tree is a tree whose branches have two types: dependency relations and dominance relations (respectively noted by plain lines and dotted lines). Two nodes linked by a negative dominance relation are potentially the two parts of a same node; only the lower part can have dependents.

The next figures give an α-tree (= to be substituted) and a β-tree (= to be adjoined) with the corresponding PUG structures.[7] A substitution node (like D↓) gives a negative node, which will unify with the root of an α tree. A β-tree gives a white root node and a black foot node, which will unify with the upper and the lower part of a split node. This is why the root and the foot node are linked by a positive dominance link, which will unify with a negative dominance link connecting the two parts of a split node.

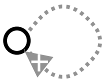


An α tree and its PUG translation



A β tree and its PUG translation

At the end of the derivation, the structure must be a tree and all nodes must be reconstructed: this is why we introduce the next rule, which presents a positive dominance link linking a node to itself and which will force two semi-nodes to unify by neutralizing the dominance link between them.



This last rule again shows the advantage of PUG: the reunification of nodes, which is procedurally ensured in Vijay-Shanker 1992 is given here as a declarative rule.

3.4 HPSG (Head-driven Phrase Structure Grammar)

There are two ways to translate feature structures (FSs) into PUG. Clearly atomic values must be labels and (embedded) feature structures must be nodes, but features can be translated by maps or by edges, that is, objects. Encoding features by maps ensures to identify them in PUG. Encoding them by edges allows us to polarize them and control the number of identifications.[8]

For the sake of clarification of HSPG structures, we choose to translate structural features such as HDTR and NHDTR, which give the phrase structure and which never unify with other "features", by edges and other features by maps (which will be represented by hashed arrows). In any case, the result looks like a dag whose "edges" (true edges and maps) represent features and whose nodes represent values (e.g. Kesper & Mönnich 2003). We exemplify the translation of HPSG in PUG with the schema of combination

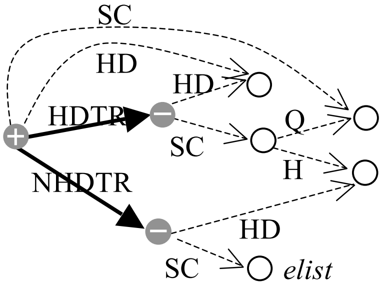
[7] For sake of simplicity, we leave aside the precedence relation on sister nodes. It might be encoded in the same way as context-free rewriting systems, by modeling semi-nodes of TAG trees by edges. It does not pose any problem but would make the figures difficult to read.

[8] Perrier 2000 uses a feature-structure based formalism where only features are polarized. Although more or less equivalent we prefer to polarize the FS themselves, i.e. the nodes.

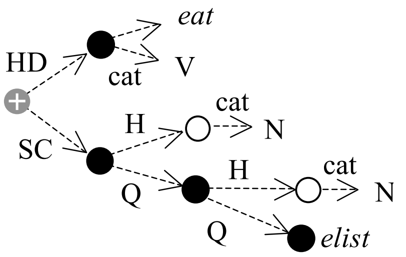
of head phrase with a subcategorized sister phrase, namely the *head-daughter-phrase*:[9]

$$\left[\begin{array}{ll} \text{HEAD:}\boxed{1} & \\ \text{SUBCAT:}\boxed{3} & \\ \text{HDTR:} & \left[\begin{array}{l}\text{HEAD:}\boxed{1}\\ \text{SUBCAT:}\langle\boxed{2}\rangle\oplus\boxed{3}\end{array}\right] \\ \text{NHDTR:} & \left[\begin{array}{l}\text{HEAD:}\boxed{2}\\ \text{SUBCAT}: \textit{elist}\end{array}\right] \end{array}\right]$$

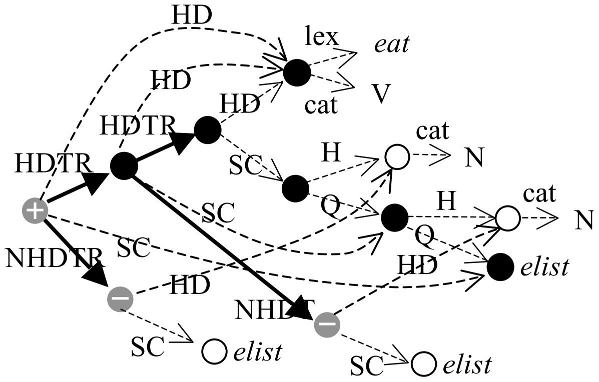
This FS gives the following structure, where a list is represented recursively in two pieces: its head (value of H) and its queue (value of Q).



A negative node of this FS can be neutralized by the combination with a similar FS representing a phrase or with a lexical entry. The next figure proposes a lexical entry for *eat*, indicating that *eat* is a V whose SUBCAT list contains two phrases headed by an N (for sake of simplicity we deal with the subject as a subcategorized phrase).



The combination of two *head-daughter-phrases* with the lexical entry of *eat* gives us the previous lexicalized rule, equivalent to the rule for *eat* of the dependency grammar G_4 (/subj/ is the NHDTR of the maximal projection and /obj/ the NHDTR of the intermediate projection of *eat*).



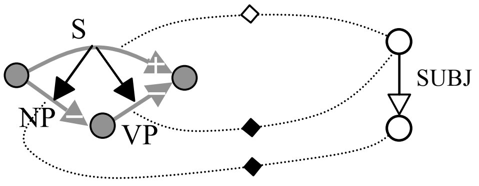
Polarization of objects shows exactly what is constructed by each rule and what are the requests filled by other rules. Moreover it allows us to force SUBCAT lists to be instantiated (and therefore allows us to control the saturation of the valence), which is ensured in the usual formalism of HPSG by a bottom-up procedural presentation.

3.5 LFG (Lexical Functional Grammar) and synchronous grammars

We propose a translation of LFG into PUG that makes LFG appear as a synchronous grammar approach (see Shieber & Schabes 1990). LFG synchronizes two structures (a phrase structure or c-structure and a dependency/functional structure or f-structure) and it can be viewed as the synchronization of a phrase structure grammar and a dependency grammar.

Let us consider a first LFG rule and its translation in PUG:

$$[1]\quad S \;\rightarrow\; \underset{\downarrow\,=\,\uparrow\,\text{SUBJ}}{NP} \qquad \underset{\downarrow\,=\,\uparrow}{VP}$$



Equations under phrases (in the right side of [1]) ensure the synchronization between the objects of the c-structure and the f-structure: each phrase is synchronized with a "functional" node. Symbols ↓ and ↑ respectively designate the functional node synchronized with the current phrase and the one synchronized with the mother phrase (here S). Thus the equation ↓=↑ means that the current phrase (VP) and its mother (S) are synchronized with the same functional node. The

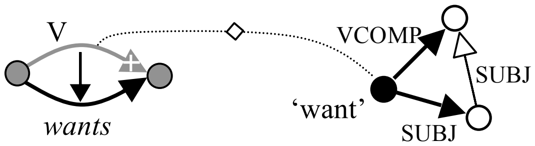
[9] Numbers in boxes are values shared by several features. The value of SUBCAT (= SC) is a list (the list of subcategorized phrases). The non-head daughter phrase (NHDTR) has a saturated valence and so needs an empty SUBCAT list (*elist*). The subcat list of the head daughter phrase (HDTR) is the concatenation, noted ⊕, of two lists: a list with one element that is the description of the non-head daughter phrase and the SUBCAT list of the whole phrase. The rest of the description of this phrase (value of HEAD) is equal to the one of the head daughter phrase.

expression ↑ SUBJ designates the functional node depending on ↑ by the relation SUBJ.

In PUG we model the synchronization of the phrases and the functional nodes by *synchronization links* (represented by dotted lines with diamond-shaped polarities) (see Bresnan 2000 for non-formalized similar representations). The two synchronizations ensured by the two constraints ↓=↑ SUBJ and ↓=↑ of [1], and therefore built by this rule, are polarized in black.

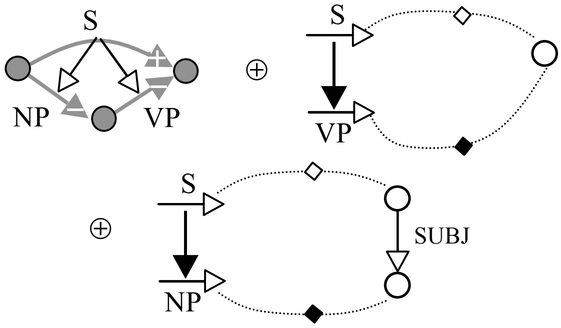
A phrasal rule such as [1] introduces an f-structure with a totally white polarization. It will be neutralized by lexical rules such as [2]:

[2] V → *wants*
↑ PRED = 'want ⟨SUBJ,VCOMP⟩'
↑ SUBJ = ↑ VCOMP SUBJ



The feature Pred is interpreted as the labeling of the functional node, while the valence ⟨SUBJ,VCOMP⟩ gives us two black edges and two white nodes. The functional equation ↑SUBJ = ↑ VCOMP SUBJ introduces a white edge SUBJ between the nodes ↑ SUBJ and ↑VCOMP (and is therefore to be interpreted very differently from the constraints of [1], which introduce black synchronization links.)

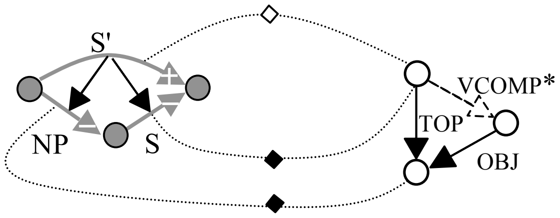
PUG allows to easily split up a rule into more elementary rules. For instance, the rule [1] can be split up into three rules: a phrase structure rules linearizing the daughter phrases and two rules of synchronization indicating the functional link between a phrase and one of its daughter phrases.



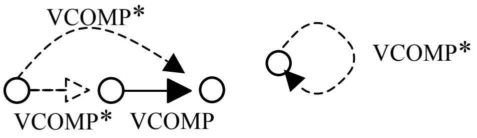
Our decomposition shows that LFG articulated two different grammars: a classical phrase structure generating the c-structure and an interface grammar between c- and f-structures (and even a third grammar because the f-structure is really generated only by the lexical rules). With PUG it is easy to join two (or more) grammars: it suffices to add on the objects by both grammars a white polarity that will be saturated in the other grammar (and vice versa) (Kahane & Lareau 2005).

Let us consider another problem, illustrated here by the rule for the topicalization of an object. The unbounded dependency of the object with its functional governor is an undetermined path expressed by a regular expression (here VCOMP* OBJ; functional uncertainty, Kaplan & Zaenen 1989).

[3] S' → NP S
↓ = ↑ VCOMP* OBJ ↓ = ↑
↓ = ↑ TOP



The path VCOMP* (represented by a dashed arrow) is expanded by the following regular grammar, with two rules, one for the propagation and one for the ending.



Again the translation into PUG brings to the fore some fundamental components of the formalism (like synchronization links) and some non-explicit mechanisms such as the fact that the lexical equation ↑ PRED = 'want ⟨SUBJ,VCOMP⟩' introduces both resources (a node 'want') and needs (its valence).

4 Conclusion

The PUG formalism is extremely simple: it only imposes that combining two structures involves at least the unification of two objects. Forcing or forbidding more objects to combine is then entirely controlled by polarization of objects. Polarization will thus guide the process of combination of elementary structures. In spite of its simplicity, the PUG formalism is powerful enough to elegantly simulate most of the rule-based formalisms used in formal linguistics and NLP. This sheds new light on these formalisms and allows us to bring to the fore the exact nature of the structures they handle and to extract some procedural mechanisms hidden by the formalism. But above all, the PUG formalism allows us to write separately several modules of the grammar handling various structures and to put them together in a same formalism by synchronization of the grammars, as we show with our translation of LFG. Thus PUGs extend unification grammars based on feature structures by allowing a greatest diversity of geometric structures and a best control of resources. Further investigations must concern the computational properties of PUGs, notably restrictions allowing polynomial time parsing.

Acknowledgements

I thank Benoît Crabbé, Denys Duchier, Kim Gerdes, François Lareau, François Métayer, Piet Mertens, Guy Perrier, Alain Polguère and Benoît Sagot for their numerous remarks and enlightening commentaries.

References

Ajdukiewicz K. 1935. Die syntaktische Konnexität. *Studia Philosophica,* 1:1-27.

Bonfante G., Guillaume B., Perrier G. 2004. Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation. *Proceedings of CoLing*, Genève, Switzerland, 303-309.

Bresnan J. 2001. *Lexical-Functional Syntax*. Blackwell.

Burroni A. 1993. Higher-dimensional word problems with applications to equational logic. *Theoretical Computer Sciences*, 115:43-62.

Duchier D., Thater S. 1999. Parsing with tree descriptions: a constraint-based approach. *NLULP 1999 (Natural Language Understanding and Logic Programming)*, Las Cruces, NM.

Dymetman M. 1999. Some Remarks on the Geometry of Grammar. *Proceedings of MOL'6 (6th Meeting on the Mathematics of Language)*, Orlando, FA.

Girard J.-Y. 1987. Linear logic. *Theoretical Computer Sciences,* 50(1):1-102.

Kahane S., Lareau F. 2005. Meaning-Text Unification Grammar: modularity and polarization, *Second International Conference on Meaning-Text Theory*, Moscow, 197-206.

Kaplan R.M., Zaenen A. 1989. Long-distance dependencies, constituent structure, and functional uncertainty. In *Alternative Conceptions of Phrase Structure*, M. Baltin & A. Kroch (eds), 17-42, Chicago Univ. Press.

Kepser S., Mönnich U. 2003. Graph properties of HPSG feature structures. *Proceedings of Formal Grammar*, Vienna, Austria, 115-124.

Nasr A. 1995. A formalism and a parser for lexicalised dependency grammars. *Proceedings of the 4th Int. Workshop on Parsing Tecnologies*, State Univ. of NY Press.

Perrier G. 2000. Interaction Grammars. *Proceedings of CoLing*, Sarrebrücken, 7 p.

Shieber S. M. & Schabes Y. 1990. Synchronous tree-adjoining grammars, *Proceedings of CoLing*, vol. 3, 253-258, Helsinki, Finland.

Tesnière L. 1934. Comment construire une syntaxe. *Bulletin de la Faculté des Lettres de Strasbourg,* 7: 219-229.

van Kampen E. 1933. On some lemmas in the theory of groups. *American Journal of Mathematics,* 55:268–73.

Vijay-Shanker K. 1992. Using description of trees in a Tree Adjoining Grammar. *Computational Linguistics*, 18(4):481-517.

Partially Specified Signatures: a Vehicle for Grammar Modularity

Yael Cohen-Sygal
Dept. of Computer Science
University of Haifa
yaelc@cs.haifa.ac.il

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

Abstract

This work provides the essential foundations for modular construction of (typed) unification grammars for natural languages. Much of the information in such grammars is encoded in the signature, and hence the key is facilitating a modularized development of type signatures. We introduce a definition of signature modules and show how two modules combine. Our definitions are motivated by the actual needs of grammar developers obtained through a careful examination of large scale grammars. We show that our definitions meet these needs by conforming to a detailed set of desiderata.

1 Introduction

Development of large scale grammars for natural languages is an active area of research in human language technology. Such grammars are developed not only for purposes of theoretical linguistic research, but also for natural language applications such as machine translation, speech generation, etc. Wide-coverage grammars are being developed for various languages (Oepen et al., 2002; Hinrichs et al., 2004; Bender et al., 2005; King et al., 2005) in several theoretical frameworks, e.g., LFG (Dalrymple, 2001) and HPSG (Pollard and Sag, 1994).

Grammar development is a complex enterprise: it is not unusual for a single grammar to be developed by a team including several linguists, computational linguists and computer scientists. The scale of grammars is overwhelming: for example, the English resource grammar (Copestake and Flickinger, 2000) includes thousands of types. This raises problems reminiscent of those encountered in large-scale software development. Yet while software engineering provides adequate solutions for the programmer, no grammar development environment supports even the most basic needs, such as grammar modularization, combination of sub-grammars, separate compilation and automatic linkage of grammars, information encapsulation, etc.

This work provides the essential foundations for modular construction of signatures in typed unification grammars. After a review of some basic notions and a survey of related work we list a set of desiderata in section 4, which leads to a definition of signature modules in section 5. In section 6 we show how two modules are combined, outlining the mathematical properties of the combination (proofs are suppressed for lack of space). Extending the resulting module to a stand-alone type signature is the topic of section 7. We conclude with suggestions for future research.

2 Type signatures

We assume familiarity with theories of (typed) unification grammars, as formulated by, e.g., Carpenter (1992) and Penn (2000). The definitions in this section set the notation and recall basic notions. For a partial function F, '$F(x)\downarrow$' means that F is defined for the value x.

Definition 1 *Given a partially ordered set $\langle P, \leq \rangle$, the set of* **upper bounds** *of a subset $S \subseteq P$ is the set $S^u = \{y \in P \mid \forall x \in S \;\; x \leq y\}$.*

For a given partially ordered set $\langle P, \leq \rangle$, if $S \subseteq P$ has a least element then it is unique.

Definition 2 *A partially ordered set $\langle P, \leq \rangle$ is a* **bounded complete partial order** *(BCPO) if for every $S \subseteq P$ such that $S^u \neq \emptyset$, S^u has a least element, called a* **least upper bound (lub)**.

Definition 3 *A* **type signature** *is a structure $\langle \textsc{Type}, \sqsubseteq, \textsc{Feat}, Approp \rangle$, where:*

1. *$\langle \textsc{Type}, \sqsubseteq \rangle$ is a finite bounded complete partial order (the* **type hierarchy***)*

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 145–152,
Sydney, July 2006. ©2006 Association for Computational Linguistics

2. FEAT *is a finite set, disjoint from* TYPE.

3. $Approp$: TYPE × FEAT → TYPE *(the* ***appropriateness specification****) is a partial function such that for every* $F \in$ FEAT*:*

   (a) ***(Feature Introduction)*** *there exists a type* $Intro(F) \in$ TYPE *such that* $Approp(Intro(F), F)\downarrow$*, and for every* $t \in$ TYPE*, if* $Approp(t, F)\downarrow$*, then* $Intro(F) \sqsubseteq t$*;*

   (b) ***(Upward Closure)*** *if* $Approp(s, F)\downarrow$ *and* $s \sqsubseteq t$*, then* $Approp(t, F)\downarrow$ *and* $Approp(s, F) \sqsubseteq Approp(t, F)$*.*

Notice that every signature has a least type, since the subset $S = \emptyset$ of TYPE has the non-empty set of upper bounds, $S^u =$ TYPE, which must have a least element due to bounded completeness.

Definition 4 *Let* $\langle$TYPE$, \sqsubseteq\rangle$ *be a type hierarchy and let* $x, y \in$ TYPE*. If* $x \sqsubseteq y$*, then* x *is a* **supertype** *of* y *and* y *is a* **subtype** *of* x*. If* $x \sqsubseteq y$*,* $x \neq y$ *and there is no* z *such that* $x \sqsubseteq z \sqsubseteq y$ *and* $z \neq x, y$ *then* x *is an* **immediate supertype** *of* y *and* y *is an* **immediate subtype** *of* x*.*

3 Related Work

Several authors address the issue of grammar modularization in unification formalisms. Moshier (1997) views HPSG , and in particular its signature, as a collection of constraints over maps between sets. This allows the grammar writer to specify any partial information about the signature, and provides the needed mathematical and computational capabilities to integrate the information with the rest of the signature. However, this work does not define modules or module interaction. It does not address several basic issues such as bounded completeness of the partial order and the feature introduction and upward closure conditions of the appropriateness specification. Furthermore, Moshier (1997) shows how signatures are distributed into components, but not the conditions they are required to obey in order to assure the well-definedness of the combination.

Keselj (2001) presents a modular HPSG, where each module is an ordinary type signature, but each of the sets FEAT and TYPE is divided into two disjoint sets of private and public elements. In this solution, modules do not support specification of partial information; module combination is not associative; and the only channel of interaction between modules is the names of types.

Kaplan et al. (2002) introduce a system designed for building a grammar by both extending and restricting another grammar. An LFG grammar is presented to the system in a priority-ordered sequence of files where the grammar can include only one definition of an item of a given type (e.g., rule) with a particular name. Items in a higher priority file override lower priority items of the same type with the same name. The override convention makes it possible to add, delete or modify rules. However, a basis grammar is needed and when modifying a rule, the entire rule has to be rewritten even if the modifications are minor. The only interaction among files in this approach is overriding of information.

King et al. (2005) augment LFG with a makeshift signature to allow modular development of *untyped* unification grammars. In addition, they suggest that any development team should agree in advance on the feature space. This work emphasizes the observation that the modularization of the signature is the key for modular development of grammars. However, the proposed solution is adhoc and cannot be taken seriously as a concept of modularization. In particular, the suggestion for an agreement on the feature space undermines the essence of modular design.

Several works address the problem of modularity in other, related, formalisms. Candito (1996) introduces a description language for the trees of LTAG. Combining two descriptions is done by conjunction. To constrain undesired combinations, Candito (1996) uses a finite set of names where each node of a tree description is associated with a name. The only channel of interaction between two descriptions is the names of the nodes, which can be used only to allow identification but not to prevent it. To overcome these shortcomings, Crabbé and Duchier (2004) suggest to replace node naming by colors. Then, when unifying two trees, the colors can prevent or force the identification of nodes. Adapting this solution to type signatures would yield undesired order-dependence (see below).

4 Desiderata

To better understand the needs of grammar developers we carefully explored two existing grammars: the LINGO grammar matrix (Bender et al., 2002), which is a basis grammar for the rapid development of cross-linguistically consistent grammars; and a grammar of a fragment of Modern Hebrew, focusing on inverted constructions (Melnik, 2006). These grammars were chosen since they are comprehensive enough to reflect the kind of data large scale grammar encode, but are not too large to encumber this process. Motivated by these two grammars, we experimented with ways to divide the signatures of grammars into modules and with different methods of module interaction. This process resulted in the following desiderata for a beneficial solution for signature modularization:

1. The grammar designer should be provided with as much flexibility as possible. Modules should not be unnecessarily constrained.
2. Signature modules should provide means for specifying *partial* information about the components of a grammar.
3. A good solution should enable one module to refer to types defined in another. Moreover, it should enable the designer of module M_i to use a type defined in M_j without specifying the type explicitly. Rather, some of the attributes of the type can be (partially) specified, e.g., its immediate subtypes or its appropriateness conditions.
4. While modules can specify partial information, it must be possible to deterministically extend a module (which can be the result of the combination of several modules) into a full type signature.
5. Signature combination must be associative and commutative: the order in which modules are combined must not affect the result.

The solution we propose below satisfies these requirements.[1]

5 Partially specified signatures

We define *partially specified signatures (PSSs)*, also referred to as *modules* below, which are structures containing partial information about a signature: part of the subsumption relation and part of the appropriateness specification. We assume enumerable, disjoint sets TYPE of types and FEAT of features, over which signatures are defined. We begin, however, by defining *partially labeled graphs*, of which PSSs are a special case.

Definition 5 *A* ***partially labeled graph (PLG)*** *over* TYPE *and* FEAT *is a finite, directed labeled graph $S = \langle Q, T, \preceq, Ap\rangle$, where:*

1. *Q is a finite, nonempty set of nodes, disjoint from* TYPE *and* FEAT.
2. *$T : Q \to$* TYPE *is a partial function, marking some of the nodes with types.*
3. *$\preceq \subseteq Q \times Q$ is a relation specifying (immediate) subsumption.*
4. *$Ap \subseteq Q \times$* FEAT *$\times Q$ is a relation specifying appropriateness.*

Definition 6 *A* ***partially specified signature (PSS)*** *over* TYPE *and* FEAT *is a PLG $S = \langle Q, T, \preceq, Ap\rangle$, where:*

1. *T is one to one.*
2. *'$\preceq$' is antireflexive; its reflexive-transitive closure, denoted '$\overset{*}{\preceq}$', is antisymmetric.*
3. (a) *(Relaxed Upward Closure) for all $q_1, q_1', q_2 \in Q$ and $F \in$* FEAT*, if $(q_1, F, q_2) \in Ap$ and $q_1 \overset{*}{\preceq} q_1'$, then there exists $q_2' \in Q$ such that $q_2 \overset{*}{\preceq} q_2'$ and $(q_1', F, q_2') \in Ap$; and*
   (b) *(Maximality) for all $q_1, q_2 \in Q$ and $F \in$* FEAT*, if $(q_1, F, q_2) \in Ap$ then for all $q_2' \in Q$ such that $q_2' \overset{*}{\preceq} q_2$ and $q_2 \neq q_2'$, $(q_1, F, q_2') \notin Ap$.*

A PSS is a finite directed graph whose nodes denote types and whose edges denote the subsumption and appropriateness relations. Nodes can be *marked* by types through the function T, but can also be *anonymous* (unmarked). Anonymous nodes facilitate reference, in one module, to types that are defined in another module. T is one-to-one since we assume that two marked nodes denote different types.

The '$\preceq$' relation specifies an immediate subsumption order over the nodes, with the intention that this order hold later for the types denoted by nodes. This is why '$\overset{*}{\preceq}$' is required to be a partial order. The type hierarchy of a type signature is a BCPO, but current approaches (Copestake, 2002) relax this requirement to allow more flexibility in grammar design. PSS subsumption is also a partial order but not necessarily a bounded complete

[1] The examples in the paper are inspired by actual grammars but are obviously much simplified.

one. After all modules are combined, the resulting subsumption relation will be extended to a BCPO (see section 7), but any intermediate result can be a general partial order. Relaxing the BCPO requirement also helps guaranteeing the associativity of module combination.

Consider now the appropriateness relation. In contrast to type signatures, Ap is not required to be a function. Rather, it is a relation which may specify *several* appropriate nodes for the values of a feature F at a node q. The intention is that the eventual value of $Approp(T(q), F)$ be the *lub* of the types of all those nodes q' such that $Ap(q, F, q')$. This relaxation allows more ways for modules to interact. We do restrict the Ap relation, however. Condition 3a enforces a relaxed version of upward closure. Condition 3b disallows redundant appropriateness arcs: if two nodes are appropriate for the same node and feature, then they should not be related by subsumption. The feature introduction condition of type signatures is not enforced by PSSs. This, again, results in more flexibility for the grammar designer; the condition is restored after all modules combine, see section 7.

Example 1 *A simple PSS S_1 is depicted in Figure 1, where solid arrows represent the '$\preceq$' (subsumption) relation and dashed arrows, labeled by features, the Ap relation. S_1 stipulates two subtypes of cat, n and v, with a common subtype, $gerund$. The feature* AGR *is appropriate for all three categories, with distinct (but anonymous) values for $Approp(n,\text{AGR})$ and $Approp(v,\text{AGR})$. $Approp(gerund,\text{AGR})$ will eventually be the lub of $Approp(n,\text{AGR})$ and $Approp(v,\text{AGR})$, hence the multiple outgoing* AGR *arcs from $gerund$.*

Observe that in S_1, '$\preceq$' is not a BCPO, Ap is not a function and the feature introduction condition does not hold.

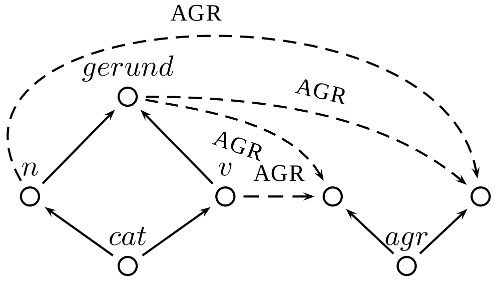


Figure 1: A partially specified signature, S_1

We impose an additional restriction on PSSs: a PSS is *well-formed* if any two different anonymous nodes are *distinguishable*, i.e., if each node is unique with respect to the information it encodes. If two nodes are indistinguishable then one of them can be removed without affecting the information encoded by the PSS. The existence of indistinguishable nodes in a PSS unnecessarily increases its size, resulting in inefficient processing.

Given a PSS S, it can be *compacted* into a PSS, $compact(S)$, by unifying all the indistinguishable nodes in S. $compact(S)$ encodes the same information as S but does not include indistinguishable nodes. Two nodes, only one of which is anonymous, can still be otherwise indistinguishable. Such nodes will, eventually, be coalesced, but only after all modules are combined (to ensure the associativity of module combination). The detailed computation of the compacted PSS is suppressed for lack of space.

Example 2 *Let S_2 be the PSS of Figure 2. S_2 includes two pairs of indistinguishable nodes: q_2, q_4 and q_6, q_7. The compacted PSS of S_2 is depicted in Figure 3. All nodes in $compact(S_2)$ are pairwise distinguishable.*

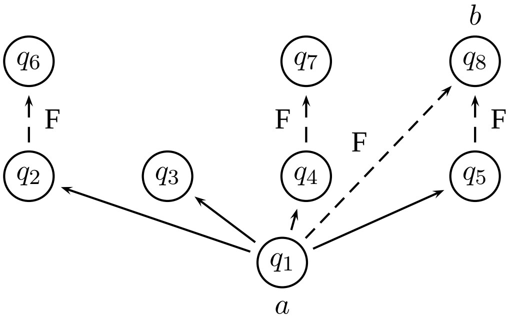


Figure 2: A partially specified signature with indistinguishable nodes, S_2

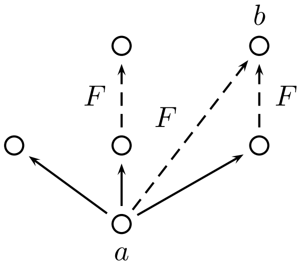


Figure 3: The compacted partially specified signature of S_2

Proposition 1 *If S is a PSS then $compact(S)$ is a well formed PSS.*

6 Module combination

We now describe how to combine modules, an operation we call *merge* bellow. When two modules are combined, nodes that are marked by the same type are coalesced along with their attributes. Nodes that are marked by different types cannot be coalesced and must denote different types. The main complication is caused when two *anonymous* nodes are considered: such nodes are coalesced only if they are indistinguishable.

The merge of two modules is performed in several stages: First, the two graphs are unioned (this is a simple pointwise union of the coordinates of the graph, see definition 7). Then the resulting graph is compacted, coalescing nodes marked by the same type as well as indistinguishable anonymous nodes. However, the resulting graph does not necessarily maintain the relaxed upward closure and maximality conditions, and therefore some modifications are needed. This is done by *Ap-Closure*, see definition 8. Finally, the addition of appropriateness arcs may turn two anonymous distinguishable nodes into indistinguishable ones and therefore another compactness operation is needed (definition 9).

Definition 7 *Let $S_1 = \langle Q_1, T_1, \preceq_1, Ap_1 \rangle$, $S_2 = \langle Q_2, T_2, \preceq_2, Ap_2 \rangle$ be two PLGssuch that $Q_1 \cap Q_2 = \emptyset$. The* **union** *of S_1 and S_2, denoted $S_1 \cup S_2$, is the PLG $S = \langle Q_1 \cup Q_2, T_1 \cup T_2, \preceq_1 \cup \preceq_2, Ap_1 \cup Ap_2 \rangle$.*

Definition 8 *Let $S = \langle Q, T, \preceq, Ap \rangle$ be a PLG. The* **Ap-Closure** *of S, denoted $ApCl(S)$, is the PLG $\langle Q, T, \preceq, Ap'' \rangle$ where:*

- *$Ap' = \{(q_1, F, q_2) \mid q_1, q_2 \in Q$ and there exists $q'_1 \in Q$ such that $q'_1 \overset{*}{\preceq} q_1$ and $(q'_1, F, q_2) \in Ap\}$*
- *$Ap'' = \{(q_1, F, q_2) \in Ap' \mid$ for all $q'_2 \in Q$, such that $q_2 \overset{*}{\preceq} q'_2$ and $q_2 \neq q'_2, (q_1, F, q'_2) \notin Ap'\}$*

Ap-Closure adds to a PLG the arcs required for it to maintain the relaxed upward closure and maximality conditions. First, arcs are added (Ap') to maintain upward closure (to create the relations between elements separated between the two modules and related by mutual elements). Then, redundant arcs are removed to maintain the maximality condition (the removed arcs may be added by Ap' but may also exist in Ap). Notice that $Ap \subseteq Ap'$ since for all $(q_1, F, q_2) \in Ap$, by choosing $q'_1 = q_1$ it follows that $q'_1 = q_1 \overset{*}{\preceq} q_1$ and $(q'_1, F, q_2) = (q_1, F, q_2) \in Ap$ and hence $(q'_1, F, q_2) = (q_1, F, q_2) \in Ap'$.

Two PSSs can be merged only if the resulting subsumption relation is indeed a partial order, where the only obstacle can be the antisymmetry of the resulting relation. The combination of the appropriateness relations, in contrast, cannot cause the merge operation to fail because any violation of the appropriateness conditions in PSSs can be deterministically resolved.

Definition 9 *Let $S_1 = \langle Q_1, T_1, \preceq_1, Ap_1 \rangle$, $S_2 = \langle Q_2, T_2, \preceq_2, Ap_2 \rangle$ be two PSSs such that $Q_1 \cap Q_2 = \emptyset$. S_1, S_2 are* ***mergeable*** *if there are no $q_1, q_2 \in Q_1$ and $q_3, q_4 \in Q_2$ such that the following hold:*

1. *$T_1(q_1)\downarrow$, $T_1(q_2)\downarrow$, $T_2(q_3)\downarrow$ and $T_2(q_4)\downarrow$*
2. *$T_1(q_1) = T_2(q_4)$ and $T_1(q_2) = T_2(q_3)$*
3. *$q_1 \overset{*}{\preceq}_1 q_2$ and $q_3 \overset{*}{\preceq}_2 q_4$*

If S_1 and S_2 are mergeable, then their ***merge****, denoted $S_1 \Cup S_2$, is $compact(ApCl(compact(S_1 \cup S_2)))$.*

In the merged module, pairs of nodes marked by the same type and pairs of indistinguishable anonymous nodes are coalesced. An anonymous node cannot be coalesced with a typed node, even if they are otherwise indistinguishable, since that will result in an unassociative combination operation. Anonymous nodes are assigned types only after all modules combine, see section 7.1.

If a node has multiple outgoing Ap-arcs labeled with the same feature, these arcs are not replaced by a single arc, even if the *lub* of the target nodes exists in the resulting PSS. Again, this is done to guarantee the associativity of the merge operation.

Example 3 *Figure 4 depicts a naïve agreement module, S_5. Combined with S_1 of Figure 1, $S_1 \Cup S_5 = S_5 \Cup S_1 = S_6$, where S_6 is depicted in Figure 5. All dashed arrows are labeled* AGR, *but these labels are suppressed for readability.*

Example 4 *Let S_7 and S_8 be the PSSs depicted in Figures 6 and 7, respectively. Then $S_7 \Cup S_8 = S_8 \Cup S_7 = S_9$, where S_9 is depicted in Figure 8. By standard convention, Ap arcs that can be inferred by upward closure are not depicted.*

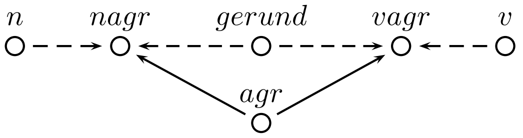


Figure 4: Naïve agreement module, S_5

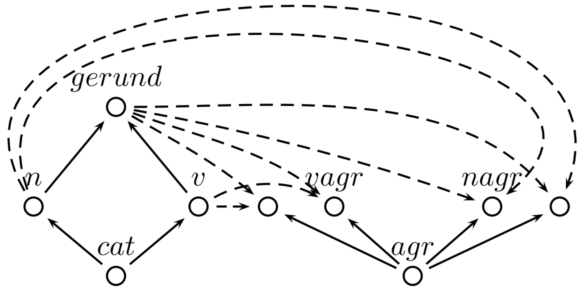


Figure 5: $S_6 = S_1 \uplus S_5$

Proposition 2 *Given two mergeable PSSs S_1, S_2, $S_1 \uplus S_2$ is a well formed PSS.*

Proposition 3 *PSS merge is commutative: for any two PSSs, S_1, S_2, $S_1 \uplus S_2 = S_2 \uplus S_1$. In particular, either both are defined or both are undefined.*

Proposition 4 *PSS merge is associative: for all S_1, S_2, S_3, $(S_1 \uplus S_2) \uplus S_3 = S_1 \uplus (S_2 \uplus S_3)$.*

7 Extending PSSs to type signatures

When developing large scale grammars, the signature can be distributed among several modules. A PSS encodes only partial information and therefore is not required to conform with all the constraints imposed on ordinary signatures. After all the modules are combined, however, the PSS must be extended into a signature. This process is done in 4 stages, each dealing with one property: 1. Name resolution: assigning types to anonymous nodes (section 7.1); 2. Determinizing *Ap*, converting it from a relation to a function (section 7.2); 3. Extending '$\preceq$' to a BCPO. This is done using the algorithm of Penn (2000); 4. Extending *Ap* to a full appropriateness specification by enforcing the feature introduction condition: Again, we use the algorithm of Penn (2000).

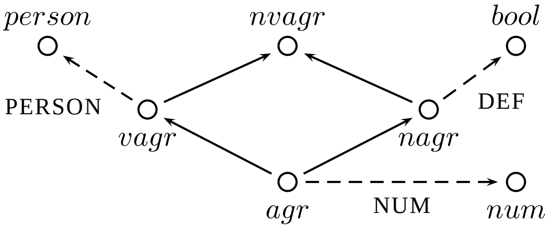


Figure 6: An agreement module, S_7

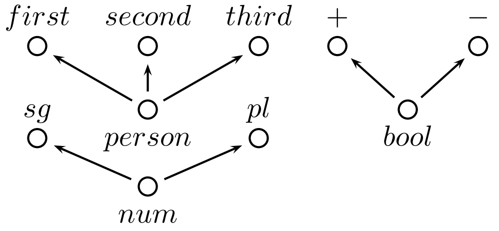


Figure 7: A partially specified signature, S_8

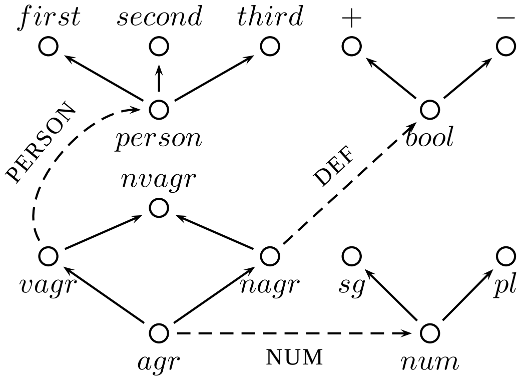


Figure 8: $S_9 = S_7 \uplus S_8$

7.1 Name resolution

By the definition of a well-formed PSS, each anonymous node is unique with respect to the information it encodes among the anonymous nodes, but there may exist a *marked* node encoding the same information. The goal of the name resolution procedure is to assign a type to every anonymous node, by coalescing it with a similar marked node, if one exists. If no such node exists, or if there is more than one such node, the anonymous node is given an arbitrary type.

The name resolution algorithm iterates as long as there are nodes to coalesce. In each iteration, for each anonymous node the set of its similar typed nodes is computed. Then, using this computation, anonymous nodes are coalesced with their paired similar typed node, if such a node uniquely exists. After coalescing all such pairs, the resulting PSS may be non well-formed and therefore the PSS is compacted. Compactness can trigger more pairs that need to be coalesced, and therefore the above procedure is repeated. When no pairs that need to be coalesced are left, the remaining anonymous nodes are assigned arbitrary names and the algorithm halts. The detailed algorithm is suppressed for lack of space.

Example 5 *Let S_6 be the PSS depicted in Figure 5. Executing the name resolution algorithm on this module results in the PSS of Figure 9 (*AGR*-labels are suppressed for readability.) The two anonymous nodes in S_6 are coalesced with the nodes marked $nagr$ and $vagr$, as per their attributes. Cf. Figure 1, in particular how two anonymous nodes in S_1 are assigned types from S_5 (Figure 4).*

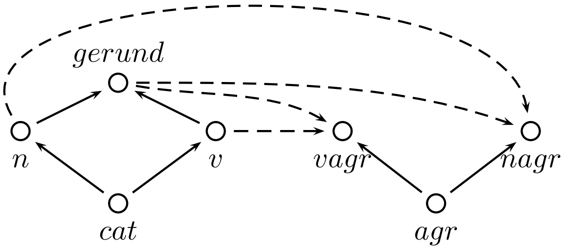


Figure 9: Name resolution result for S_6

7.2 Appropriateness consolidation

For each node q, the set of outgoing appropriateness arcs with the same label F, $\{(q,F,q')\}$, is replaced by the single arc (q,F,q_l), where q_l is marked by the *lub* of the types of all q'. If no *lub* exists, a new node is added and is marked by the *lub*. The result is that the appropriateness relation is a function, and upward closure is preserved; feature introduction is dealt with separately.

The input to the following procedure is a PSS whose typing function, T, is total; its output is a PSS whose typing function, T, is total and whose appropriateness relation is a function. Let $S = \langle Q, T, \preceq, Ap\rangle$ be a PSS. For each $q \in Q$ and $F \in$ FEAT, let

$$\begin{aligned}&target(q,F) = \{q' \mid (q,F,q') \in Ap\}\\&sup(q) = \{q' \in Q \mid q' \preceq q\}\\&sub(q) = \{q' \in Q \mid q \preceq q'\}\\&out(q) = \{(F,q') \mid (q,F,q') \in Ap\end{aligned}$$

Algorithm 1 Appropriateness consolidation ($S = \langle Q, T, \preceq, Ap\rangle$)

```
1. Find a node $q$ and a feature $F$ for which $|target(q,F)| > 1$ and for all $q' \in Q$ such that $q' \overset{*}{\preceq} q$, $|target(q',F)| \leq 1$. If no such pair exists, halt.
2. If $target(q,F)$ has a lub, $p$, then:
   (a) for all $q' \in target(q,F)$, remove the arc $(q,F,q')$ from $Ap$.
   (b) add the arc $(q,F,p)$ to $Ap$.
   (c) for all $q' \in Q$ such that $q \overset{*}{\preceq} q'$, if $(q',F,p) \notin Ap$ then add $(q',F,p)$ to $Ap$.
   (d) go to (1).
3. (a) Add a new node, $p$, to $Q$ with:
       • $sup(p) = target(q,F)$
       • $sub(p) = (target(q,F))^u$
       • $out(p) = \bigcup_{q' \in target(q,F)} out(q')$
   (b) Mark $p$ with a fresh type from NAMES.
   (c) For all $q' \in Q$ such that $q \overset{*}{\preceq} q'$, add $(q',F,p)$ to $Ap$.
   (d) For all $q' \in target(q,F)$, remove the arc $(q,F,q')$ from $Ap$.
   (e) Add $(q,F,p)$ to $Ap$.
   (f) go to (1).
```

The order in which nodes are selected in step 1 of the algorithm is from supertypes to subtypes. This is done to preserve upward closure. In addition, when replacing a set of outgoing appropriateness arcs with the same label F, $\{(q,F,q')\}$, by a single arc (q,F,q_l), q_l is added as an appropriate value for F and all the subtypes of q. Again, this is done to preserve upward closure. If a new node is added (stage 3), then its appropriate features and values are inherited from its immediate supertypes. During the iterations of the algorithm, condition 3b (maximality) of the definition of a PSS may be violated but the resulting graph is guaranteed to be a PSS.

Example 6 *Consider the PSS depicted in Figure 9. Executing the appropriateness consolidation algorithm on this module results in the module depicted in Figure 10.* AGR*-labels are suppressed.*

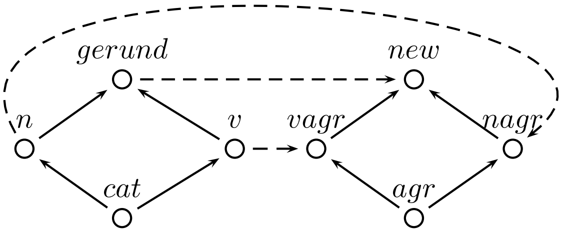


Figure 10: Appropriateness consolidation result

8 Conclusions

We advocate the use of PSSs as the correct concept of signature modules, supporting interaction among grammar modules. Unlike existing approaches, our solution is formally defined, mathematically proven and can be easily and efficiently implemented. Module combination is a commutative and associative operation which meets all the desiderata listed in section 4.

There is an obvious trade-off between flexibility and strong typedeness, and our definitions can be finely tuned to fit various points along this spectrum. In this paper we prefer flexibility, following Melnik (2005), but future work will investigate other options.

There are various other directions for future research. First, grammar *rules* can be distributed among modules in addition to the signature. The definition of modules can then be extended to include also parts of the grammar. Then, various combination operators can be defined for grammar modules (cf. Wintner (2002)). We are actively pursuing this line of research.

Finally, while this work is mainly theoretical, it has important practical implications. We would like to integrate our solutions in an existing environment for grammar development. An environment that supports modular construction of large scale grammars will greatly contribute to grammar development and will have a significant impact on practical implementations of grammatical formalisms.

9 Acknowledgments

We are grateful to Gerald Penn and Nissim Francez for their comments on an earlier version of this paper. This research was supported by The Israel Science Foundation (grant no. 136/01).

References

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2002. The grammar matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In *Proceedings of ACL Workshop on Grammar Engineering. Taipei, Taiwan*, pages 8–14.

Emily M. Bender, Dan Flickinger, Fredrik Fouvry, and Melanie Siegel. 2005. Shared representation in multilingual grammar engineering. *Research on Language and Computation*, 3:131–138.

Marie-Hélène Candito. 1996. A principle-based hierarchical representation of LTAGs. In *COLING-96*, pages 194–199, Copenhagen, Denemark.

Bob Carpenter. 1992. *The Logic of Typed Feature Structures*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press.

Ann Copestake and Dan Flickinger. 2000. An open-source grammar development environment and broad-coverage English grammar using HPSG. In *Proceedings of LREC*, Athens, Greece.

Ann Copestake. 2002. *Implementing typed feature structures grammars*. CSLI publications, Stanford.

Benoit Crabbé and Denys Duchier. 2004. Metagrammar redux. In *CSLP*, Copenhagen, Denemark.

Mary Dalrymple. 2001. *Lexical Functional Grammar*, volume 34 of *Syntax and Semantics*. Academic Press.

Erhard W. Hinrichs, W. Detmar Meurers, and Shuly Wintner. 2004. Linguistic theory and grammar implementation. *Research on Language and Computation*, 2:155–163.

Ronald M. Kaplan, Tracy Holloway King, and John T. Maxwell. 2002. Adapting existing grammars: the XLE experience. In *COLING-02 workshop on Grammar engineering and evaluation*, pages 1–7, Morristown, NJ, USA.

Vlado Keselj. 2001. Modular HPSG. Technical Report CS-2001-05, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada.

Tracy Holloway King, Martin Forst, Jonas Kuhn, and Miriam Butt. 2005. The feature space in parallel grammar writing. *Research on Language and Computation*, 3:139–163.

Nurit Melnik. 2005. From "hand-written" to implemented HPSG theories. In *Proceedings of HPSG-2005, Lisbon, Portugal*.

Nurit Melnik. 2006. A constructional approach to verb-initial constructions in Modern Hebrew. *Cognitive Linguistics*, 17(2). To appear.

Andrew M. Moshier. 1997. Is HPSG featureless or unprincipled? *Linguistics and Philosophy*, 20(6):669–695.

Stephan Oepen, Daniel Flickinger, J. Tsujii, and Hans Uszkoreit, editors. 2002. *Collaborative Language Engineering: A Case Study in Efficient Grammar-Based Processing*. CSLI Publications, Stanford.

Gerald B. Penn. 2000. *The algebraic structure of attributed type signatures*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press and CSLI Publications.

Shuly Wintner. 2002. Modular context-free grammars. *Grammars*, 5(1):41–63.

Morphology-Syntax Interface for Turkish LFG

Özlem Çetinoğlu
Faculty of Engineering and Natural Sciences
Sabancı University
34956, Istanbul, Turkey
ozlemc@su.sabanciuniv.edu

Kemal Oflazer
Faculty of Engineering and Natural Sciences
Sabancı University
34956, Istanbul, Turkey
oflazer@sabanciuniv.edu

Abstract

This paper investigates the use of sublexical units as a solution to handling the complex morphology with productive derivational processes, in the development of a lexical functional grammar for Turkish. Such sublexical units make it possible to expose the internal structure of words with multiple derivations to the grammar rules in a uniform manner. This in turn leads to more succinct and manageable rules. Further, the semantics of the derivations can also be systematically reflected in a compositional way by constructing PRED values on the fly. We illustrate how we use sublexical units for handling simple productive derivational morphology and more interesting cases such as causativization, etc., which change verb valency. Our priority is to handle several linguistic phenomena in order to observe the effects of our approach on both the c-structure and the f-structure representation, and grammar writing, leaving the coverage and evaluation issues aside for the moment.

1 Introduction

This paper presents highlights of a large scale lexical functional grammar for Turkish that is being developed in the context of the ParGram project[1] In order to incorporate in a manageable way, the complex morphology and the syntactic relations mediated by morphological units, and to handle lexical representations of very productive derivations, we have opted to develop the grammar using sublexical units called *inflectional groups*.

Inflectional groups (IGs hereafter) represent the inflectional properties of segments of a complex word structure separated by derivational boundaries. An IG is typically larger than a morpheme but smaller than a word (except when the word has no derivational morphology in which case the IG corresponds to the word). It turns out that it is the IGs that actually define syntactic relations between words. A grammar for Turkish that is based on words as units would have to refer to information encoded at arbitrary positions in words, making the task of the grammar writer much harder. On the other hand, treating morphemes as units in the grammar level implies that the grammar will have to know about morphotactics making either the morphological analyzer redundant, or repeating the information in the morphological analyzer at the grammar level which is not very desirable. IGs bring a certain form of normalization to the lexical representation of a language like Turkish, so that units in which the grammar rules refer to are simple enough to allow easy access to the information encoded in complex word structures.

That IGs delineate productive derivational processes in words necessitates a mechanism that reflects the effect of the derivations to semantic representations and valency changes. For instance, English LFG (Kaplan and Bresnan, 1982) represents derivations as a part of the lexicon; both *happy* and *happiness* are separately lexicalized. Lexicalized representations of adjectives such as *easy* and *easier* are related, so that both lexicalized and phrasal comparatives would have the same feature structure; *easier* would have the feature structure

(1)
$$\left[\begin{array}{ll} \text{PRED} & \text{`easy'} \\ \text{ADJUNCT} & \left[\begin{array}{ll} \text{PRED} & \text{`more'} \end{array}\right] \\ \text{DEG-DIM} & \text{pos} \\ \text{DEGREE} & \text{comparative} \end{array}\right]$$

Encoding derivations in the lexicon could be applicable for languages with relatively unproductive derivational phenomena, but it certainly is not

[1] http://www2.parc.com/istl/groups/nltt/pargram/

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 153–160,
Sydney, July 2006. ©2006 Association for Computational Linguistics

possible to represent in the grammar lexicon,[2] all derived forms as lexemes for an agglutinative language like Turkish. Thus one needs to incorporate such derivational processes in a principled way along with the computation of the effects on derivations on the representation of the semantic information.

Lexical functional grammar (LFG) (Kaplan and Bresnan, 1982) is a theory representing the syntax in two parallel levels: Constituent structures (c-structures) have the form of context-free phrase structure trees. Functional structures (f-structures) are sets of pairs of attributes and values; attributes may be features, such as tense and gender, or functions, such as subject and object. C-structures define the syntactic representation and f-structures define more semantic representation. Therefore c-structures are more language specific whereas f-structures of the same phrase for different languages are expected to be similar to each other.

The remainder of the paper is organized as follows: Section 2 reviews the related work both on Turkish, and on issues similar to those addressed in this paper. Section 3 motivates and presents IGs while Section 4 explains how they are employed in a LFG setting. Section 5 summarizes the architecture and the current status of the our system. Finally we give conclusions in Section 6.

2 Related Work

Güngördü and Oflazer (1995) describes a rather extensive grammar for Turkish using the LFG formalism. Although this grammar had a good coverage and handled phenomena such as free-constituent order, the underlying implementation was based on pseudo-unification. But most crucially, it employed a rather standard approach to represent the lexical units: words with multiple nested derivations were represented with complex nested feature structures where linguistically relevant information could be embedded at unpredictable depths which made access to them in rules extremely complex and unwieldy.

Bozşahin (2002) employed morphemes overtly as lexical units in a CCG framework to account for a variety of linguistic phenomena in a prototype implementation. The drawback was that morphotactics was explicitly raised to the level of the sentence grammar, hence the categorial lexicon accounted for both constituent order and the morpheme order with no distinction. Oflazer's dependency parser (2003) used IGs as units between which dependency relations were established. Another parser based on IGs is Eryiğit and Oflazer's (2006) statistical dependency parser for Turkish. Çakıcı (2005), used relations between IG-based representations encoded within the Turkish Treebank (Oflazer et al., 2003) to automatically induce a CCG grammar lexicon for Turkish.

In a more general setting, Butt and King (2005) have handled the morphological causative in Urdu as a separate node in c-structure rules using LFG's restriction operator in semantic construction of causatives. Their approach is quite similar to ours yet differs in an important way: the rules explicitly use morphemes as constituents so it is not clear if this is just for this case, or all morphology is handled at the syntax level.

3 Inflectional Groups as Sublexical Units

Turkish is an agglutinative language where a sequence of inflectional and derivational morphemes get affixed to a root (Oflazer, 1994). At the syntax level, the unmarked constituent order is SOV, but constituent order may vary freely as demanded by the discourse context. Essentially all constituent orders are possible, especially at the main sentence level, with very minimal formal constraints. In written text however, the unmarked order is dominant at both the main sentence and embedded clause level.

Turkish morphotactics is quite complicated: a given word form may involve multiple derivations and the number of word forms one can generate from a nominal or verbal root is theoretically infinite. Turkish words found in typical text average about 3-4 morphemes including the stem, with an average of about 1.23 derivations per word, but given that certain noninflecting function words such as conjuctions, determiners, etc. are rather frequent, this number is rather close to 2 for inflecting word classes. Statistics from the Turkish Treebank indicate that for sentences ranging between 2 words to 40 words (with an average of about 8 words), the number of IGs range from 2 to 55 IGs (with an average of 10 IGs per sentence) (Eryiğit and Oflazer, 2006).

The morphological analysis of a word can be represented as a sequence of tags corresponding to the morphemes. In our morphological analyzer output, the tag ^DB denotes derivation boundaries that we also use to define IGs. If we represent the morphological information in Turkish in the following general form:

$$\texttt{root+IG}_1 + \texttt{\^{}DB+IG}_2 + \texttt{\^{}DB+}\cdots + \texttt{\^{}DB+IG}_n.$$

then each $\texttt{IG}_i$ denotes the relevant sequence of inflectional features including the part-of-speech for the root (in IG_1) and for any of the derived forms. A given word may have multiple such representations depending on any morphological ambiguity brought about by alternative segmentations of the

[2] We use this term to distinguish the lexicon used by the morphological analyzer.

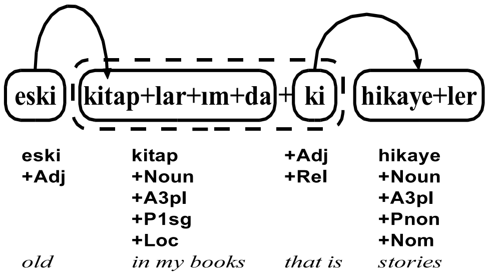


Figure 1: Modifier-head relations in the NP *eski kitaplarımdaki hikayeler*

word, and by ambiguous interpretations of morphemes.

For instance, the morphological analysis of the derived modifier `cezalandırılacak` (literally, "(the one) that will be given punishment") would be :[3]

```
ceza(punishment)+Noun+A3sg+Pnon+Nom
         ^DB+Verb+Acquire
         ^DB+Verb+Caus
         ^DB+Verb+Pass+Pos
         ^DB+Adj+FutPart+Pnon
```

The five IGs in this word are:

1. `+Noun+A3sg+Pnon+Nom`
2. `+Verb+Acquire`
3. `+Verb+Caus`
4. `+Verb+Pass+Pos`
5. `+Adj+FutPart+Pnon`

The first IG indicates that the root is a singular noun with nominative case marker and no possessive marker. The second IG indicates a derivation into a verb whose semantics is "to acquire" the preceding noun. The third IG indicates that a causative verb (equivalent to "to punish" in English), is derived from the previous verb. The fourth IG indicates the derivation of a passive verb with positive polarity from the previous verb. Finally the last IG represents a derivation into future participle which will function as a modifier in the sentence.

The simple phrase *eski kitaplarımdaki hikayeler* (the stories in my old books) in Figure 1 will help clarify how IGs are involved in syntactic relations: Here, *eski* (old) modifies *kitap* (book) and not *hikayeler* (stories),[4] and the locative phrase *eski kitaplarımda* (in my old books) modifies *hikayeler* with the help of derivational suffix *-ki*. Morpheme boundaries are represented by '+' sign and morphemes in solid boxes actually define one IG. The dashed box around solid boxes is for word boundary. As the example indicates, IGs may consist of one or more morphemes.

Example (2) shows the corresponding f-structure for this NP. Supporting the dependency representation in Figure 1, f-structure of adjective *eski* is placed as the adjunct of *kitaplarımda*, at the innermost level. The semantics of the relative suffix *-ki* is shown as 'rel⟨↑ OBJ⟩' where the f-structure that represents the NP *eski kitaplarımda* is the OBJ of the derived adjective. The new f-structure with a PRED constructed on the fly, then modifies the noun *hikayeler*. The derived adjective behaves essentially like a lexical adjective. The effect of using IGs as the representative units can be explicitly seen in c-structure where each IG corresponds to a separate node as in Example (3).[5] Here, DS stands for derivational suffix.

(2)

$$\left[\begin{array}{lll} \text{PRED} & \text{`hikaye'} & \\ \text{ADJUNCT} & \left[\begin{array}{ll} \text{PRED} & \text{`rel}\langle\text{kitap}\rangle\text{'} \\ \text{OBJ} & \left[\begin{array}{ll} \text{PRED} & \text{`kitap'} \\ \text{ADJUNCT} & \left[\begin{array}{ll} \text{PRED} & \text{`eski'} \\ \text{ATYPE} & \text{attributive} \end{array}\right] \\ \text{CASE loc, NUM pl} & \end{array}\right] \\ \text{ATYPE} & \text{attributive} \end{array}\right] & \\ \text{CASE NOM, NUM PL} & & \end{array}\right]$$

(3)

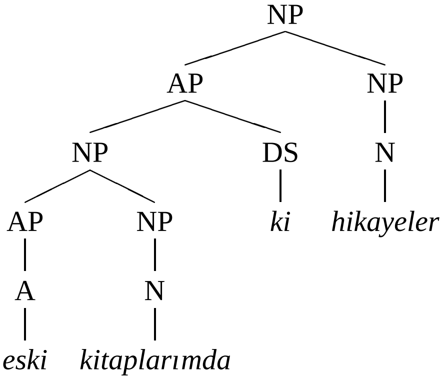


Figure 2 shows the modifier-head relations for a more complex example given in Example (4) where we observe a chain/hierarchy of relations between IGs

(4)

mavi	*renkli*	
blue	color-WITH	
elbiselideki		*kitap*
dress-WITH-LOC-REL		book

[3] The morphological features other than the obvious part-of-speech features are: `+A3sg`: 3sg number-person agreement, `+Pnon`: no possesive agreement, `+Nom`: Nominative case, `+Acquire`: acquire verb, `+Caus`: causative verb, `+Pass`: passive verb, `+FutPart`: Derived future participle, `+Pos`: Positive Polarity.

[4] Though looking at just the last POS of the words one sees an `+Adj +Adj +Noun` sequence which may imply that both adjectives modify the noun *hikayeler*

[5] Note that placing the sublexical units of a word in separate nodes goes against the Lexical Integrity principle of LFG (Dalrymple, 2001). The issue is currently being discussed within the LFG community (T. H. King, personal communication).

‘the book on the one with the blue colored dress’

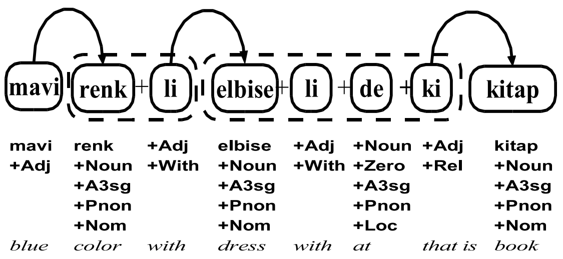

mavi
renk
li
elbise
li
de
ki
kitap
mavi +Adj
renk +Noun +A3sg +Pnon +Nom
+Adj +With
elbise +Noun +A3sg +Pnon +Nom
+Adj +With
+Noun +Zero +A3sg +Pnon +Loc
+Adj +Rel
kitap +Noun +A3sg +Pnon +Nom
blue
color
with
dress
with
at
that is
book

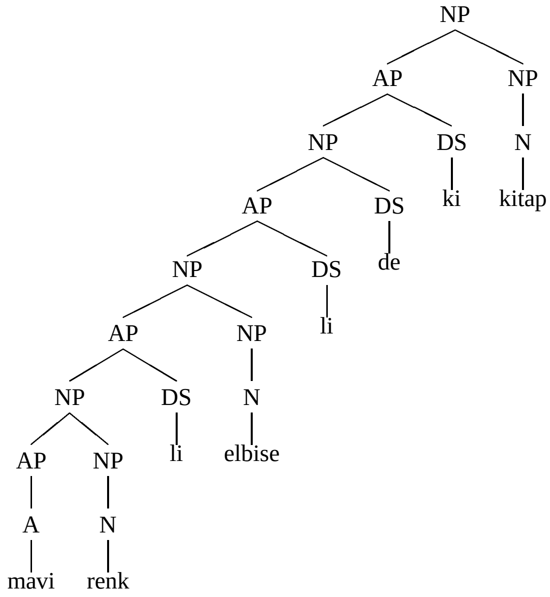
Figure 2: Syntactic Relations in the NP *mavi renkli elbiselideki kitap*

Examples (5) and (6) show respectively the constituent structure (c-structure) and the corresponding feature structure (f-structure) for this noun phrase. Within the tree representation, each IG corresponds to a separate node. Thus, the LFG grammar rules constructing the c-structures are coded using IGs as units of parsing. If an IG contains the root morpheme of a word, then the node corresponding to that IG is named as one of the syntactic category symbols. The rest of the IGs are given the node name DS (to indicate derivational suffix), no matter what the content of the IG is.

The semantic representation of derivational suffixes plays an important role in f-structure construction. In almost all cases, each derivation that is induced by an overt or a covert affix gets a OBJ feature which is then unified with the f-structure of the preceding stem already constructed, to obtain the feature structure of the derived form, with the PRED of the derived form being constructed on the fly. A PRED feature thus constructed however is not meant to necessarily have a precise lexical semantics. Most derivational suffixes have a consistent (lexical) semantics[6], but some don’t, that is, the precise additional lexical semantics that the derivational suffix brings in, depends on the stem it is affixed to. Nevertheless, we represent both cases in the same manner, leaving the determination of the precise lexical semantics aside.

If we consider Figure 2 in terms of dependency relations, the adjective *mavi* (blue) modifies the noun *renk* (color) and then the derivational suffix *-li* (with) kicks in although the *-li* is attached to *renk* only. Therefore, the semantics of the phrase should be `with(blue color)`, not `blue with(color)`. With the approach we take, this difference can easily be represented in both the f-structure as in the leftmost branch in Example (5) and the c-structure as in the middle ADJUNCT f-structure in Example (6). Each DS in c-structure gives rise to an OBJect in c-structure. More precisely, a derived phrase is always represented as a binary tree where the right daughter is always a DS. In f-structure DS unifies with the mother f-structure and inserts PRED feature which subcategorizes for a OBJ. The left daughter of the binary tree is the original form of the phrase that is derived, and it unifies with the OBJ of the mother f-structure.

(5)



4 Inflectional Groups in Practice

We have already seen how the IGs are used to construct on the fly PRED features that reflect the lexical semantics of the derivation. In this section we describe how we handle phenomena where the derivational suffix in question does not explicitly affect the semantic representation in PRED feature but determines the semantic role so as to unify the derived form or its components with the appropriate external f-structure.

4.1 Sentential Complements and Adjuncts, and Relative Clauses

In Turkish, sentential complements and adjuncts are marked by productive verbal derivations into nominals (infinitives, participles) or adverbials, while relative clauses with subject and non-subject (object or adjunct) gaps are formed by participles which function as adjectivals modifying a head noun.

Example (7) shows a simple sentence that will be used in the following examples.

[6] e.g., the “to acquire” example earlier

(6)

$$\left[\begin{array}{ll}
\text{PRED} & \text{`kitap'} \\
\text{ADJUNCT} & \left[\begin{array}{ll}
\text{PRED} & \text{`rel}\langle\text{zero-deriv}\rangle\text{'} \\
\text{OBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{`zero-deriv}\langle\text{with}\rangle\text{'} \\
\text{OBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{`with}\langle\text{elbise}\rangle\text{'} \\
\text{OBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{`elbise'} \\
\textbf{ADJUNCT} & \left[\begin{array}{ll}
\text{PRED} & \text{`with}\langle\text{renk}\rangle\text{'} \\
\text{OBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{`renk'} \\
\text{ADJUNCT} & \left[\text{PRED}\quad\text{`mavi'}\right] \\
\multicolumn{2}{l}{\text{CASE nom, NUM sg, PERS 3}}
\end{array}\right] \\
\multicolumn{2}{l}{\text{ATYPE attributive}}
\end{array}\right] \\
\multicolumn{2}{l}{\text{CASE nom, NUM sg, PERS 3}}
\end{array}\right] \\
\multicolumn{2}{l}{\text{ATYPE attributive}}
\end{array}\right] \\
\multicolumn{2}{l}{\text{CASE loc, NUM sg, PERS 3}}
\end{array}\right] \\
\multicolumn{2}{l}{\text{ATYPE attributive}}
\end{array}\right] \\
\multicolumn{2}{l}{\text{CASE NOM, NUM SG, PERS 3}}
\end{array}\right]$$

(7) *Kız* *adamı* *aradı.*
Girl-NOM man-ACC call-PAST
'The girl called the man'

In (8), we see a past-participle form heading a sentential complement functioning as an object for the verb *söyledi* (said).

(8) *Manav* *kızın* *adamı*
Grocer-NOM girl-GEN man-ACC
aradığını *söyledi.*
call-PASTPART-ACC say-PAST
'The grocer said that the girl called the man'

Once the grammar encounters such a sentential complement, everything up to the participle IG is parsed, as a normal sentence and then the participle IG appends nominal features, e.g., CASE, to the existing f-structure. The final f-structure is for a noun phrase, which now is the object of the matrix verb, as shown in Example (9). Since the participle IG has the right set of syntactic features of a noun, no new rules are needed to incorporate the derived f-structure to the rest of the grammar, that is, the derived phrase can be used as if it is a simple NP within the rules. The same mechanism is used for all kinds of verbal derivations into infinitives, adverbial adjuncts, including those derivations encoded by lexical reduplications identified by multi-word construct processors.

(9)

$$\left[\begin{array}{ll}
\text{PRED} & \text{`söyle}\langle\text{manav, ara}\rangle\text{'} \\
\text{SUBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{`manav'} \\
\multicolumn{2}{l}{\text{CASE nom, NUM sg, PERS 3}}
\end{array}\right] \\
\text{OBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{`ara}\langle\text{kız, adam}\rangle\text{'} \\
\text{SUBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{`kız'} \\
\multicolumn{2}{l}{\text{CASE gen, NUM sg, PERS 3}}
\end{array}\right] \\
\text{OBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{`adam'} \\
\multicolumn{2}{l}{\text{CASE acc, NUM sg, PERS 3}}
\end{array}\right] \\
\text{CHECK} & \left[\text{PART}\quad\text{pastpart}\right] \\
\multicolumn{2}{l}{\text{CASE acc, NUM sg, PERS 3, VTYPE main}} \\
\multicolumn{2}{l}{\text{CLAUSE-TYPE nom}}
\end{array}\right] \\
\text{TNS-ASP} & \left[\text{TENSE}\quad\text{past}\right] \\
\multicolumn{2}{l}{\text{NUM SG, PERS 3, VTYPE MAIN}}
\end{array}\right]$$

Relative clauses also admit to a similar mechanism. Relative clauses in Turkish are gapped sentences which function as modifiers of nominal heads. Turkish relative clauses have been previously studied (Barker et al., 1990; Güngördü and Engdahl, 1998) and found to pose interesting issues for linguistic and computational modeling. Our aim here is not to address this problem in its generality but show with a simple example, how our treatment of IGs encoding derived forms handle the mechanics of generating f-structures for such cases.

Kaplan and Zaenen (1988) have suggested a general approach for handling long distance dependencies. They have extended the LFG notation and allowed regular expressions in place of simple attributes within f-structure constraints so that phenomena requiring infinite disjunctive enumeration can be described with a finite expression. We basically follow this approach and once we derive the participle phrase we unify it with the appropriate argument of the verb using rules based on functional uncertainty. Example (10) shows a relative clause where a participle form is used as a modifier of a head noun, *adam* in this case.

(10) *Manavın* *kızın* $[]_i$
Grocer-GEN girl-GEN obj-gap
aradığını *söylediği* *adam*$_i$
call-PASTPART-ACC say-PASTPART man-NOM
'The man the grocer said the girl called'

This time, the sentence is parsed with a gap with an appropriate functional uncertainty constraint, and when the participle IG is encountered the sentence f-structure is derived into an adjective and the gap in the derived form, the object here, is then unified with the head word as marked with co-indexation in Example (11).

The example sentence (10) includes Example (8) as a relative clause with the object extracted, hence the similarity in the f-structures can be observed easily. The ADJUNCT in Example (11) is almost the same as the whole f-structure of Example (9), differing in TNS-ASP and ADJUNCT-TYPE features. At the grammar level, both the relative clause and the complete sentence is parsed with the same core sentence rule. To understand whether the core sentence is a complete sentence or not, the finite verb requirement is checked. Since the requirement is met by the existence of TENSE feature, Example (8) is parsed as a complete sentence. Indeed the relative clause also includes temporal information as 'pastpart' value of PART feature, of the ADJUNCT f-structure, denoting a past event.

(11)
$$\left[\begin{array}{ll}
\text{PRED} & \text{'adam'}\ \boxed{1}\\
\text{ADJUNCT} & \left[\begin{array}{ll}
\text{PRED} & \text{`söyle}\langle\text{manav, ara}\rangle\text{'}\\
\text{SUBJ} & \left[\begin{array}{l}\text{PRED `manav'}\\ \text{CASE gen, NUM sg, PERS 3}\end{array}\right]\\
\text{OBJ} & \left[\begin{array}{ll}
\text{PRED} & \text{`ara}\langle\text{kız, adam}\rangle\text{'}\\
\text{SUBJ} & \left[\begin{array}{l}\text{PRED `kız'}\\ \text{CASE gen, NUM sg, PERS 3}\end{array}\right]\\
\text{OBJ} & \left[\text{PRED `adam'}\right]\boxed{1}\\
\text{CHECK} & \left[\text{PART pastpart}\right]\\
\multicolumn{2}{l}{\text{CASE acc, NUM sg, PERS 3, VTYPE main}}\\
\multicolumn{2}{l}{\text{CLAUSE-TYPE nom}}
\end{array}\right]\\
\text{CHECK} & \left[\text{PART pastpart}\right]\\
\multicolumn{2}{l}{\text{NUM sg, PERS 3, VTYPE main}}\\
\multicolumn{2}{l}{\text{ADJUNCT-TYPE relative}}
\end{array}\right]\\
\multicolumn{2}{l}{\text{CASE NOM, NUM SG, PERS 3}}
\end{array}\right]$$

4.2 Causatives

Turkish verbal morphotactics allows the production multiply causative forms for verbs.[7] Such verb formations are also treated as verbal derivations and hence define IGs. For instance, the morphological analysis for the verb *aradı* (s/he called) is

`ara+Verb+Pos+Past+A3sg`

and for its causative *arattı* (s/he made (someone else) call) the analysis is

`ara+Verb^DB+Verb+Caus+Pos+Past+A3sg`.

In Example (12) we see a sentence and its causative form followed by respective f-structures for these sentences in Examples (13) and (14). The detailed morphological analyses of the verbs are given to emphasize the morphosyntactic relation between the bare and causatived versions of the verb.

(12) a. *Kız* *adamı* *aradı.*
Girl-NOM man-ACC call-PAST
'The girl called the man'

b. *Manav* *kıza* *adamı* *arattı.*
Grocer-NOM girl-DAT man-ACC call-CAUS-PAST
'The grocer made the girl call the man'

(13)
$$\left[\begin{array}{ll}
\text{PRED} & \text{`ara}\langle\text{kız, adam}\rangle\text{'}\\
\text{SUBJ} & \left[\begin{array}{l}\text{PRED `kız'}\\ \text{CASE nom, NUM sg, PERS 3}\end{array}\right]\\
\text{OBJ} & \left[\begin{array}{l}\text{PRED `adam'}\\ \text{CASE acc, NUM sg, PERS 3}\end{array}\right]\\
\text{TNS-ASP} & \left[\text{TENSE past}\right]\\
\multicolumn{2}{l}{\text{NUM SG, PERS 3,VTYPE MAIN}}
\end{array}\right]$$

(14)
$$\left[\begin{array}{ll}
\text{PRED} & \text{`caus}\langle\text{manav, kız, adam, ara}\langle\text{kız , adam}\rangle\rangle\text{'}\\
\text{SUBJ} & \left[\text{PRED `manav'}\right]\\
\text{OBJ} & \left[\text{PRED `kız'}\right]\boxed{1}\\
\text{OBJTH} & \left[\text{PRED `adam'}\right]\boxed{2}\\
\text{XCOMP} & \left[\begin{array}{ll}
\text{PRED} & \text{`ara}\langle\text{kız , adam}\rangle\text{'}\\
\text{SUBJ} & \left[\begin{array}{l}\text{PRED `kız'}\\ \text{CASE dat, NUM sg, PERS 3}\end{array}\right]\boxed{1}\\
\text{OBJ} & \left[\begin{array}{l}\text{PRED `adam'}\\ \text{CASE acc, NUM sg, PERS 3}\end{array}\right]\boxed{2}\\
\text{VTYPE} & \text{main}
\end{array}\right]\\
\text{TNS-ASP} & \left[\text{TENSE past}\right]\\
\multicolumn{2}{l}{\text{NUM SG, PERS 3,VTYPE MAIN}}
\end{array}\right]$$

The end-result of processing an IG which has a verb with a causative form is to create a larger f-structure whose PRED feature has a SUBJect, an OBJect and a XCOMPlement. The f-structure of the first verb is the complement in the f-structure of the causative form, that is, its whole structure is embedded into the mother f-structure in an encapsulated way. The object of the causative (causee - that who is caused by the causer – the subject of the causative verb) is unified with the subject the inner f-structure. If the original verb is transitive, the object of the original verb is further unified with the OBJTH of the causative verb. All of grammatical functions in the inner f-structure, namely XCOMP, are also represented in the mother f-structure and are placed as arguments of `caus` since the flat representation is required to enable free word order in sentence level.

Though not explicit in the sample f-structures, the important part is unifying the object and former subject with appropriate case markers, since the functions of the phrases in the sentence are decided with the help of case markers due to free word order. If the verb that is causativized subcategorizes for an direct object in accusative case, after causative formation, the new object unified with the subject of the causativized verb should be in dative case (Example 15). But if the verb in question subcategorizes for a dative or an ablative oblique object, then this object will be transformed into a direct object in accusative case after causativization (Example 16). That is, the causativation will select the case of the object of the causative verb, so as not to "interfere" with the object of the verb that is causativized. In causativized intransitive verbs the causative object is always in accusative case.

[7] Passive, reflexive, reciprocal/collective verb formations are also handled in morphology, though the latter two are not productive due to semantic constraints. On the other hand it is possible for a verb to have multiple causative markers, though in practice 2-3 seem to be the maximum observed.

(15) a.

adam	*kadını*	*aradı.*
man-NOM	woman-ACC	call-PAST

'the man called the woman'

b.

adama	*kadını*	*arattı.*
man-DAT	woman-ACC	call-CAUS-PAST

'(s/he) made the man call the woman'

(16) a.

adam	*kadına*	*vurdu.*
man-NOM	woman-DAT	hit-PAST

'the man hit the woman'

b.

adamı	*kadına*	*vurdurdu.*
man-ACC	woman-DAT	hit-CAUS-PAST

'(s/he) made the man hit the woman'

All other derivational phenomena can be solved in a similar way by establishing the appropriate semantic representation for the derived IG and its effect on the semantic representation.

5 Current Implementation

The implementation of the Turkish LFG grammar is based on the Xerox Linguistic Environment (XLE) (Maxwell III and Kaplan, 1996), a grammar development platform that facilitates the integration of various modules, such as tokenizers, finite-state morphological analyzers, and lexicons. We have integrated into XLE, a series of finite state transducers for morphological analysis and for multi-word processing for handling lexicalized, semi-lexicalized collocations and a limited form of non-lexicalized collocations.

The finite state modules provide the relevant ambiguous morphological interpretations for words and their split into IGs, but do not provide syntactically relevant semantic and subcategorization information for root words. Such information is encoded in a lexicon of root words on the grammar side.

The grammar developed so far addresses many important aspects ranging from free constituent order, subject and non-subject extractions, all kinds of subordinate clauses mediated by derivational morphology and has a very wide coverage NP subgrammar. As we have also emphasized earlier, the actual grammar rules are oblivious to the source of the IGs, so that the same rule handles an adjective - noun phrase regardless of whether the adjective is lexical or a derived one. So all such relations in Figure 2[8] are handled with the same phrase structure rule.

The grammar is however lacking the treatment of certain interesting features of Turkish such as *suspended affixation* (Kabak, 2007) in which the inflectional features of the last element in a coordination have a phrasal scope, that is, all other coordinated constituents have certain default features which are then "overridden" by the features of the last element in the coordination. A very simple case of such suspended affixation is exemplified in (17a) and (17b). Note that although this is not due to derivational morphology that we have emphasized in the previous examples, it is due to a more general nature of morphology in which affixes may have phrasal scopes.

(17) a.

kız	*adam*	*ve*	*kadını*	*aradı.*
girl	man-NOM	and	woman-ACC	call-PAST

'the girl called the man and the woman'

b.

kız	*[adam*	*ve*	*kadın]-ı*	*aradı.*
girl	[man	and	woman]-ACC	call-PAST

'the girl called the man and the woman'

Suspended affixation is an example of a phenomenon that IGs do not seem directly suitable for. The unification of the coordinated IGs have to be done in a way in which non-default features of the final constituent is percolated to the upper node in the tree as is usually done with phrase structure grammars but unlike coordination is handled in such grammars.

6 Conclusions and Future Work

This paper has described the highlights of our work on developing a LFG grammar for Turkish employing sublexical constituents, that we have called inflectional groups. Such a sublexical constituent choice has enabled us to handle the very productive derivational morphology in Turkish in a rather principled way and has made the grammar more or less oblivious to morphological complexity.

Our current and future work involves extending the coverage of the grammar and lexicon as we have so far included in the grammar lexicon only a small subset of the root lexicon of the morphological analyzer, annotated with the semantic and subcategorization features relevant to the linguistic phenomena that we have handled. We also intend to use the Turkish Treebank (Oflazer et al., 2003), as a resource to extract statistical information along the lines of Frank et al. (2003) and O'Donovan et al. (2005).

Acknowledgement

This work is supported by TUBITAK (The Scientific and Technical Research Council of Turkey) by grant 105E021.

[8] Except the last one which requires some additional treatment with respect to definiteness.

References

Chris Barker, Jorge Hankamer, and John Moore, 1990. *Grammatical Relations*, chapter Wa and Ga in Turkish. CSLI.

Cem Bozşahin. 2002. The combinatory morphemic lexicon. *Computational Linguistics*, 28(2):145–186.

Miriam Butt and Tracey Holloway King. 2005. Restriction for morphological valency alternations: The Urdu causative. In *Proceedings of The 10th International LFG Conference*, Bergen, Norway. CSLI Publications.

Ruken Çakıcı. 2005. Automatic induction of a CCG grammar for Turkish. In *Proceedings of the ACL Student Research Workshop*, pages 73–78, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Mary Dalrymple. 2001. *Lexical Functional Grammar*, volume 34 of *Syntax and Semantics*. Academic Press, New York.

Gülşen Eryiğit and Kemal Oflazer. 2006. Statistical dependency parsing for turkish. In *Proceedings of EACL 2006 - The 11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

Anette Frank, Louisa Sadler, Josef van Genabith, and Andy Way. 2003. From treebank resources to LFG f-structures:automatic f-structure annotation of treebank trees and CFGs extracted from treebanks. In Anne Abeille, editor, *Treebanks*. Kluwer Academic Publishers, Dordrecht.

Zelal Güngördü and Elisabeth Engdahl. 1998. A relational approach to relativization in Turkish. In *Joint Conference on Formal Grammar, HPSG and Categorial Grammar*, Saarbrücken, Germany, August.

Zelal Güngördü and Kemal Oflazer. 1995. Parsing Turkish using the Lexical Functional Grammar formalism. *Machine Translation*, 10(4):515–544.

Barış Kabak. 2007. Turkish suspended affixation. *Linguistics*, 45. (to appear).

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. MIT Press, Cambridge, MA.

Ronald M. Kaplan and Annie Zaenen. 1988. Long-distance dependencies, constituent structure, and functional uncertainty. In M. Baitin and A. Kroch, editors, *Alternative Conceptions of Phrase Structure*. University of Chicago Press, Chicago.

John T. Maxwell III and Ronald M. Kaplan. 1996. An efficient parser for LFG. In Miriam Butt and Tracy Holloway King, editors, *The Proceedings of the LFG ’96 Conference*, Rank Xerox, Grenoble.

Ruth O’Donovan, Michael Burke, Aoife Cahill, Josef van Genabith, and Andy Way. 2005. Large-scale induction and evaluation of lexical resources from the Penn-II and Penn-III Treebanks. *Computational Linguistics*, 31(3):329–365.

Kemal Oflazer, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. 2003. Building a Turkish treebank. In Anne Abeille, editor, *Building and Exploiting Syntactically-annotated Corpora*. Kluwer Academic Publishers.

Kemal Oflazer. 1994. Two-level description of Turkish morphology. *Literary and Linguistic Computing*, 9(2):137–148.

Kemal Oflazer. 2003. Dependency parsing with an extended finite-state approach. *Computational Linguistics*, 29(4):515–544.

PCFGs with Syntactic and Prosodic Indicators of Speech Repairs

John Hale[a] Izhak Shafran[b] Lisa Yung[c]
Bonnie Dorr[d] Mary Harper[de] Anna Krasnyanskaya[f] Matthew Lease[g]
Yang Liu[h] Brian Roark[i] Matthew Snover[d] Robin Stewart[j]

[a] Michigan State University; [b,c] Johns Hopkins University; [d] University of Maryland, College Park; [e] Purdue University
[f] UCLA; [g] Brown University; [h] University of Texas at Dallas; [i] Oregon Health & Sciences University; [j] Williams College

Abstract

A grammatical method of combining two kinds of speech repair cues is presented. One cue, prosodic disjuncture, is detected by a decision tree-based ensemble classifier that uses acoustic cues to identify where normal prosody seems to be interrupted (Lickley, 1996). The other cue, syntactic parallelism, codifies the expectation that repairs continue a syntactic category that was left unfinished in the reparandum (Levelt, 1983). The two cues are combined in a Treebank PCFG whose states are split using a few simple tree transformations. Parsing performance on the Switchboard and Fisher corpora suggests that these two cues help to locate speech repairs in a synergistic way.

1 Introduction

Speech repairs, as in example (1), are one kind of disfluent element that complicates any sort of syntax-sensitive processing of conversational speech.

(1) and [the first kind of invasion of] the first type of privacy seemed invaded to me

The problem is that the bracketed *reparandum* region (following the terminology of Shriberg (1994)) is approximately repeated as the speaker "repairs" what he or she has already uttered. This extra material renders the entire utterance ungrammatical—the string would not be generated by a correct grammar of fluent English. In particular, attractive tools for natural language understanding systems, such as Treebank grammars for written corpora, naturally lack appropriate rules for analyzing these constructions.

One possible response to this mismatch between grammatical resources and the brute facts of disfluent speech is to make one look more like the other, for the purpose of parsing. In this separate-processing approach, reparanda are located through a variety of acoustic, lexical or string-based techniques, then excised before submission to a parser (Stolcke and Shriberg, 1996; Heeman and Allen, 1999; Spilker et al., 2000; Johnson and Charniak, 2004). The resulting parse tree then has the reparandum re-attached in a standardized way (Charniak and Johnson, 2001).

An alternative strategy, adopted in this paper, is to use the same grammar to model fluent speech, disfluent speech, and their interleaving.

Such an integrated approach can use syntactic properties of the reparandum itself. For instance, in example (1) the reparandum is an *unfinished* noun phrase, the repair a *finished* noun phrase. This sort of phrasal correspondence, while not absolute, is strong in conversational speech, and cannot be exploited on the separate-processing approach. Section 3 applies metarules (Weischedel and Sondheimer, 1983; McKelvie, 1998a; Core and Schubert, 1999) in recognizing these correspondences using standard context-free grammars.

At the same time as it defies parsing, conversational speech offers the possibility of leveraging prosodic cues to speech repairs. Sec-

The authors are very grateful for Eugene Charniak's help adapting his parser. We also thank the Center for Language and Speech processing at Johns Hopkins for hosting the summer workshop where much of this work was done. This material is based upon work supported by the National Science Foundation (NSF) under Grant No. 0121285. 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 NSF.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 161–168,
Sydney, July 2006. ©2006 Association for Computational Linguistics

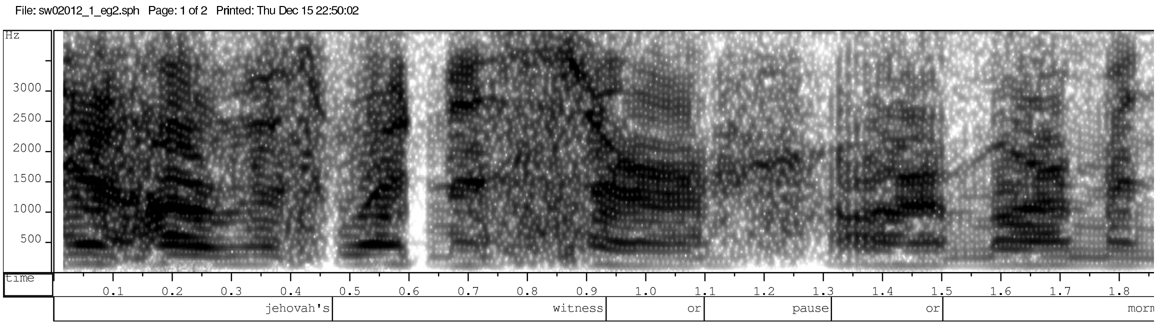


Figure 1: The pause between two *or* s and the glottalization at the end of the first makes it easy for a listener to identify the repair.

tion 2 describes a classifier that learns to label prosodic breaks suggesting upcoming disfluency. These marks can be propagated up into parse trees and used in a probabilistic context-free grammar (PCFG) whose states are systematically split to encode the additional information.

Section 4 reports results on Switchboard (Godfrey et al., 1992) and Fisher EARS RT04F data, suggesting these two features can bring about independent improvements in speech repair detection. Section 5 suggests underlying linguistic and statistical reasons for these improvements. Section 6 compares the proposed grammatical method to other related work, including state of the art separate-processing approaches. Section 7 concludes by indicating a way that string- and tree-based approaches to reparandum identification could be combined.

2 Prosodic disjuncture

Everyday experience as well as acoustic analysis suggests that the syntactic interruption in speech repairs is typically accompanied by a change in prosody (Nakatani and Hirschberg, 1994; Shriberg, 1994). For instance, the spectrogram corresponding to example (2), shown in Figure 1,

(2) the jehovah's witness or [or] mormons or someone

reveals a noticeable pause between the occurrence of the two *or*s, and an unexpected glottalization at the end of the first one. Both kinds of cues have been advanced as explanations for human listeners' ability to identify the reparandum even before the repair occurs.

Retaining only the second explanation, Lickley (1996) proposes that there is no "edit signal" per se but that repair is cued by the absence of smooth formant transitions and lack of normal juncture phenomena.

One way to capture this notion in the syntax is to enhance the input with a special disjuncture symbol. This symbol can then be propagated in the grammar, as illustrated in Figure 2. This work uses a suffix ˜+ to encode the perception of abnormal prosody after a word, along with phrasal -BRK tags to decorate the path upwards to reparandum constituents labeled EDITED. Such disjuncture symbols are identified in the ToBI labeling scheme as break indices (Price et al., 1991; Silverman et al., 1992).

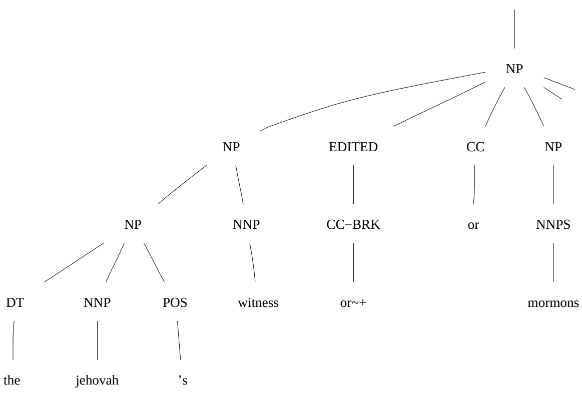


Figure 2: Propagating *BRK*, the evidence of disfluent juncture, from acoustics to syntax.

The availability of a corpus annotated with ToBI labels makes it possible to design a break index classifier via supervised training. The corpus is a subset of the Switchboard corpus, consisting of sixty-four telephone conversations manually annotated by an experienced linguist according to a simplified ToBI labeling scheme (Ostendorf et al., 2001). In ToBI, degree of disjuncture is indicated by integer values from 0 to 4, where a value of 0 corresponds to clitic and 4 to a major phrase break. In addition, a suffix p denotes perceptually disfluent events reflecting, for example, hesitation or planning. In conversational speech the intermediate levels occur infrequently and the break indices can be broadly categorized into three groups, namely, 1, 4 and p as in Wong et al. (2005).

A classifier was developed to predict three break indices at each word boundary based on variations in pitch, duration and energy associated with word, syllable or sub-syllabic constituents (Shriberg et al., 2005; Sonmez et al., 1998). To compute these features, phone-level time-alignments were obtained from an automatic speech recognition system. The duration of these phonological constituents were derived from the ASR alignment, while energy and pitch were computed every 10ms with *snack*, a public-domain sound toolkit (Sjlander, 2001). The duration, energy, and pitch were post-processed according to stylization procedures outlined in Sonmez et al. (1998) and normalized to account for variability across speakers.

Since the input vector can have missing values such as the absence of pitch during unvoiced sound, only decision tree based classifiers were investigated. Decision trees can handle missing features gracefully. By choosing different combinations of splitting and stopping criteria, an ensemble of decision trees was built using the publicly-available IND package (Buntine, 1992). These individual classifiers were then combined into ensemble-based classifiers.

Several classifiers were investigated for detecting break indices. On ten-fold cross-validation, a bagging-based classifier (Breiman, 1996) predicted prosodic breaks with an accuracy of 83.12% while chance was 67.66%. This compares favorably with the performance of the supervised classifiers on a similar task in Wong et al. (2005). Random forests and hidden Markov models provide marginal improvements at considerable computational cost (Harper et al., 2005).

For speech repair, the focus is on detecting disfluent breaks. The precision and recall trade-off on its detection can be adjusted using a threshold on the posterior probability of predicting "p", as shown in Figure 3.

In essence, the large number of acoustic and prosodic features related to disfluency are encoded via the ToBI label 'p', and provided as additional observations to the PCFG. This is unlike previous work on incorporating prosodic information (Gregory et al., 2004; Lease et al., 2005; Kahn et al., 2005) as described further in Section 6.

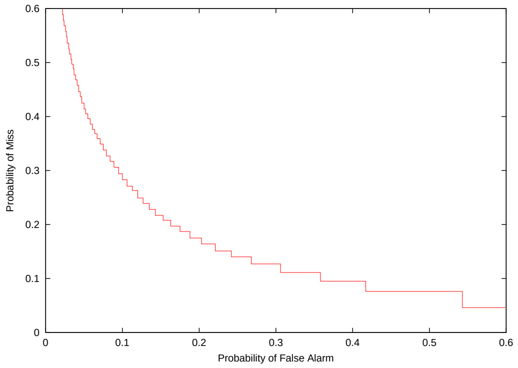


Figure 3: DET curve for detecting disfluent breaks from acoustics.

3 Syntactic parallelism

The other striking property of speech repairs is their parallel character: subsequent repair regions 'line up' with preceding reparandum regions. This property can be harnessed to better estimate the length of the reparandum by considering parallelism from the perspective of syntax. For instance, in Figure 4(a) the unfinished reparandum noun phrase is repaired by another noun phrase – the syntactic categories are parallel.

3.1 Levelt's WFR and Conjunction

The idea that the reparandum is syntactically parallel to the repair can be traced back to Levelt (1983). Examining a corpus of Dutch picture descriptions, Levelt proposes a bi-conditional well-formedness rule for repairs (WFR) that relates the structure of repairs to the structure of conjunctions. The WFR conceptualizes repairs as the conjunction of an unfinished reparandum string (α) with a properly finished repair (γ). Its original formulation, repeated here, ignores optional interregna like "er" or "I mean."

Well-formedness rule for repairs (WFR) A repair $\langle \alpha\, \gamma \rangle$ is well-formed if and only if there is a string β such that the string $\langle \alpha\beta$ *and** $\gamma \rangle$ is well-formed, where β is a completion of the constituent directly dominating the last element of α. (*and* is to be deleted if that last element is itself a sentence connective)

In other words, the string α is a prefix of a phrase whose completion, β—if it were present—would render the whole phrase $\alpha\beta$ grammatically conjoinable with the repair γ. In example (1) α is the string ‘the first kind of invasion of’, γ is ‘the first type of privacy’ and β is probably the single word ‘privacy.’

This kind of conjoinability typically requires the syntactic categories of the conjuncts to be the same (Chomsky, 1957, 36). That is, a rule schema such as (2) where X is a syntactic category, is preferred over one where X is not constrained to be the same on either side of the conjunction.

$$\mathrm{X} \;\rightarrow\; \mathrm{X\,Conj\,X} \tag{2}$$

If, as schema (2) suggests, conjunction does favor like-categories, and, as Levelt suggests, well-formed repairs are conjoinable with finished versions of their reparanda, then the syntactic categories of repairs ought to match the syntactic categories of (finished versions of) reparanda.

3.2 A WFR for grammars

Levelt’s WFR imposes two requirements on a grammar

- distinguishing a separate category of ‘unfinished’ phrases
- identifying a syntactic category for reparanda

Both requirements can be met by adapting Treebank grammars to mirror the analysis of McKelvie[1] (1998a; 1998b). McKelvie derives phrase structure rules for speech repairs from fluent rules by adding a new feature called abort that can take values true and false. For a given grammar rule of the form

$$\mathrm{A} \rightarrow \mathrm{B\,C}$$

a metarule creates other rules of the form

$$\begin{array}{l}\mathrm{A}\,[\mathtt{abort} = Q] \rightarrow \\ \qquad \mathrm{B}\,[\mathtt{abort} = \mathit{false}]\;\mathrm{C}\,[\mathtt{abort} = Q]\end{array}$$

where Q is a propositional variable. These rules say, in effect, that the constituent A is aborted just in case the last daughter C is aborted. Rules that don’t involve a constant value for Q ensure that the same value appears on parents and children. The WFR is then implemented by rule schemas such as (3)

$$\mathrm{X} \rightarrow \mathrm{X}\,[\mathtt{abort} = \mathit{true}]\,(\mathrm{AFF})\,\mathrm{X} \tag{3}$$

that permit the optional interregnum AFF to conjoin an unfinished X-phrase (the reparandum) with a finished X-phrase (the repair) that comes after it.

3.3 A WFR for Treebanks

McKelvie’s formulation of Levelt’s WFR can be applied to Treebanks by systematically recoding the annotations to indicate which phrases are unfinished and to distinguish matching from non-matching repairs.

3.3.1 Unfinished phrases

Some Treebanks already mark unfinished phrases. For instance, the Penn Treebank policy (Marcus et al., 1993; Marcus et al., 1994) is to annotate the lowest node that is unfinished with an -UNF tag as in Figure 4(a).

It is straightforward to propagate this mark upwards in the tree from wherever it is annotated to the nearest enclosing EDITED node, just as -BRK is propagated upwards from disjuncture marks on individual words. This percolation simulates the action of McKelvie’s $[\mathtt{abort} = \mathit{true}]$. The resulting PCFG is one in which distributions on phrase structure rules with ‘missing’ daughters are segregated from distributions on ‘complete’ rules.

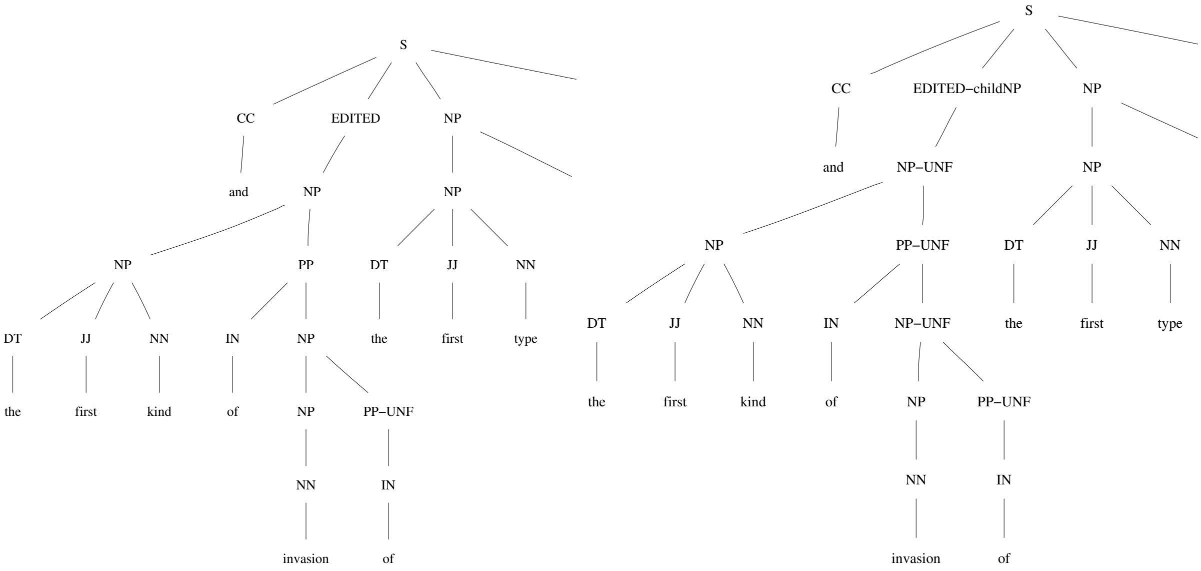
3.4 Reparanda categories

The other key element of Levelt’s WFR is the idea of conjunction of elements that are in some sense the same. In the Penn Treebank annotation scheme, reparanda always receive the label EDITED. This means that the syntactic category of the reparandum is hidden from any rule which could favor matching it with that of the repair. Adding an additional mark on this EDITED node (a kind of daughter annotation) rectifies the situation, as depicted in Figure 4(b), which adds the notation -childNP to a tree in which the unfinished tags have been propagated upwards. This allows a Treebank PCFG to represent the generalization that speech repairs tend to respect syntactic category.

4 Results

Three kinds of experiments examined the effectiveness of syntactic and prosodic indicators of

[1] McKelvie’s metarule approach declaratively expresses Hindle’s (1983) Stack Editor and Category Copy Editor rules. This classic work effectively states the WFR as a program for the Fidditch deterministic parser.



(a) The lowest unfinished node is given.

(b) -UNF propagated, daughter-annotated Switchboard tree

Figure 4: Input (a) and output (b) of tree transformations.

speech repairs. The first two use the CYK algorithm to find the most likely parse tree on a grammar read-off from example trees annotated as in Figures 2 and 4. The third experiment measures the benefit from syntactic indicators alone in Charniak's lexicalized parser (Charniak, 2000). The tables in subsections 4.1, 4.2, and 4.3 summarize the accuracy of output parse trees on two measures. One is the standard Parseval F-measure, which tracks the precision and recall for all labeled constituents as compared to a gold-standard parse. The other measure, EDIT-finding F, restricts consideration to just constituents that are reparanda. It measures the per-word performance identifying a word as dominated by EDITED or not. As in previous studies, reference transcripts were used in all cases. A check $(\surd)$ indicates an experiment where prosodic breaks where automatically inferred by the classifier described in section 2, whereas in the $(\times)$ rows no prosodic information was used.

4.1 CYK on Fisher

Table 1 summarizes the accuracy of a standard CYK parser on the newly-treebanked Fisher corpus (LDC2005E15) of phone conversations, collected as part of the DARPA EARS program. The parser was trained on the entire Switchboard corpus (ca. 107K utterances) then tested on the 5368-utterance 'dev2' subset of the Fisher data. This test set was tagged using MXPOST (Ratnaparkhi, 1996) which was itself trained on Switchboard. Finally, as described in section 2 these tags were augmented with a special prosodic break symbol if the decision tree rated the probability a ToBI 'p' symbol higher than the threshold value of 0.75.

Annotation	**Break index**	**Parseval F**	**EDIT F**
none	×	66.54	22.9
	√	66.08	26.1
daughter annotation	×	66.41	29.4
	√	65.81	31.6
-UNF propagation	×	67.06	31.5
	√	66.45	34.8
both	×	69.21	40.2
	√	67.02	40.6

Table 1: Improvement on Fisher, MXPOSTed tags.

The Fisher results in Table 1 show that syntactic and prosodic indicators provide different kinds of benefits that combine in an additive way. Presumably because of state-splitting, improvement in EDIT-finding comes at the cost of a small decrement in overall parsing performance.

4.2 CYK on Switchboard

Table 2 presents the results of similar experiments on the Switchboard corpus following the train/dev/test partition of Charniak and Johnson (2001). In these experiments, the parser was given correct part-of-speech tags as input.

Annotation	Break index	Parseval F	EDIT F
none	×	70.92	18.2
	√	69.98	22.5
daughter annotation	×	71.13	25.0
	√	70.06	25.5
-UNF propagation	×	71.71	31.1
	√	70.36	30.0
both	×	71.16	41.7
	√	71.05	36.2

Table 2: Improvement on Switchboard, gold tags.

The Switchboard results demonstrate independent improvement from the syntactic annotations. The prosodic annotation helps on its own and in combination with the daughter annotation that implements Levelt's WFR.

4.3 Lexicalized parser

Finally, Table 3 reports the performance of Charniak's non-reranking, lexicalized parser on the Switchboard corpus, using the same test/dev/train partition.

Annotation	Parseval F	EDIT F
baseline	83.86	57.6
daughter annotation	80.85	67.2
-UNF propagation	81.68	64.7
both	80.16	70.0
flattened EDITED	82.13	64.4

Table 3: Charniak as an improved EDIT-finder.

Since Charniak's parser does its own tagging, this experiment did not examine the utility of prosodic disjuncture marks. However, the combination of daughter annotation and `-UNF` propagation does lead to a better grammar-based reparandum-finder than parsers trained on flattened EDITED regions. More broadly, the results suggest that Levelt's WFR is synergistic with the kind of head-to-head lexical dependencies that Charniak's parser uses.

5 Discussion

The pattern of improvement in tables 1, 2, and 3 from *none* or *baseline* rows where no syntactic parallelism or break index information is used, to subsequent rows where it is used, suggest why these techniques work. Unfinished-category annotation improves performance by preventing the grammar of unfinished constituents from being polluted by the grammar of finished constituents. Such purification is independent of the fact that rules with daughters labeled `EDITED-childXP` tend to also mention categories labeled XP further to the right (or NP and VP, when XP starts with S). This preference for syntactic parallelism can be triggered either by externally-suggested ToBI break indices or grammar rules annotated with `-UNF`. The prediction of a disfluent break could be further improved by POS features and N-gram language model scores (Spilker et al., 2001; Liu, 2004).

6 Related Work

There have been relatively few attempts to harness prosodic cues in parsing. In a spoken language system for VERBMOBIL task, Batliner and colleagues (2001) utilize prosodic cues to dramatically reduce lexical analyses of disfluencies in a end-to-end real-time system. They tackle speech repair by a cascade of two stages – identification of potential interruption points using prosodic cues with 90% recall and many false alarms, and the lexical analyses of their neighborhood. Their approach, however, does not exploit the synergy between prosodic and syntactic features in speech repair. In Gregory et al. (2004), over 100 real-valued acoustic and prosodic features were quantized into a heuristically selected set of discrete symbols, which were then treated as pseudo-punctuation in a PCFG, assuming that prosodic cues function like punctuation. The resulting grammar suffered from data sparsity and failed to provide any benefits. Maximum entropy based models have been more successful in utilizing prosodic cues. For instance, in Lease et al. (2005), interruption point probabilities, predicted by prosodic classifiers, were quantized and introduced as features into a speech repair model along with a variety of TAG and PCFG features. Towards a clearer picture of the interaction with syntax and prosody, this work uses ToBI to capture prosodic cues. Such a method is analogous to Kahn et al. (2005) but in a generative framework.

The TAG-based model of Johnson and Charniak (2004) is a separate-processing approach that represents the state of the art in reparandum-finding.

Johnson and Charniak explicitly model the crossed dependencies between individual words in the reparandum and repair regions, intersecting this sequence model with a parser-derived language model for fluent speech. This second step improves on Stolcke and Shriberg (1996) and Heeman and Allen (1999) and outperforms the specific grammar-based reparandum-finders tested in section 4. However, because of separate-processing the TAG channel model's analyses do not reflect the syntactic structure of the sentence being analyzed, and thus that particular TAG-based model cannot make use of properties that depend on the phrase structure of the reparandum region. This includes the syntactic category parallelism discussed in section 3 but also predicate-argument structure. If edit hypotheses were augmented to mention particular tree nodes where the reparandum should be attached, such syntactic parallelism constraints could be exploited in the reranking framework of Johnson et al. (2004).

The approach in section 3 is more closely related to that of Core and Schubert (1999) who also use metarules to allow a parser to switch from speaker to speaker as users interrupt one another. They describe their metarule facility as a modification of chart parsing that involves copying of specific arcs just in case specific conditions arise. That approach uses a combination of longest-first heuristics and thresholds rather than a complete probabilistic model such as a PCFG.

Section 3's PCFG approach can also be viewed as a declarative generalization of Roark's (2004) EDIT-CHILD function. This function helps an incremental parser decide upon particular tree-drawing actions in syntactically-parallel contexts like speech repairs. Whereas Roark conditions the expansion of the first constituent of the repair upon the corresponding first constituent of the reparandum, in the PCFG approach there exists a separate rule (and thus a separate probability) for each alternative sequence of reparandum constituents.

7 Conclusion

Conventional PCFGs can improve their detection of speech repairs by incorporating Lickley's hypothesis about interrupted prosody and by implementing Levelt's well-formedness rule. These benefits are additive.

The strengths of these simple tree-based techniques should be combinable with sophisticated string-based (Johnson and Charniak, 2004; Liu, 2004; Zhang and Weng, 2005) approaches by applying the methods of Wieling et al. (2005) for constraining parses by externally-suggested brackets.

References

L. Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

W. Buntine. 1992. Tree classication software. In *Technology 2002: The Third National Technology Transfer Conference and Exposition*, Baltimore.

E. Charniak and M. Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 118–126.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL-00*, pages 132–139.

N. Chomsky. 1957. *Syntactic Structures*. Anua Linguarum Series Minor 4, Series Volume 4. Mouton de Gruyter, The Hague.

M. G. Core and L. K. Schubert. 1999. A syntactic framework for speech repairs and other disruptions. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 413–420.

J. J. Godfrey, E. C. Holliman, and J. McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of ICASSP*, volume I, pages 517–520, San Francisco.

M. Gregory, M. Johnson, and E. Charniak. 2004. Sentence-internal prosody does not help parsing the way punctuation does. In *Proceedings of North American Association for Computational Linguistics*.

M. Harper, B. Dorr, J. Hale, B. Roark, I. Shafran, M. Lease, Y. Liu, M. Snover, and L. Yung. 2005. Parsing and spoken structural event detection. In *2005 Johns Hopkins Summer Workshop Final Report*.

P. A. Heeman and J. F. Allen. 1999. Speech repairs, intonational phrases and discourse markers: modeling speakers' utterances in spoken dialog. *Computational Linguistics*, 25(4):527–571.

D. Hindle. 1983. Deterministic parsing of syntactic non-fluencies. In *Proceedings of the ACL*.

M. Johnson and E. Charniak. 2004. A TAG-based noisy channel model of speech repairs. In *Proceedings of ACL*, pages 33–39.

M. Johnson, E. Charniak, and M. Lease. 2004. An improved model for recognizing disfluencies in conversational speech. In *Proceedings of Rich Transcription Workshop*.

J. G. Kahn, M. Lease, E. Charniak, M. Johnson, and M. Ostendorf. 2005. Effective use of prosody in parsing conversational speech. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 233–240.

M. Lease, E. Charniak, and M. Johnson. 2005. Parsing and its applications for conversational speech. In *Proceedings of ICASSP*.

W. J. M. Levelt. 1983. Monitoring and self-repair in speech. *Cognitive Science*, 14:41–104.

R. J. Lickley. 1996. Juncture cues to disfluency. In *Proceedings the International Conference on Speech and Language Processing*.

Y. Liu. 2004. *Structural Event Detection for Rich Transcription of Speech*. Ph.D. thesis, Purdue University.

M. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

M. Marcus, G. Kim, M. A. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. 1994. The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the 1994 ARPA Human Language Technology Workshop*.

D. McKelvie. 1998a. SDP – Spoken Dialog Parser. ESRC project on Robust Parsing and Part-of-speech Tagging of Transcribed Speech Corpora, May.

D. McKelvie. 1998b. The syntax of disfluency in spontaneous spoken language. ESRC project on Robust Parsing and Part-of-speech Tagging of Transcribed Speech Corpora, May.

C. Nakatani and J. Hirschberg. 1994. A corpus-based study of repair cues in spontaneous speech. *Journal of the Acoustical Society of America*, 95(3):1603–1616, March.

M. Ostendorf, I. Shafran, S. Shattuck-Hufnagel, L. Carmichael, and W. Byrne. 2001. A prosodically labelled database of spontaneous speech. In *Proc. ISCA Tutorial and Research Workshop on Prosody in Speech Recognition and Understanding*, pages 119–121.

P. Price, M. Ostendorf, S. Shattuck-Hufnagel, and C. Fong. 1991. The use of prosody in syntactic disambiguation. *Journal of the Acoustic Society of America*, 90:2956–2970.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of Empirical Methods in Natural Language Processing Conference*, pages 133–141.

B. Roark. 2004. Robust garden path parsing. *Natural Language Engineering*, 10(1):1–24.

E. Shriberg, L. Ferrer, S. Kajarekar, A. Venkataraman, and A. Stolcke. 2005. Modeling prosodic feature sequences for speaker recognition. *Speech Communication*, 46(3-4):455–472.

E. Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.D. thesis, UC Berkeley.

H. F. Silverman, M. Beckman, J. Pitrelli, M. Ostendorf, C. Wightman, P. Price, J. Pierrehumbert, and J. Hirshberg. 1992. ToBI: A standard for labeling English prosody. In *Proceedings of ICSLP*, volume 2, pages 867–870.

K. Sjlander, 2001. *The Snack sound visualization module*. Royal Institute of Technology in Stockholm. http://www.speech.kth.se/SNACK.

K. Sonmez, E. Shriberg, L. Heck, and M. Weintraub. 1998. Modeling dynamic prosodic variation for speaker verification. In *Proceedings of ICSLP*, volume 7, pages 3189–3192.

Jörg Spilker, Martin Klarner, and Günther Görz. 2000. Processing self-corrections in a speech-to-speech system. In Wolfgang Wahlster, editor, *Verbmobil: Foundations of speech-to-speech translation*, pages 131–140. Springer-Verlag, Berlin.

J. Spilker, A. Batliner, and E. Nöth. 2001. How to repair speech repairs in an end-to-end system. In R. Lickley and L. Shriberg, editors, *Proc. of ISCA Workshop on Disfluency in Spontaneous Speech*, pages 73–76.

A. Stolcke and E. Shriberg. 1996. Statistical language modeling for speech disfluencies. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 405–408, Atlanta, GA.

R. M. Weischedel and N. K. Sondheimer. 1983. Meta-rules as a basis for processing ill-formed input. *American Journal of Computational Linguistics*, 9(3-4):161–177.

M. Wieling, M-J. Nederhof, and G. van Noord. 2005. Parsing partially bracketed input. Talk presented at Computational Linguistics in the Netherlands.

D. Wong, M. Ostendorf, and J. G. Kahn. 2005. Using weakly supervised learning to improve prosody labeling. Technical Report UWEETR-2005-0003, University of Washington Electrical Engineering Dept.

Q. Zhang and F. Weng. 2005. Exploring features for identifying edited regions in disfluent sentences. In *Proceedings of the Nineth International Workshop on Parsing Technologies*, pages 179–185.

Dependency Parsing of Japanese Spoken Monologue Based on Clause Boundaries

Tomohiro Ohno[†a)] **Shigeki Matsubara**[‡] **Hideki Kashioka**[§]
Takehiko Maruyama[♯] and **Yasuyoshi Inagaki**[♮]
[†]Graduate School of Information Science, Nagoya University, Japan
[‡]Information Technology Center, Nagoya University, Japan
[§]ATR Spoken Language Communication Research Laboratories, Japan
[♯]The National Institute for Japanese Language, Japan
[♮]Faculty of Information Science and Technology, Aichi Prefectural University, Japan
[a)]`ohno@el.itc.nagoya-u.ac.jp`

Abstract

Spoken monologues feature greater sentence length and structural complexity than do spoken dialogues. To achieve high parsing performance for spoken monologues, it could prove effective to simplify the structure by dividing a sentence into suitable language units. This paper proposes a method for dependency parsing of Japanese monologues based on sentence segmentation. In this method, the dependency parsing is executed in two stages: at the clause level and the sentence level. First, the dependencies within a clause are identified by dividing a sentence into clauses and executing stochastic dependency parsing for each clause. Next, the dependencies over clause boundaries are identified stochastically, and the dependency structure of the entire sentence is thus completed. An experiment using a spoken monologue corpus shows this method to be effective for efficient dependency parsing of Japanese monologue sentences.

1 Introduction

Recently, monologue data such as a lecture and commentary by a professional have been considered as human valuable intellectual property and have gathered attention. In applications, such as automatic summarization, machine translation and so on, for using these monologue data as intellectual property effectively and efficiently, it is necessary not only just to accumulate but also to structure the monologue data. However, few attempts have been made to parse spoken monologues. Spontaneously spoken monologues include a lot of grammatically ill-formed linguistic phenomena such as fillers, hesitations and self-repairs. In order to robustly deal with their extragrammaticality, some techniques for parsing of dialogue sentences have been proposed (Core and Schubert, 1999; Delmonte, 2003; Ohno et al., 2005b). On the other hand, monologues also have the characteristic feature that a sentence is generally longer and structurally more complicated than a sentence in dialogues which have been dealt with by the previous researches. Therefore, for a monologue sentence the parsing time would increase and the parsing accuracy would decrease. It is thought that more effective, high-performance spoken monologue parsing could be achieved by dividing a sentence into suitable language units for simplicity.

This paper proposes a method for dependency parsing of monologue sentences based on sentence segmentation. The method executes dependency parsing in two stages: at the clause level and at the sentence level. First, a dependency relation from one *bunsetsu*[1] to another within a clause is identified by dividing a sentence into clauses based on clause boundary detection and then executing stochastic dependency parsing for each clause. Next, the dependency structure of the entire sentence is completed by identifying the dependencies over clause boundaries stochastically. An experiment on monologue dependency parsing showed that the parsing time can be drasti-

[1] A *bunsetsu* is the linguistic unit in Japanese that roughly corresponds to a basic phrase in English. A bunsetsu consists of one independent word and more than zero ancillary words. A *dependency* is a modification relation in which a *dependent bunsetsu* depends on a *head bunsetsu*. That is, the dependent bunsetsu and the head bunsetsu work as modifier and modifyee, respectively.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 169–176,
Sydney, July 2006. ©2006 Association for Computational Linguistics

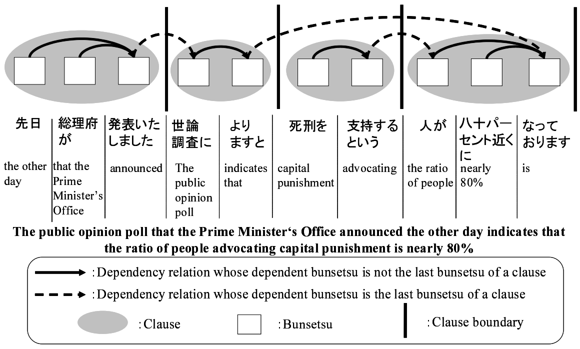

先日
the other day
総理府が
that the Prime Minister's Office
発表いたしました
announced
世論調査に
The public opinion poll
よりますと
indicates that
死刑を
capital punishment
支持するという
advocating
人が
the ratio of people
八十パーセント近くに
nearly 80%
なっております
is
The public opinion poll that the Prime Minister's Office announced the other day indicates that the ratio of people advocating capital punishment is nearly 80%
:Dependency relation whose dependent bunsetsu is not the last bunsetsu of a clause
:Dependency relation whose dependent bunsetsu is the last bunsetsu of a clause
:Clause
:Bunsetsu
:Clause boundary

Figure 1: Relation between clause boundary and dependency structure

cally shortened and the parsing accuracy can be increased.

This paper is organized as follows: The next section describes a parsing unit of Japanese monologue. Section 3 presents dependency parsing based on clause boundaries. The parsing experiment and the discussion are reported in Sections 4 and 5, respectively. The related works are described in Section 6.

2 Parsing Unit of Japanese Monologues

Our method achieves an efficient parsing by adopting a shorter unit than a sentence as a parsing unit. Since the search range of a dependency relation can be narrowed by dividing a long monologue sentence into small units, we can expect the parsing time to be shortened.

2.1 Clauses and Dependencies

In Japanese, a clause basically contains one verb phrase. Therefore, a complex sentence or a compound sentence contains one or more clauses. Moreover, since a clause constitutes a syntactically sufficient and semantically meaningful language unit, it can be used as an alternative parsing unit to a sentence.

Our proposed method assumes that a sentence is a sequence of one or more clauses, and every bunsetsu in a clause, except the final bunsetsu, depends on another bunsetsu in the same clause. As an example, the dependency structure of the Japanese sentence:

> 先日総理府が発表いたしました世論調査によりますと死刑を支持するという人が八十パーセント近くになっております（The public opinion poll that the Prime Minister's Office announced the other day indicates that the ratio of people advocating capital punishment is nearly 80%)

is presented in Fig. 1. This sentence consists of four clauses:

- 先日総理府が発表いたしました (that the Prime Minister's Office announced the other day)
- 世論調査によりますと (The public opinion poll indicates that)
- 死刑を支持するという (advocating capital punishment)
- 人が八十パーセント近くになっております (the ratio of people is nearly 80%)

Each clause forms a dependency structure (solid arrows in Fig. 1), and a dependency relation from the final bunsetsu links the clause with another clause (dotted arrows in Fig. 1).

2.2 Clause Boundary Unit

In adopting a clause as an alternative parsing unit, it is necessary to divide a monologue sentence into clauses as the preprocessing for the following dependency parsing. However, since some kinds of clauses are embedded in main clauses, it is fundamentally difficult to divide a monologue into clauses in one dimension (Kashioka and Maruyama, 2004).

Therefore, by using a clause boundary annotation program (Maruyama et al., 2004), we approximately achieve the clause segmentation of a monologue sentence. This program can identify units corresponding to clauses by detecting the end boundaries of clauses. Furthermore, the program can specify the positions and types of clause boundaries simply from a local morphological analysis. That is, for a sentence morphologically analyzed by ChaSen (Matsumoto et al., 1999), the positions of clause boundaries are identified and clause boundary labels are inserted there. There exist 147 labels such as "compound clause" and "adnominal clause." [2]

In our research, we adopt the unit sandwiched between two clause boundaries detected by clause boundary analysis, were called the ***clause boundary unit***, as an alternative parsing unit. Here, we regard the label name provided for the end boundary of a clause boundary unit as that unit's type.

[2] The labels include a few other constituents that do not strictly represent clause boundaries but can be regarded as being syntactically independent elements, such as "topicalized element," "conjunctives," "interjections," and so on.

Table 1: 200 sentences in “Asu-Wo-Yomu”

sentences	200
clause boundary units	951
bunsetsus	2,430
morphemes	6,017
dependencies over clause boundaries	94

2.3 Relation between Clause Boundary Units and Dependency Structures

To clarify the relation between clause boundary units and dependency structures, we investigated the monologue corpus “Asu-Wo-Yomu [3].” In the investigation, we used 200 sentences for which morphological analysis, bunsetsu segmentation, clause boundary analysis, and dependency parsing were automatically performed and then modified by hand. Here, the specification of the parts-of-speech is in accordance with that of the IPA parts-of-speech used in the ChaSen morphological analyzer (Matsumoto et al., 1999), the rules of the bunsetsu segmentation with those of CSJ (Maekawa et al., 2000), the rules of the clause boundary analysis with those of Maruyama et al. (Maruyama et al., 2004), and the dependency grammar with that of the Kyoto Corpus (Kurohashi and Nagao, 1997).

Table 1 shows the results of analyzing the 200 sentences. Among the 1,479 bunsetsus in the difference set between all bunsetsus (2,430) and the final bunsetsus (951) of clause boundary units, only 94 bunsetsus depend on a bunsetsu located outside the clause boundary unit. This result means that 93.6% (1,385/1,479) of all dependency relations are within a clause boundary unit. Therefore, the results confirmed that the assumption made by our research is valid to some extent.

3 Dependency Parsing Based on Clause Boundaries

In accordance with the assumption described in Section 2, in our method, the transcribed sentence on which morphological analysis, clause boundary detection, and bunsetsu segmentation are performed is considered the input [4]. The dependency parsing is executed based on the following procedures:

1. **Clause-level parsing:** The internal dependency relations of clause boundary units are identified for every clause boundary unit in one sentence.
2. **Sentence-level parsing:** The dependency relations in which the dependent unit is the final bunsetsu of the clause boundary units are identified.

In this paper, we describe a sequence of clause boundary units in a sentence as $C_1 \cdots C_m$, a sequence of bunsetsus in a clause boundary unit C_i as $b_1^i \cdots b_{n_i}^i$, a dependency relation in which the dependent bunsetsu is a bunsetsu b_k^i as $dep(b_k^i)$, and a dependency structure of a sentence as $\{dep(b_1^1), \cdots, dep(b_{n_m-1}^m)\}$.

First, our method parses the dependency structure $\{dep(b_1^i), \cdots, dep(b_{n_i-1}^i)\}$ within the clause boundary unit whenever a clause boundary unit C_i is inputted. Then, it parses the dependency structure $\{dep(b_{n_1}^1), \cdots, dep(b_{n_{m-1}}^{m-1})\}$, which is a set of dependency relations whose dependent bunsetsu is the final bunsetsu of each clause boundary unit in the input sentence. In addition, in both of the above procedures, our method assumes the following three syntactic constraints:

1. No dependency is directed from right to left.
2. Dependencies don’t cross each other.
3. Each bunsetsu, except the final one in a sentence, depends on only one bunsetsu.

These constraints are usually used for Japanese dependency parsing.

3.1 Clause-level Dependency Parsing

Dependency parsing within a clause boundary unit, when the sequence of bunsetsus in an input clause boundary unit C_i is described as B_i $(= b_1^i \cdots b_{n_i}^i)$, identifies the dependency structure S_i $(= \{dep(b_1^i), \cdots, dep(b_{n_i-1}^i)\})$, which maximizes the conditional probability $P(S_i|B_i)$. At this level, the head bunsetsu of the final bunsetsu $b_{n_i}^i$ of a clause boundary unit is not identified.

Assuming that each dependency is independent of the others, $P(S_i|B_i)$ can be calculated as follows:

$$P(S_i|B_i) = \prod_{k=1}^{n_i-1} P(b_k^i \overset{rel}{\rightarrow} b_l^i|B_i), \qquad (1)$$

[3] Asu-Wo-Yomu is a collection of transcriptions of a TV commentary program of the Japan Broadcasting Corporation (NHK). The commentator speaks on some current social issue for 10 minutes.

[4] It is difficult to preliminarily divide a monologue into sentences because there are no clear sentence breaks in monologues. However, since some methods for detecting sentence boundaries have already been proposed (Huang and Zweig, 2002; Shitaoka et al., 2004), we assume that they can be detected automatically before dependency parsing.

where $P(b_k^i \overset{rel}{\to} b_l^i|B_i)$ is the probability that a bunsetsu b_k^i depends on a bunsetsu b_l^i when the sequence of bunsetsus B_i is provided. Unlike the conventional stochastic sentence-by-sentence dependency parsing method, in our method, B_i is the sequence of bunsetsus that constitutes not a sentence but a clause. The structure S_i, which maximizes the conditional probability $P(S_i|B_i)$, is regarded as the dependency structure of B_i and calculated by dynamic programming (DP).

Next, we explain the calculation of $P(b_k^i \overset{rel}{\to} b_l^i|B_i)$. First, the basic form of independent words in a dependent bunsetsu is represented by h_k^i, its parts-of-speech t_k^i, and type of dependency r_k^i, while the basic form of the independent word in a head bunsetsu is represented by h_l^i, and its parts-of-speech t_l^i. Furthermore, the distance between bunsetsus is described as d_{kl}^{ii}. Here, if a dependent bunsetsu has one or more ancillary words, the type of dependency is the lexicon, part-of-speech and conjugated form of the rightmost ancillary word, and if not so, it is the part-of-speech and conjugated form of the rightmost morpheme. The type of dependency r_k^i is the same attribute used in our stochastic method proposed for robust dependency parsing of spoken language dialogue (Ohno et al., 2005b). Then d_{kl}^{ii} takes 1 or more than 1, that is, a binary value. Incidentally, the above attributes are the same as those used by the conventional stochastic dependency parsing methods (Collins, 1996; Ratnaparkhi, 1997; Fujio and Matsumoto, 1998; Uchimoto et al., 1999; Charniak, 2000; Kudo and Matsumoto, 2002).

Additionally, we prepared the attribute e_l^i to indicate whether b_l^i is the final bunsetsu of a clause boundary unit. Since we can consider a clause boundary unit as a unit corresponding to a simple sentence, we can treat the final bunsetsu of a clause boundary unit as a sentence-end bunsetsu. The attribute that indicates whether a head bunsetsu is a sentence-end bunsetsu has often been used in conventional sentence-by-sentence parsing methods (e.g. Uchimoto et al., 1999).

By using the above attributes, the conditional probability $P(b_k^i \overset{rel}{\to} b_l^i|B_i)$ is calculated as follows:

$$\begin{aligned}
&P(b_k^i \overset{rel}{\to} b_l^i|B_i) \qquad\qquad (2)\\
&\cong \quad P(b_k^i \overset{rel}{\to} b_l^i|h_k^i, h_l^i, t_k^i, t_l^i, r_k^i, d_{kl}^{ii}, e_l^i)\\
&= \quad \frac{F(b_k^i \overset{rel}{\to} b_l^i, h_k^i, h_l^i, t_k^i, t_l^i, r_k^i, d_{kl}^{ii}, e_l^i)}{F(h_k^i, h_l^i, t_k^i, t_l^i, r_k^i, d_{kl}^{ii}, e_l^i)}.
\end{aligned}$$

Note that F is a co-occurrence frequency function.

In order to resolve the sparse data problems caused by estimating $P(b_k^i \overset{rel}{\to} b_l^i|B_i)$ with formula (2), we adopted the smoothing method described by Fujio and Matsumoto (Fujio and Matsumoto, 1998): if $F(h_k^i, h_l^i, t_k^i, t_l^i, r_k^i, d_{kl}^{ii}, e_l^i)$ in formula (2) is 0, we estimate $P(b_k^i \overset{rel}{\to} b_l^i|B_i)$ by using formula (3).

$$\begin{aligned}
&P(b_k^i \overset{rel}{\to} b_l^i|B_i) \qquad\qquad (3)\\
&\cong \quad P(b_k^i \overset{rel}{\to} b_l^i|t_k^i, t_l^i, r_k^i, d_{kl}^{ii}, e_l^i)\\
&= \quad \frac{F(b_k^i \overset{rel}{\to} b_l^i, t_k^i, t_l^i, r_k^i, d_{kl}^{ii}, e_l^i)}{F(t_k^i, t_l^i, r_k^i, d_{kl}^{ii}, e_l^i)}
\end{aligned}$$

3.2 Sentence-level Dependency Parsing

Here, the head bunsetsu of the final bunsetsu of a clause boundary unit is identified. Let $B\ (= B_1 \cdots B_n)$ be the sequence of bunsetsus of one sentence and S_{fin} be a set of dependency relations whose dependent bunsetsu is the final bunsetsu of a clause boundary unit, $\{dep(b_{n_1}^1), \cdots, dep(b_{n_{m-1}}^{m-1})\}$; then S_{fin}, which makes $P(S_{fin}|B)$ the maximum, is calculated by DP. The $P(S_{fin}|B)$ can be calculated as follows:

$$P(S_{fin}|B) = \prod_{i=1}^{m-1} P(b_{n_i}^i \overset{rel}{\to} b_l^j|B), \qquad (4)$$

where $P(b_{n_i}^i \overset{rel}{\to} b_l^j|B)$ is the probability that a bunsetsu $b_{n_i}^i$ depends on a bunsetsu b_l^j when the sequence of the sentence's bunsetsus, B, is provided. Our method parses by giving consideration to the dependency structures in each clause boundary unit, which were previously parsed. That is, the method does not consider all bunsetsus located on the right-hand side as candidates for a head bunsetsu but calculates only dependency relations within each clause boundary unit that do not cross any other relation in previously parsed dependency structures. In the case of Fig. 1, the method calculates by assuming that only three bunsetsus "人が (the ratio of people)," or "なっております (is)" can be the head bunsetsu of the bunsetsu "指示するという (advocating)."

In addition, $P(b_{n_i}^i \overset{rel}{\to} b_l^j|B)$ is calculated as in Eq. (5). Equation (5) uses all of the attributes used in Eq. (2), in addition to the attribute s_l^j, which indicates whether the head bunsetsu of b_l^j is the final bunsetsu of a sentence. Here, we take into

Table 2: Size of experimental data set (Asu-Wo-Yomu)

	test data	learning data
programs	8	95
sentences	500	5,532
clause boundary units	2,237	26,318
bunsetsus	5,298	65,821
morphemes	13,342	165,129

Note that the commentator of each program is different.

Table 3: Experimental results on parsing time

	our method	conv. method
average time (msec)	10.9	51.9

programming language: LISP
computer used: Pentium4 2.4 GHz, Linux

account the analysis result that about 70% of the final bunsetsus of clause boundary units depend on the final bunsetsu of other clause boundary units [5] and also use the attribute e^j_l at this phase.

$$
\begin{aligned}
&P(b^i_{n_i} \overset{rel}{\rightarrow} b^j_l | B) \qquad (5)\\
&\cong P(b^i_{n_i} \overset{rel}{\rightarrow} b^j_l | h^i_{n_i}, h^j_l, t^i_{n_i}, t^j_l, r^i_{n_i}, d^{ij}_{n_i l}, e^j_l, s^j_l)\\
&= \frac{F(b^i_{n_i} \overset{rel}{\rightarrow} b^j_l, h^i_{n_i}, h^j_l, t^i_{n_i}, t^j_l, r^i_{n_i}, d^{ij}_{n_i l}, e^j_l, s^j_l)}{F(h^i_{n_i}, h^j_l, t^i_{n_i}, t^j_l, r^i_{n_i}, d^{ij}_{n_i l}, e^j_l, s^j_l)}
\end{aligned}
$$

4 Parsing Experiment

To evaluate the effectiveness of our method for Japanese spoken monologue, we conducted an experiment on dependency parsing.

4.1 Outline of Experiment

We used the spoken monologue corpus "Asu-Wo-Yomu," annotated with information on morphological analysis, clause boundary detection, bunsetsu segmentation, and dependency analysis[6]. Table 2 shows the data used for the experiment. We used 500 sentences as the test data. Although our method assumes that a dependency relation does not cross clause boundaries, there were 152 dependency relations that contradicted this assumption. This means that the dependency accuracy of our method is not over 96.8% (4,646/4,798). On the other hand, we used 5,532 sentences as the learning data.

To carry out comparative evaluation of our method's effectiveness, we executed parsing for the above-mentioned data by the following two methods and obtained, respectively, the parsing time and parsing accuracy.

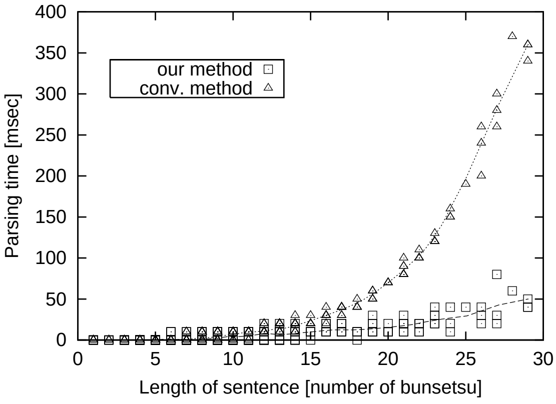


Figure 2: Relation between sentence length and parsing time

- **Our method:** First, our method provides clause boundaries for a sequence of bunsetsus of an input sentence and identifies all clause boundary units in a sentence by performing clause boundary analysis (CBAP) (Maruyama et al., 2004). After that, our method executes the dependency parsing described in Section 3.
- **Conventional method:** This method parses a sentence at one time without dividing it into clause boundary units. Here, the probability that a bunsetsu depends on another bunsetsu, when the sequence of bunsetsus of a sentence is provided, is calculated as in Eq. (5), where the attribute e was eliminated. This conventional method has been implemented by us based on the previous research (Fujio and Matsumoto, 1998).

4.2 Experimental Results

The parsing times of both methods are shown in Table 3. The parsing speed of our method improves by about 5 times on average in comparison with the conventional method. Here, the parsing time of our method includes the time taken not only for the dependency parsing but also for the clause boundary analysis. The average time taken for clause boundary analysis was about 1.2 millisecond per sentence. Therefore, the time cost of performing clause boundary analysis as a preprocessing of dependency parsing can be considered small enough to disregard. Figure 2 shows the relation between sentence length and parsing time

[5] We analyzed the 200 sentences described in Section 2.3 and confirmed 70.6% (522/751) of the final bunsetsus of clause boundary units depended on the final bunsetsu of other clause boundary units.

[6] Here, the specifications of these annotations are in accordance with those described in Section 2.3.

Table 4: Experimental results on parsing accuracy

	our method	conv. method
bunsetsu within a clause boundary unit (except final bunsetsu)	88.2% (2,701/3,061)	84.7% (2,592/3,061)
final bunsetsu of a clause boundary unit	65.6% (1,140/1,737)	63.3% (1,100/1,737)
total	80.1% (3,841/4,798)	76.9% (3,692/4,798)

Table 5: Experimental results on clause boundary analysis (CBAP)

recall	95.7% (2,140/2,237)
precision	96.9% (2,140/2,209)

for both methods, and it is clear from this figure that the parsing time of the conventional method begins to rapidly increase when the length of a sentence becomes 12 or more bunsetsus. In contrast, our method changes little in relation to parsing time. Here, since the sentences used in the experiment are composed of 11.8 bunsetsus on average, this result shows that our method is suitable for improving the parsing time of a monologue sentence whose length is longer than the average.

Table 4 shows the parsing accuracy of both methods. The first line of Table 4 shows the parsing accuracy for all bunsetsus within clause boundary units except the final bunsetsus of the clause boundary units. The second line shows the parsing accuracy for the final bunsetsus of all clause boundary units except the sentence-end bunsetsus. We confirmed that our method could analyze with a higher accuracy than the conventional method. Here, Table 5 shows the accuracy of the clause boundary analysis executed by CBAP. Since the precision and recall is high, we can assume that the clause boundary analysis exerts almost no harmful influence on the following dependency parsing.

As mentioned above, it is clear that our method is more effective than the conventional method in shortening parsing time and increasing parsing accuracy.

5 Discussions

Our method assumes that dependency relations within a clause boundary unit do not cross clause boundaries. Due to this assumption, the method cannot correctly parse the dependency relations over clause boundaries. However, the experimental results indicated that the accuracy of our method was higher than that of the conventional method.

In this section, we first discuss the effect of our method on parsing accuracy, separately for bunsetsus within clause boundary units (except the final bunsetsus) and the final bunsetsus of clause boundary units. Next, we discuss the problem of our method's inability to parse dependency relations over clause boundaries.

Table 6: Comparison of parsing accuracy between conventional method and our method (for bunsetsu within a clause boundary unit except final bunsetsu)

our method conv. method	correct	incorrect	total
correct	2,499	93	2,592
incorrect	202	267	469
total	2,701	360	3,061

5.1 Parsing Accuracy for Bunsetsu within a Clause Boundary Unit (except final bunsetsu)

Table 6 compares parsing accuracies for bunsetsus within clause boundary units (except the final bunsetsus) between the conventional method and our method. There are 3,061 bunsetsus within clause boundary units except the final bunsetsu, among which 2,499 were correctly parsed by both methods. There were 202 dependency relations correctly parsed by our method but incorrectly parsed by the conventional method. This means that our method can narrow down the candidates for a head bunsetsu.

In contrast, 93 dependency relations were correctly parsed solely by the conventional method. Among these, 46 were dependency relations over clause boundaries, which cannot in principle be parsed by our method. This means that our method can correctly parse almost all of the dependency relations that the conventional method can correctly parse except for dependency relations over clause boundaries.

5.2 Parsing Accuracy for Final Bunsetsu of a Clause Boundary Unit

We can see from Table 4 that the parsing accuracy for the final bunsetsus of clause boundary units by both methods is much worse than that for bunsetsus within the clause boundary units (except the final bunsetsus). This means that it is difficult

Table 7: Comparison of parsing accuracy between conventional method and our method (for final bunsetsu of a clause boundary unit)

our method conv. method	correct	incorrect	total
correct	1037	63	1,100
incorrect	103	534	637
total	1,140	597	1,737

Table 8: Parsing accuracy for dependency relations over clause boundaries

	our method	conv. method
recall	1.3% (2/152)	30.3% (46/152)
precision	11.8% (2/ 17)	25.3% (46/182)

to identify dependency relations whose dependent bunsetsu is the final one of a clause boundary unit.

Table 7 compares how the two methods parse the dependency relations when the dependent bunsetsu is the final bunsetsu of a clause boundary unit. There are 1,737 dependency relations whose dependent bunsetsu is the final bunsetsu of a clause boundary unit, among which 1,037 were correctly parsed by both methods. The number of dependency relations correctly parsed only by our method was 103. This number is higher than that of dependency relations correctly parsed by only the conventional method. This result might be attributed to our method's effect; that is, our method narrows down the candidates internally for a head bunsetsu based on the first-parsed dependency structure for clause boundary units.

5.3 Dependency Relations over Clause Boundaries

Table 8 shows the accuracy of both methods for parsing dependency relations over clause boundaries. Since our method parses based on the assumption that those dependency relations do not exist, it cannot correctly parse anything. Although, from the experimental results, our method could identify two dependency relations over clause boundaries, these were identified only because dependency parsing for some sentences was performed based on wrong clause boundaries that were provided by clause boundary analysis. On the other hand, the conventional method correctly parsed 46 dependency relations among 152 that crossed a clause boundary in the test data. Since the conventional method could correctly parse only 30.3% of those dependency relations, we can see that it is in principle difficult to identify the dependency relations.

6 Related Works

Since monologue sentences tend to be long and have complex structures, it is important to consider the features. Although there have been very few studies on parsing monologue sentences, some studies on parsing written language have dealt with long-sentence parsing. To resolve the syntactic ambiguity of a long sentence, some of them have focused attention on the "clause."

First, there are the studies that focused attention on compound clauses (Agarwal and Boggess, 1992; Kurohashi and Nagao, 1994). These tried to improve the parsing accuracy of long sentences by identifying the boundaries of coordinate structures. Next, other research efforts utilized the three categories into which various types of subordinate clauses are hierarchically classified based on the "scope-embedding preference" of Japanese subordinate clauses (Shirai et al., 1995; Utsuro et al., 2000). Furthermore, Kim et al. (Kim and Lee, 2004) divided a sentence into "S(ubject)-clauses," which were defined as a group of words containing several predicates and their common subject. The above studies have attempted to reduce the parsing ambiguity between specific types of clauses in order to improve the parsing accuracy of an entire sentence.

On the other hand, our method utilizes all types of clauses without limiting them to specific types of clauses. To improve the accuracy of long-sentence parsing, we thought that it would be more effective to cyclopaedically divide a sentence into all types of clauses and then parse the local dependency structure of each clause. Moreover, since our method can perform dependency parsing clause-by-clause, we can reasonably expect our method to be applicable to incremental parsing (Ohno et al., 2005a).

7 Conclusions

In this paper, we proposed a technique for dependency parsing of monologue sentences based on clause-boundary detection. The method can achieve more effective, high-performance spoken monologue parsing by dividing a sentence into clauses, which are considered as suitable language units for simplicity. To evaluate the effectiveness of our method for Japanese spoken monologue, we conducted an experiment on dependency parsing of the spoken monologue sentences recorded in the "Asu-Wo-Yomu." From the experimental results, we confirmed that our method shortened the parsing time and increased the parsing accuracy compared with the conventional method, which parses a sentence without dividing it into clauses.

Future research will include making a thorough investigation into the relation between dependency type and the type of clause boundary unit. After that, we plan to investigate techniques for identifying the dependency relations over clause boundaries. Furthermore, as the experiment described in this paper has shown the effectiveness of our technique for dependency parsing of long sentences in spoken monologues, so our technique can be expected to be effective in written language also. Therefore, we want to examine the effectiveness by conducting the parsing experiment of long sentences in written language such as newspaper articles.

8 Acknowledgements

This research was supported in part by a contract with the Strategic Information and Communications R&D Promotion Programme, Ministry of Internal Affairs and Communications and the Grand-in-Aid for Young Scientists of JSPS. The first author is partially supported by JSPS Research Fellowships for Young Scientists.

References

R. Agarwal and L. Boggess. 1992. A simple but useful approach to conjunct indentification. In *Proc. of 30th ACL*, pages 15–21.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. of 1st NAACL*, pages 132–139.

M. Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proc. of 34th ACL*, pages 184–191.

Mark G. Core and Lenhart K. Schubert. 1999. A syntactic framework for speech repairs and other disruptions. In *Proc. of 37th ACL*, pages 413–420.

R. Delmonte. 2003. Parsing spontaneous speech. In *Proc. of 8th EUROSPEECH*, pages 1999–2004.

M. Fujio and Y. Matsumoto. 1998. Japanese dependency structure analysis based on lexicalized statistics. In *Proc. of 3rd EMNLP*, pages 87–96.

J. Huang and G. Zweig. 2002. Maximum entropy model for punctuation annotation from speech. In *Proc. of 7th ICSLP*, pages 917–920.

H. Kashioka and T. Maruyama. 2004. Segmentation of semantic unit in Japanese monologue. In *Proc. of ICSLT-O-COCOSDA 2004*, pages 87–92.

M. Kim and J. Lee. 2004. Syntactic analysis of long sentences based on s-clauses. In *Proc. of 1st IJC-NLP*, pages 420–427.

T. Kudo and Y. Matsumoto. 2002. Japanese dependency analyisis using cascaded chunking. In *Proc. of 6th CoNLL*, pages 63–69.

S. Kurohashi and M. Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

S. Kurohashi and M. Nagao. 1997. Building a Japanese parsed corpus while improving the parsing system. In *Proc. of 4th NLPRS*, pages 451–456.

K. Maekawa, H. Koiso, S. Furui, and H. Isahara. 2000. Spontaneous speech corpus of Japanese. In *Proc. of 2nd LREC*, pages 947–952.

T. Maruyama, H. Kashioka, T. Kumano, and H. Tanaka. 2004. Development and evaluation of Japanese clause boundaries annotation program. *Journal of Natural Language Processing*, 11(3):39–68. (In Japanese).

Y. Matsumoto, A. Kitauchi, T. Yamashita, and Y. Hirano, 1999. *Japanese Morphological Analysis System ChaSen version 2.0 Manual*. NAIST Technical Report, NAIST-IS-TR99009.

T. Ohno, S. Matsubara, H. Kashioka, N. Kato, and Y. Inagaki. 2005a. Incremental dependency parsing of Japanese spoken monologue based on clause boundaries. In *Proc. of 9th EUROSPEECH*, pages 3449–3452.

T. Ohno, S. Matsubara, N. Kawaguchi, and Y. Inagaki. 2005b. Robust dependency parsing of spontaneous Japanese spoken language. *IEICE Transactions on Information and Systems*, E88-D(3):545–552.

A. Ratnaparkhi. 1997. A liner observed time statistical parser based on maximum entropy models. In *Proc. of 2nd EMNLP*, pages 1–10.

S. Shirai, S. Ikehara, A. Yokoo, and J. Kimura. 1995. A new dependency analysis method based on semantically embedded sentence structures and its performance on Japanese subordinate clause. *Journal of Information Processing Society of Japan*, 36(10):2353–2361. (In Japanese).

K. Shitaoka, K. Uchimoto, T. Kawahara, and H. Isahara. 2004. Dependency structure analysis and sentence boundary detection in spontaneous Japanese. In *Proc. of 20th COLING*, pages 1107–1113.

K. Uchimoto, S. Sekine, and K. Isahara. 1999. Japanese dependency structure analysis based on maximum entropy models. In *Proc. of 9th EACL*, pages 196–203.

T. Utsuro, S. Nishiokayama, M. Fujio, and Y. Matsumoto. 2000. Analyzing dependencies of Japanese subordinate clauses based on statistics of scope embedding preference. In *Proc. of 6th ANLP*, pages 110–117.

Trace Prediction and Recovery
With Unlexicalized PCFGs and Slash Features

Helmut Schmid
IMS, University of Stuttgart
schmid@ims.uni-stuttgart.de

Abstract

This paper describes a parser which generates parse trees with empty elements in which traces and fillers are co-indexed. The parser is an unlexicalized PCFG parser which is guaranteed to return the most probable parse. The grammar is extracted from a version of the PENN treebank which was automatically annotated with features in the style of Klein and Manning (2003). The annotation includes GPSG-style slash features which link traces and fillers, and other features which improve the general parsing accuracy. In an evaluation on the PENN treebank (Marcus et al., 1993), the parser outperformed other unlexicalized PCFG parsers in terms of labeled bracketing f-score. Its results for the empty category prediction task and the trace-filler co-indexation task exceed all previously reported results with 84.1% and 77.4% f-score, respectively.

1 Introduction

Empty categories (also called null elements) are used in the annotation of the PENN treebank (Marcus et al., 1993) in order to represent syntactic phenomena like constituent movement (e.g. wh-extraction), discontinuous constituents, and missing elements (PRO elements, empty complementizers and relative pronouns). Moved constituents are co-indexed with a trace which is located at the position where the moved constituent is to be interpreted. Figure 1 shows an example of constituent movement in a relative clause.

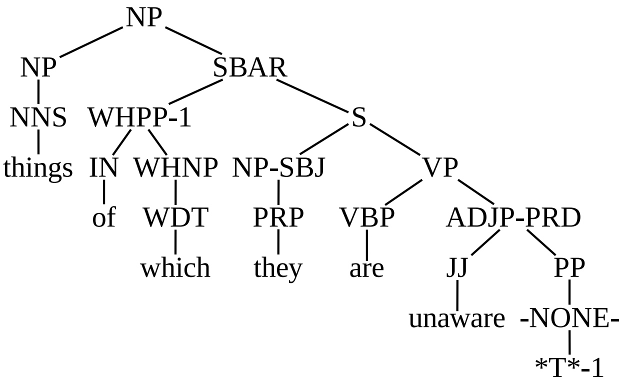


Figure 1: Co-indexation of traces and fillers

Empty categories provide important information for the semantic interpretation, in particular for determining the predicate-argument structure of a sentence. However, most broad-coverage statistical parsers (Collins, 1997; Charniak, 2000, and others) which are trained on the PENN treebank generate parse trees without empty categories. In order to augment such parsers with empty category prediction, three rather different strategies have been proposed: (i) pre-processing of the input sentence with a tagger which inserts empty categories into the input string of the parser (Dienes and Dubey, 2003b; Dienes and Dubey, 2003a). The parser treats the empty elements like normal input tokens. (ii) post-processing of the parse trees with a pattern matcher which adds empty categories after parsing (Johnson, 2001; Campbell, 2004; Levy and Manning, 2004) (iii) in-processing of the empty categories with a slash percolation mechanism (Dienes and Dubey, 2003b; Dienes and Dubey, 2003a). The empty elements are here generated by the grammar.

Good results have been obtained with all three approaches, but (Dienes and Dubey, 2003b) reported that in their experiments, the in-processing of the empty categories only worked with lexicalized parsing. They explain that their unlex-

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 177–184,
Sydney, July 2006. ©2006 Association for Computational Linguistics

icalized PCFG parser produced poor results because the beam search strategy applied there eliminated many correct constituents with empty elements. The scores of these constituents were too low compared with the scores of constituents without empty elements. They speculated that "doing an exhaustive search might help" here.

In this paper, we confirm this hypothesis and show that it is possible to accurately predict empty categories with unlexicalized PCFG parsing and slash features if the true Viterbi parse is computed. In our experiments, we used the BitPar parser (Schmid, 2004) and a PCFG which was extracted from a version of the PENN treebank that was automatically annotated with features in the style of (Klein and Manning, 2003).

2 Feature Annotation

A context-free grammar which generates empty categories has to make sure that a filler exists for each trace and vice versa. A well-known technique which enforces this constraint is the GPSG-style percolation of a slash feature: All constituents on the direct path from the trace to the filler are annotated with a special feature which represents the category of the filler as shown in figure 2. In order to restore the original treebank annotation with co-reference indices from the representation with slash features, the parse tree has to be traversed starting at a trace node and following the nodes annotated with the respective filler category until the filler node is encountered. Normally, the filler node is a sister node of an ancestor node of the trace, i.e. the filler c-commands the trace node, but in case of clausal fillers it is also possible that the filler dominates the trace. An example is the sentence *"S-1 She had – he informed her *-1 – kidney trouble"* whose parse tree is shown in figure 3.

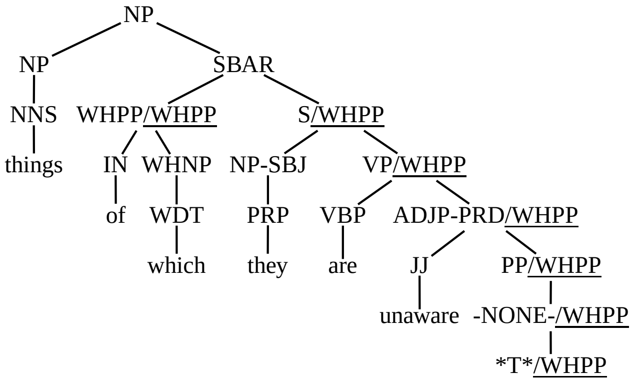


Figure 2: Slash features: The filler node of category WHNP is linked to the trace node via percolation of a slash feature. The trace node is labeled with *T*.

Besides the slash features, we used other features in order to improve the parsing accuracy of the PCFG, inspired by the work of Klein and Manning (2003). The most important ones of these features[1] will now be described in detail. Section 4.3 shows the impact of these features on labeled bracketing accuracy and empty category prediction.

VP feature VPs were annotated with a feature that distinguishes between finite, infinitive, to-infinitive, gerund, past participle, and passive VPs.

S feature The S node feature distinguishes between imperatives, finite clauses, and several types of small clauses.

Parent features Modifier categories like SBAR, PP, ADVP, RB and NP-ADV were annotated with a parent feature (cf. Johnson (1998)). The parent features distinguish between verbal (VP), adjectival (ADJP, WHADJP), adverbial (ADVP, WHADVP), nominal (NP, WHNP, QP), prepositional (PP) and other parents.

PENN tags The PENN treebank annotation uses semantic tags to refine syntactic categories. Most parsers ignore this information. We preserved the tags ADV, CLR, DIR, EXT, IMP, LGS, LOC, MNR, NOM, PRD, PRP, SBJ and TMP in combination with selected categories.

Auxiliary feature We added a feature to the part-of-speech tags of verbs in order to distinguish between *be*, *do*, *have*, and full verbs.

Agreement feature Finite VPs are marked with *3s* (*n3s*) if they are headed by a verb with part-of-speech VBZ (VBP).

Genitive feature NP nodes which dominate a node of the category POS (possessive marker) are marked with a genitive flag.

Base NPs NPs dominating a node of category NN, NNS, NNP, NNPS, DT, CD, JJ, JJR, JJS, PRP, RB, or EX are marked as base NPs.

[1] The complete annotation program is available from the author's home page at http://www.ims.uni-stuttgart.de/ schmid

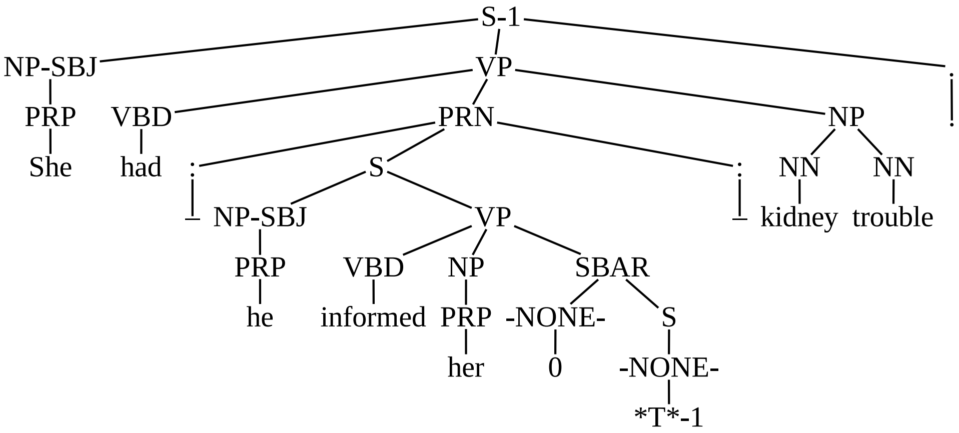


Figure 3: Example of a filler which dominates its trace

IN feature The part-of-speech tags of the 45 most frequent prepositions were lexicalized by adding the preposition as a feature. The new part-of-speech tag of the preposition “by” is “IN/by”.

Irregular adverbs The part-of-speech tags of the adverbs “as”, “so”, “about”, and “not” were also lexicalized.

Currency feature NP and QP nodes are marked with a currency flag if they dominate a node of category $, #, or SYM.

Percent feature Nodes of the category NP or QP are marked with a percent flag if they dominate the subtree (NN %). Any node which immediately dominates the token %, is marked, as well.

Punctuation feature Nodes which dominate sentential punctuation (.?!) are marked.

DT feature Nodes of category DT are split into indefinite articles (*a, an*), definite articles (*the*), and demonstratives (*this, that, those, these*).

WH feature The wh-tags (WDT, WP, WRB, WDT) of the words *which, what, who, how,* and *that* are also lexicalized.

Colon feature The part-of-speech tag ’:’ was replaced with “;”, “–” or “...” if it dominated a corresponding token.

DomV feature Nodes of a non-verbal syntactic category are marked with a feature if they dominate a node of category VP, SINV, S, SQ, SBAR, or SBARQ.

Gap feature S nodes dominating an empty NP are marked with the feature *gap*.

Subcategorization feature The part-of-speech tags of verbs are annotated with a feature which encodes the sequence of arguments. The encoding maps reflexive NPs to *r*, NP/NP-PRD/SBAR-NOM to *n*, ADJP-PRD to *j*, ADVP-PRD to *a*, PRT to *t*, PP/PP-DIR to *p*, SBAR/SBAR-CLR to *b*, S/fin to *sf*, S/ppres/gap to *sg*, S/to/gap to *st*, other S nodes to *so*, VP/ppres to *vg*, VP/ppast to *vn*, VP/pas to *vp*, VP/inf to *vi*, and other VPs to *vo*. A verb with an NP and a PP argument, for instance, is annotated with the feature *np*.

Adjectives, adverbs, and nouns may also get a subcat feature which encodes a single argument using a less fine-grained encoding which maps PP to *p*, NP to *n*, S to *s*, and SBAR to *b*. A node of category NN or NNS e.g. is marked with a subcat feature if it is followed by an argument category unless the argument is a PP which is headed by the preposition *of*.

RC feature In relative clauses with an empty relative pronoun of category WHADVP, we mark the SBAR node of the relative clause, the NP node to which it is attached, and its head child of category NN or NNS, if the head word is either *way, ways, reason, reasons, day, days, time, moment, place,* or *position*. This feature helps the parser to correctly insert WHADVP rather than WHNP. Figure 4 shows a sample tree.

TMP features Each node on the path between an NP-TMP or PP-TMP node and its nominal head is labeled with the feature *tmp*. This feature helps the parser to identify temporal NPs and PPs.

MNR and EXT features Similarly, each node on the path between an NP-EXT, NP-MNR or ADVP-TMP node and its head is labeled with the

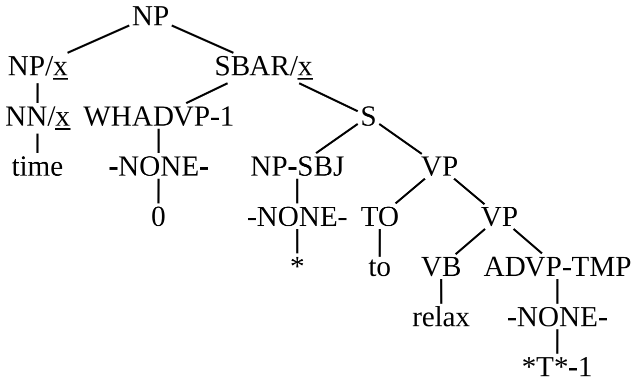


Figure 4: Annotation of relative clauses with empty relative pronoun of category WHADVP

feature *ext* or *mnr*.

ADJP features Nodes of category ADJP which are dominated by an NP node are labeled with the feature “post” if they are in final position and the feature “attr” otherwise.

JJ feature Nodes of category JJ which are dominated by an ADJP-PRD node are labeled with the feature “prd”.

JJ-tmp feature JJ nodes which are dominated by an NP-TMP node and which themselves dominate one of the words “last”, “next”, “late”, “previous”, “early”, or “past” are labeled with *tmp*.

QP feature If some node dominates an NP node followed by an NP-ADV node as in (NP (NP one dollar) (NP-ADV a day)), the first child NP node is labeled with the feature “qp”. If the parent is an NP node, it is also labeled with “qp”.

NP-pp feature NP nodes which dominate a PP node are labeled with the feature *pp*. If this PP itself is headed by the preposition *of*, then it is annotated with the feature *of*.

MWL feature In adverbial phrases which neither dominate an adverb nor another adverbial phrase, we lexicalize the part-of-speech tags of a small set of words like “least” (at least), “kind”, or “sort” which appear frequently in such adverbial phrases.

Case feature Pronouns like *he* or *him* , but not ambiguous pronouns like *it* are marked with *nom* or *acc*, respectively.

Expletives If a subject NP dominates an NP which consists of the pronoun *it*, and an S-trace in sentences like *It is important to...*, the dominated NP is marked with the feature *expl*.

LST feature The parent nodes of LST nodes[2] are marked with the feature *lst*.

Complex conjunctions In SBAR constituents starting with an IN and an NN child node (usually indicating one of the two complex conjunctions “in order to” or “in case of”), we mark the NN child with the feature *sbar*.

LGS feature The PENN treebank marks the logical subject of passive clauses which are realized by a *by*-PP with the semantic tag LGS. We move this tag to the dominating PP.

OC feature Verbs are marked with an *object control* feature if they have an NP argument which dominates an NP filler and an S argument which dominates an NP trace. An example is the sentence *She asked him to come.*

Corrections The part-of-speech tags of the PENN treebank are not always correct. Some of the errors (like the tag NNS in VP-initial position) can be identified and corrected automatically in the training data. Correcting tags did not always improve parsing accuracy, so it was done selectively.

The gap and domV features described above were also used by Klein and Manning (2003).

All features were automatically added to the PENN treebank by means of an annotation program. Figure 5 shows an example of an annotated parse tree.

3 Parameter Smoothing

We extracted the grammar from sections 2–21 of the annotated version of the PENN treebank. In order to increase the coverage of the grammar, we selectively applied markovization to the grammar (cf. Klein and Manning (2003)) by replacing long infrequent rules with a set of binary rules. Markovization was only applied if none of the non-terminals on the right hand side of the rule had a slash feature in order to avoid negative effects on the slash feature percolation mechanism.

The probabilities of the grammar rules were directly estimated with relative frequencies. No smoothing was applied, here. The lexical probabilities, on the other hand, were smoothed with

[2] LST annotates the list symbol in enumerations.

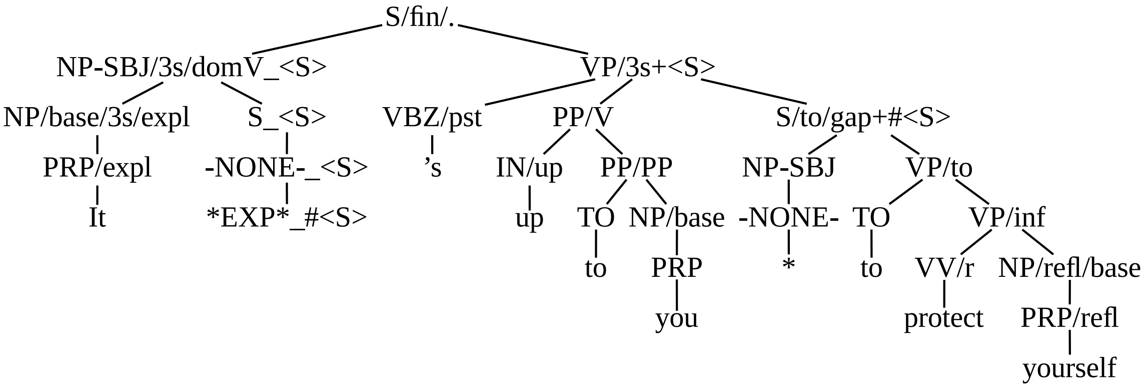


Figure 5: An Annotated Parse Tree

the following technique which was adopted from Klein and Manning (2003). Each word is assigned to one of 216 word classes. The word classes are defined with regular expressions. Examples are the class `[A-Za-z0-9-]+-old` which contains the word *20-year-old*, the class `[a-z][a-z]+ifies` which contains *clarifies*, and a class which contains a list of capitalized adjectives like *Advanced*. The word classes are ordered. If a string is matched by the regular expressions of more than one word class, then it is assigned to the first of these word classes. For each word class, we compute part-of-speech probabilities with relative frequencies. The part-of-speech frequencies $f(w,t)$ of a word w are smoothed by adding the part-of-speech probability $p(t|[w])$ of the word class $[w]$ according to equation 1 in order to obtain the smoothed frequency $\hat{f}(w,t)$. The part-of-speech probability of the word class is weighted by a parameter Θ whose value was set to 4 after testing on held-out data. The lexical probabilities are finally estimated from the smoothed frequencies according to equation 2.

$$\hat{f}(w,t) = f(w,t) + \Theta p(t|[w]) \qquad (1)$$

$$p(w|t) = \frac{\hat{f}(w,t)}{\sum_{w'} \hat{f}(w',t)} \qquad (2)$$

4 Evaluation

In our experiments, we used the usual splitting of the PENN treebank into training data (sections 2–21), held-out data (section 22), and test data (section 23).

The grammar extracted from the automatically annotated version of the training corpus contained 52,297 rules with 3,453 different non-terminals. Subtrees which dominated only empty categories were collapsed into a single empty element symbol. The parser skips over these symbols during parsing, but adds them to the output parse. Overall, there were 308 different empty element symbols in the grammar.

Parsing section 23 took 169 minutes on a Dual-Opteron system with 2.2 GHz CPUs, which is about 4.2 seconds per sentence.

	precision	recall	f-score
this paper	86.9	86.3	86.6
Klein/Manning	86.3	85.1	85.7

Table 1: Labeled bracketing accuracy on section 23

Table 1 shows the labeled bracketing accuracy of the parser on the whole section 23 and compares it to the results reported in Klein and Manning (2003) for sentences with up to 100 words.

4.1 Empty Category Prediction

Table 2 reports the accuracy of the parser in the empty category (EC) prediction task for ECs occurring more than 6 times. Following Johnson (2001), an empty category was considered correct if the treebank parse contained an empty node of the same category at the same string position. Empty SBAR nodes which dominate an empty S node are treated as a single empty element and listed as SBAR-S in table 2.

Frequent types of empty elements are recognized quite reliably. Exceptions are the traces of adverbial and prepositional phrases where the recall was only 65% and 48%, respectively, and empty relative pronouns of type WHNP and WHADVP with f-scores around 60%. A couple of empty relative pronouns of type WHADVP were mis-analyzed as WHNP which explains why the precision is higher than the recall for WHADVP, but vice versa for WHNP.

	prec.	recall	f-sc.	freq.
NP *	87.0	85.9	86.5	1607
NP *T*	84.9	87.6	86.2	508
0	95.2	89.7	92.3	416
U	95.3	93.8	94.5	388
ADVP *T*	80.3	64.7	71.7	170
S *T*	86.7	93.8	90.1	160
SBAR-S *T*	88.5	76.7	82.1	120
WHNP 0	57.6	63.6	60.4	107
WHADVP 0	75.0	50.0	60.0	36
PP *ICH*	11.1	3.4	5.3	29
PP *T*	73.7	48.3	58.3	29
SBAR *EXP*	28.6	12.5	17.4	16
VP *?*	33.3	40.0	36.4	15
S *ICH*	61.5	57.1	59.3	14
S *EXP*	66.7	71.4	69.0	14
SBAR *ICH*	60.0	25.0	35.3	12
NP *?*	50.0	9.1	15.4	11
ADJP *T*	100.0	77.8	87.5	9
SBAR-S *?*	66.7	25.0	36.4	8
VP *T*	100.0	37.5	54.5	8
overall	86.0	82.3	84.1	3716

Table 2: Accuracy of empty category prediction on section 23. The first column shows the type of the empty element and – except for empty complementizers and empty units – also the category. The last column shows the frequency in the test data.

The accuracy of the pseudo attachment labels *RNR*, *ICH*, *EXP*, and *PPA* was generally low with a precision of 41%, recall of 21%, and f-score of 28%. Empty elements with a test corpus frequency below 8 were almost never generated by the parser.

4.2 Co-Indexation

Table 3 shows the accuracy of the parser on the co-indexation task. A co-indexation of a trace and a filler is represented by a 5-tuple consisting of the category and the string position of the trace, as well as the category, start and end position of the filler. A co-indexation is judged correct if the treebank parse contains the same 5-tuple.

For NP[3] and S[4] traces of type '*T*', the co-indexation results are quite good with 85% and 92% f-score, respectively. For '*T*'-traces of other categories and for NP traces of type '*',[5] the parser shows high precision, but moderate recall. The recall of infrequent types of empty elements is again low, as in the recognition task.

	prec.	rec.	f-sc.	freq.
NP *	81.1	72.1	76.4	1140
WH NP *T*	83.7	86.8	85.2	507
S *T*	92.0	91.0	91.5	277
WH ADVP *T*	78.6	63.2	70.1	163
PP *ICH*	14.3	3.4	5.6	29
WH PP *T*	68.8	50.0	57.9	22
SBAR *EXP*	25.0	12.5	16.7	16
S *ICH*	57.1	53.3	55.2	15
S *EXP*	66.7	71.4	69.0	14
SBAR *ICH*	60.0	25.0	35.3	12
VP *T*	33.3	12.5	18.2	8
ADVP *T*	60.0	42.9	50.0	7
PP *T*	100.0	28.6	44.4	7
overall	81.7	73.5	77.4	2264

Table 3: Co-indexation accuracy on section 23. The first column shows the category and type of the trace. If the filler category of the filler is different from the category of the trace, it is added in front. The filler category is abbreviated to "WH" if the rest is identical to the trace category. The last column shows the frequency in the test data.

In order to get an impression how often EC prediction errors resulted from misplacement rather than omission, we computed EC prediction accuracies without comparing the EC positions. We observed the largest f-score increase for ADVP *T* and PP *T*, where attachment ambiguities are likely, and for VP *?* which is infrequent.

4.3 Feature Evaluation

We ran a series of evaluations on held-out data in order to determine the impact of the different features which we described in section 2 on the parsing accuracy. In each run, we deleted one of the features and measured how the accuracy changed compared to the baseline system with all features. The results are shown in table 4.

[3] NP traces of type *T* result from wh-extraction in questions and relative clauses and from fronting.

[4] S traces of type *T* occur in sentences with quoted speech like the sentence *"That's true!", he said *T*.*

[5] The trace type '*' combines two types of traces with different linguistic properties, namely empty objects of passive constructions which are co-indexed with the subject, and empty subjects of participial and infinitive clauses which are co-indexed with an NP of the matrix clause.

Feature	LB	EC	CI
slash feature	0.43	–	–
VP features	2.93	6.38	5.46
PENN tags	2.34	4.54	6.75
IN feature	2.02	2.57	5.63
S features	0.49	3.08	4.13
V subcat feature	0.68	3.17	2.94
punctuation feat.	0.82	1.11	1.86
all PENN tags	0.84	0.69	2.03
domV feature	1.76	0.15	0.00
gap feature	0.04	1.20	1.32
DT feature	0.57	0.44	0.99
RC feature	0.00	1.11	1.10
colon feature	0.41	0.84	0.44
ADV parent	0.50	0.04	0.93
auxiliary feat.	0.40	0.29	0.77
SBAR parent	0.45	0.24	0.71
agreement feat.	0.05	0.52	1.15
ADVP subcat feat.	0.33	0.32	0.55
genitive feat.	0.39	0.29	0.44
NP subcat feat.	0.33	0.08	0.76
no-tmp	0.14	0.90	0.16
base NP feat.	0.47	-0.24	0.55
tag correction	0.13	0.37	0.44
irr. adverb feat.	0.04	0.56	0.39
PP parent	0.08	0.04	0.82
ADJP features	0.14	0.41	0.33
currency feat.	0.06	0.82	0.00
qp feature	0.13	0.14	0.50
PP tmp feature	-0.24	0.65	0.60
WH feature	0.11	0.25	0.27
percent feat.	0.34	-0.10	0.10
NP-ADV parent f.	0.07	0.14	0.39
MNR feature	0.08	0.35	0.11
JJ feature	0.08	0.18	0.27
case feature	0.05	0.14	0.27
Expletive feat.	-0.01	0.16	0.27
LGS feature	0.17	0.07	0.00
ADJ subcat	0.00	0.00	0.33
OC feature	0.00	0.00	0.22
JJ-tmp feat.	0.09	0.00	0.00
refl. pronoun	0.02	-0.03	0.16
EXT feature	-0.04	0.09	0.16
MWL feature	0.05	0.00	0.00
complex conj. f.	0.07	-0.07	0.00
LST feature	0.12	-0.12	-0.11
NP-pp feature	0.13	-0.57	-0.39

Table 4: Differences between the baseline f-scores for labeled bracketing, EC prediction, and co-indexation (CI) and the f-scores without the specified feature.

5 Comparison

Table 7 compares the empty category prediction results of our parser with those reported in Johnson (2001), Dienes and Dubey (2003b) and Campbell (2004). In terms of recall and f-score, our parser outperforms the other parsers. In terms of precision, the tagger of Dienes and Dubey is the best, but its recall is the lowest of all systems.

	prec.	recall	f-score
this paper	86.0	82.3	84.1
Campbell	85.2	81.7	83.4
Dienes & Dubey	86.5	72.9	79.1
Johnson	85	74	79

Table 5: Accuracy of empty category prediction on section 23

The good performance of our parser on the empty element recognition task is remarkable considering the fact that its performance on the labeled bracketing task is 3% lower than that of the Charniak (2000) parser used by Campbell (2004).

	prec.	recall	f-score
this paper	81.7	73.5	77.4
Campbell	78.3	75.1	76.7
Dienes & Dubey (b)	81.5	68.7	74.6
Dienes & Dubey (a)	80.5	66.0	72.6
Johnson	73	63	68

Table 6: Co-indexation accuracy on section 23

Table 6 compares our co-indexation results with those reported in Johnson (2001), Dienes and Dubey (2003b), Dienes and Dubey (2003a), and Campbell (2004). Our parser achieves the highest precision and f-score. Campbell (2004) reports a higher recall, but lower precision.

Table 7 shows the trace prediction accuracies of our parser, Johnson's (2001) parser with parser input and perfect input, and Campbell's (2004) parser with perfect input. The accuracy of Johnson's parser is consistently lower than that of the other parsers and it has particular difficulties with ADVP traces, SBAR traces, and empty relative pronouns (WHNP 0). Campbell's parser and our parser cannot be directly compared, but when we take the respective performance difference to Johnson's parser as evidence, we might conclude that Campbell's parser works particularly well on NP *, *U*, and WHNP 0, whereas our system

	paper	J1	J2	C
NP *	83.2	82	91	97.5
NP *T*	86.2	81	91	96.2
0	92.3	88	96	98.5
U	94.5	92	95	98.6
ADVP *T*	71.7	56	66	79.9
S *T*	90.1	88	90	92.7
SBAR-S *T*	82.1	70	74	84.4
WHNP 0	60.4	47	77	92.4
WHADVP 0	60.0	–	–	73.3

Table 7: Comparison of the empty category prediction accuracies for different categories in this paper (paper), in (Johnson, 2001) with parser input (J1), in (Johnson, 2001) with perfect input (J2), and in (Campbell, 2004) with perfect input.

is slightly better on empty complementizers (0), ADVP traces, and SBAR traces.

6 Summary

We presented an unlexicalized PCFG parser which applies a slash feature percolation mechanism to generate parse trees with empty elements and co-indexation of traces and fillers. The grammar was extracted from a version of the PENN treebank which was annotated with slash features and a set of other features that were added in order to improve the general parsing accuracy. The parser computes true Viterbi parses unlike most other parsers for treebank grammars which are not guaranteed to produce the most likely parse tree because they apply pruning strategies like beam search.

We evaluated the parser using the standard PENN treebank training and test data. The labeled bracketing f-score of 86.6% is – to our knowledge – the best f-score reported for unlexicalized PCFGs, exceeding that of Klein and Manning (2003) by almost 1%. On the empty category prediction task, our parser outperforms the best previously reported system (Campbell, 2004) by 0.7% reaching an f-score of 84.1%, although the general parsing accuracy of our unlexicalized parser is 3% lower than that of the parser used by Campbell (2004). Our parser also ranks highest in terms of the co-indexation accuracy with 77.4% f-score, again outperforming the system of Campbell (2004) by 0.7%.

References

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the 42nd Annual Meeting of the ACL*, pages 645–652, Barcelona, Spain.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (ANLP-NAACL 2000)*, pages 132–139, Seattle, Washington.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the ACL*, Madrid, Spain.

Péter Dienes and Amit Dubey. 2003a. Antecedent recovery: Experiments with a trace tagger. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, Sapporo, Japan.

Péter Dienes and Amit Dubey. 2003b. Deep syntactic processing by combining shallow methods. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 431–438, Sapporo, Japan.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Mark Johnson. 2001. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 39th Annual Meeting of the ACL*, pages 136–143, Toulouse, France.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the ACL*, pages 423–430, Sapporo, Japan.

Roger Levy and Christopher D. Manning. 2004. Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. In *Proceedings of the 42nd Annual Meeting of the ACL*, pages 327–334, Barcelona, Spain.

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, June.

Helmut Schmid. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, volume 1, pages 162–168, Geneva, Switzerland.

Learning More Effective Dialogue Strategies Using Limited Dialogue Move Features

Matthew Frampton and **Oliver Lemon**
HCRC, School of Informatics
University of Edinburgh
Edinburgh, EH8 9LW, UK
M.J.E.Frampton@sms.ed.ac.uk, olemon@inf.ed.ac.uk

Abstract

We explore the use of restricted dialogue contexts in reinforcement learning (RL) of effective dialogue strategies for information seeking spoken dialogue systems (e.g. COMMUNICATOR (Walker et al., 2001)). The contexts we use are richer than previous research in this area, e.g. (Levin and Pieraccini, 1997; Scheffler and Young, 2001; Singh et al., 2002; Pietquin, 2004), which use only slot-based information, but are much less complex than the full dialogue "Information States" explored in (Henderson et al., 2005), for which tractabe learning is an issue. We explore how incrementally adding richer features allows learning of more effective dialogue strategies. We use 2 user simulations learned from COMMUNICATOR data (Walker et al., 2001; Georgila et al., 2005b) to explore the effects of different features on learned dialogue strategies. Our results show that adding the dialogue moves of the last system and user turns increases the average reward of the automatically learned strategies by 65.9% over the original (hand-coded) COMMUNICATOR systems, and by 7.8% over a baseline RL policy that uses only slot-status features. We show that the learned strategies exhibit an emergent "focus switching" strategy and effective use of the 'give help' action.

1 Introduction

Reinforcement Learning (RL) applied to the problem of dialogue management attempts to find optimal mappings from dialogue contexts to system actions. The idea of using *Markov Decision Processes (MDPs)* and reinforcement learning to design dialogue strategies for dialogue systems was first proposed by (Levin and Pieraccini, 1997). There, and in subsequent work such as (Singh et al., 2002; Pietquin, 2004; Scheffler and Young, 2001), only very limited state information was used in strategy learning, based always on the number and status of filled information slots in the application (e.g. departure-city is filled, destination-city is unfilled). This we refer to as *low-level* contextual information. Much prior work (Singh et al., 2002) concentrated only on specific strategy decisions (e.g. confirmation and initiative strategies), rather than the full problem of what system dialogue move to take next.

The simple strategies learned for low-level definitions of state cannot be sensitive to (sometimes critical) aspects of the dialogue context, such as the user's last dialogue move (DM) (e.g. request-help) unless that move directly affects the status of an information slot (e.g. *provide-info(destination-city)*). We refer to additional contextual information such as the system and user's last dialogue moves as *high-level* contextual information. (Frampton and Lemon, 2005) learned full strategies with limited 'high-level' information (i.e. the dialogue move(s) of the last user utterance) and only used a stochastic user simulation whose probabilities were supplied via common-sense and intuition, rather than learned from data. This paper uses data-driven n-gram user simulations (Georgila et al., 2005a) and a richer dialogue context.

On the other hand, increasing the size of the state space for RL has the danger of making the learning problem intractable, and at the very least means that data is more sparse and state approximation methods may need to be used (Henderson et al., 2005). To date, the use of very large state spaces relies on a "hybrid" supervised/reinforcement learning technique, where the reinforcement learning element has not yet been shown to significantly improve policies over the purely supervised case (Henderson et al., 2005).

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 185–192,
Sydney, July 2006.

The extended state spaces that we propose are based on theories of dialogue such as (Clark, 1996; Searle, 1969; Austin, 1962; Larsson and Traum, 2000), where which actions a dialogue participant can or should take next are not based solely on the task-state (i.e. in our domain, which slots are filled), but also on wider contextual factors such as a user's dialogue moves or speech acts. In future work we also intend to use feature selection techniques (e.g. correlation-based feature subset (CFS) evaluation (Rieser and Lemon, 2006)) on the COMMUNICATOR data (Georgila et al., 2005a; Walker et al., 2001) in order to identify additional context features that it may be effective to represent in the state.

1.1 Methodology

To explore these issues we have developed a Reinforcement Learning (RL) program to learn dialogue strategies while accurate simulated users (Georgila et al., 2005a) converse with a dialogue manager. See (Singh et al., 2002; Scheffler and Young, 2001) and (Sutton and Barto, 1998) for a detailed description of Markov Decision Processes and the relevant RL algorithms.

In dialogue management we are faced with the problem of deciding which dialogue actions it is best to perform in different states. We use (RL) because it is a method of learning by *delayed* reward using *trial-and-error search*. These two properties appear to make RL techniques a good fit with the problem of automatically optimising dialogue strategies, because in task-oriented dialogue often the "reward" of the dialogue (e.g. successfully booking a flight) is not obtainable immediately, and the large space of possible dialogues for any task makes some degree of trial-and-error exploration necessary.

We use both 4-gram and 5-gram user simulations for testing and for training (i.e. train with 4-gram, test with 5-gram, and vice-versa). These simulations also simulate ASR errors since the probabilities are learned from recognition hypotheses and system behaviour logged in the COMMUNICATOR data (Walker et al., 2001) further annotated with speech acts and contexts by (Georgila et al., 2005b). Here the task domain is flight-booking, and the aim for the dialogue manager is to obtain values for the user's flight information "slots" i.e. *departure city*, *destination city*, *departure date* and *departure time*, before making a database query. We add the *dialogue moves of the last user and system turns* as context features and use these in strategy learning. We compare the learned strategies to 2 baselines: the original COMMUNICATOR systems and an RL strategy which uses only slot status features.

1.2 Outline

Section 2 contains a description of our basic experimental framework, and a detailed description of the reinforcement learning component and user simulations. Sections 3 and 4 describe the experiments and analyse our results, and in section 5 we conclude and suggest future work.

2 The Experimental Framework

Each experiment is executed using the DIPPER Information State Update dialogue manager (Bos et al., 2003) (which here is used to track and update dialogue context rather than deciding which actions to take), a Reinforcement Learning program (which determines the next dialogue action to take), and various user simulations. In sections 2.3 and 2.4 we give more details about the reinforcement learner and user simulations.

2.1 The action set for the learner

Below is a list of all the different actions that the RL dialogue manager can take and must learn to choose between based on the context:

1. An open question e.g. 'How may I help you?'
2. Ask the value for any one of slots $1...n$.
3. Explicitly confirm any one of slots $1...n$.
4. Ask for the n^{th} slot whilst implicitly confirming[1] either slot value $n-1$ e.g. 'So you want to fly from London to where?', or slot value $n+1$
5. Give help.
6. Pass to human operator.
7. Database Query.

There are a couple of restrictions regarding which actions can be taken in which states: an open question is only available at the start of the dialogue, and the dialogue manager can only try to confirm non-empty slots.

2.2 The Reward Function

We employ an "all-or-nothing" reward function which is as follows:

1. Database query, all slots confirmed: $+100$
2. Any other database query: -75

[1] Where $n = 1$ we implicitly confirm the final slot and where $n = 4$ we implicitly confirm the first slot. This action set does not include actions that ask the n^{th} slot whilst implicitly confirming slot value $n - 2$. These will be added in future experiments as we continue to increase the action and state space.

3. User simulation hangs-up: -100

4. DIPPER passes to a human operator: -50

5. Each system turn: -5

To maximise the chances of a slot value being correct, it must be confirmed rather than just filled. The reward function reflects the fact that a successful dialogue manager must maximise its chances of getting the slots correct i.e. they must all be confirmed. (Walker et al., 2000) showed with the PARADISE evaluation that confirming slots increases user satisfaction.

The maximum reward that can be obtained for a single dialogue is 85, (the dialogue manager prompts the user, the user replies by filling all four of the slots in a single utterance, and the dialogue manager then confirms all four slots and submits a database query).

2.3 The Reinforcement Learner's Parameters

When the reinforcement learner agent is initialized, it is given a parameter string which includes the following:

1. Step Parameter: $\alpha = decreasing$

2. Discount Factor: $\gamma = 1$

3. Action Selection Type = *softmax* (alternative is ϵ-*greedy*)

4. Action Selection Parameter: *temperature* $=$ 15

5. Eligibility Trace Parameter: $\lambda = 0.9$

6. Eligibility Trace = *replacing* (alternative is *accumulating*)

7. Initial *Q-values* = 25

The reinforcement learner updates its Q-values using the *Sarsa(λ)* algorithm (see (Sutton and Barto, 1998)). The first parameter is the step-parameter α which may be a value between 0 and 1, or specified as *decreasing*. If it is *decreasing*, as it is in our experiments, then for any given Q-value update α is $\frac{1}{k}$ where k is the number of times that the state-action pair for which the update is being performed has been visited. This kind of step parameter will ensure that given a sufficient number of training dialogues, each of the Q-values will eventually converge. The second parameter (discount factor) γ may take a value between 0 and 1. For the dialogue management problem we set it to 1 so that future rewards are taken into account as strongly as possible.

Apart from updating Q-values, the reinforcement learner must also choose the next action for the dialogue manager and the third parameter specifies whether it does this by ϵ-*greedy* or *softmax* action selection (here we have used softmax).

The fifth parameter, the eligibility trace parameter λ, may take a value between 0 and 1, and the sixth parameter specifies whether the eligibility traces are *replacing* or *accumulating*. We used *replacing* traces because they produced faster learning for the slot-filling task. The seventh parameter is for supplying the initial Q-values.

2.4 N-Gram User Simulations

Here *user simulations*, rather than real users, interact with the dialogue system during learning. This is because thousands of dialogues may be necessary to train even a simple system (here we train on up to 50000 dialogues), and for a proper exploration of the state-action space the system should sometimes take actions that are not optimal for the current situation, making it a sadistic and time-consuming procedure for any human training the system. (Eckert et al., 1997) were the first to use a user simulation for this purpose, but it was not goal-directed and so could produce inconsistent utterances. The later simulations of (Pietquin, 2004) and (Scheffler and Young, 2001) were to some extent "goal-directed" and also incorporated an *ASR error simulation*. The user simulations interact with the system via *intentions*. Intentions are preferred because they are easier to generate than word sequences and because they allow error modelling of all parts of the system, for example *ASR error modelling* and semantic errors. The user and ASR simulations must be realistic if the learned strategy is to be directly applicable in a real system.

The n-gram user simulations used here (see (Georgila et al., 2005a) for details and evaluation results) treat a dialogue as a sequence of pairs of speech acts and tasks. They take as input the $n-1$ most recent speech act-task pairs in the dialogue history, and based on n-gram probabilities learned from the COMMUNICATOR data (automatically annotated with speech acts and Information States (Georgila et al., 2005b)), they then output a user utterance as a further speech-act task pair. These user simulations incorporate the effects of ASR errors since they are built from the user utterances as they were recognized by the ASR components of the original COMMUNICATOR systems. Note that the user simulations do not provide instantiated slot values e.g. a response to provide a destination city is the speech-act task pair *"[provide info] [dest city]"*. We cannot assume that two such responses in the same dialogue refer to the same destination cities. Hence in the dialogue manager's Information State where we record whether a slot is *empty*, *filled*, or *confirmed*, we only update from *filled* to *confirmed* when the slot value is implicitly or explicitly confirmed. An additional function maps the user speech-act task pairs to a form that can be interpreted by the dialogue manager. Post-mapping user responses are made up of one or more of the following types of utterance: (1) Stay quiet, (2) Provide 1 or more slot values, (3) Yes, (4) No, (5) Ask for help, (6) Hang-up, (7) Null (out-of-domain or no ASR hypothesis).

The quality of the 4 and 5-gram user simulations has been established through a variety of metrics and against the behaviour of the actual users of the COMMUNICATOR systems, see (Georgila et al., 2005a).

2.4.1 Limitations of the user simulations

The user and ASR simulations are a fundamentally important factor in determining the nature of the learned strategies. For this reason we should note the limitations of the n-gram simulations used here. A first limitation is that we cannot be sure that the COMMUNICATOR training data is sufficiently complete, and a second is that the n-gram simulations only use a window of n moves in the dialogue history. This second limitation becomes a problem when the user simulation's current move ought to take into account something that occurred at an earlier stage in the dialogue. This might result in the user simulation repeating a slot value unnecessarily, or the chance of an ASR error for a particular word being independent of whether the same word was previously recognised correctly. The latter case means we cannot simulate for example, a particular slot value always being liable to misrecognition. These limitations will affect the nature of the learned strategies. Different state features may assume more or less importance than they would if the simulations were more realistic. This is a point that we will return to in the analysis of the experimental results. In future work we will use the more accurate user simulations recently developed following (Georgila et al., 2005a) and we expect that these will improve our results still further.

3 Experiments

First we learned strategies with the 4-gram user simulation and tested with the 5-gram simulation, and then did the reverse. We experimented with different feature sets, exploring whether better strategies could be learned by adding limited context features. We used two baselines for comparison:

- The performance of the original COMMUNICATOR systems in the data set (Walker et al., 2001).
- An **RL baseline** dialogue manager learned using only slot-status features i.e. for each of slots $1-4$, is the slot *empty*, *filled* or *confirmed*?

We then learned two further strategies:

- **Strategy 2 (UDM)** was learned by adding the user's last dialogue move to the state.
- **Strategy 3 (USDM)** was learned by adding both the user and system's last dialogue moves to the state.

The possible system and user dialogue moves were those given in sections 2.1 and 2.4 respectively, and the reward function was that described in section 2.2.

3.1 The COMMUNICATOR data baseline

We computed the scores for the original handcoded COMMUNICATOR systems as was done by (Henderson et al., 2005), and we call this the "HLG05" score. This scoring function is based on task completion and dialogue length rewards as determined by the PARADISE evaluation (Walker et al., 2000). This function gives 25 points for each slot which is filled, another 25 for each that is confirmed, and deducts 1 point for each system action. In this case the maximum possible score is 197 i.e. 200 minus 3 actions, (the system prompts the user, the user replies by filling all four of the slots in one turn, and the system then confirms all four slots and offers the flight). The average score for the 1242 dialogues in the COMMUNICATOR dataset where the aim was to fill and confirm only the same four slots as we have used here was 115.26. The other COMMUNICATOR dialogues involved different slots relating to return flights, hotel-bookings and car-rentals.

4 Results

Figure 1 tracks the improvement of the 3 learned strategies for 50000 training dialogues with the 4-gram user simulation, and figure 2 for 50000 training dialogues with the 5-gram simulation. They show the average reward (according to the function of section 2.2) obtained by each strategy over intervals of 1000 training dialogues.

Table 1 shows the results for testing the strategies learned after 50000 training dialogues (the baseline RL strategy, strategy 2 (UDM) and strategy 3 (USDM)). The 'a' strategies were trained with the 4-gram user simulation and tested with

	Features	Av. Score	HLG05	Filled Slots	Conf. Slots	Length
$4 \rightarrow 5$ **gram = (a)**						
RL Baseline (a)	Slots status	51.67	190.32	100	100	-9.68
RL Strat 2, UDM (a)	+ Last User DM	53.65**	190.67	100	100	-9.33
RL Strat 3, USDM (a)	+ Last System DM	54.9**	190.98	100	100	-9.02
$5 \rightarrow 4$ **gram = (b)**						
RL Baseline (b)	Slots status	51.4	190.28	100	100	-9.72
RL Strat 2, UDM (b)	+ Last User DM	54.46*	190.83	100	100	-9.17
RL Strat 3, USDM (b)	+ Last System DM	**56.24****	**191.25**	100	100	-8.75
RL Baseline (av)	Slots status	51.54	190.3	100	100	-9.7
RL Strat 2, UDM (av)	+ Last User DM	54.06**	190.75	100	100	-9.25
RL Strat 3, USDM (av)	+ Last System DM	**55.57****	**191.16**	100	100	-8.84
COMM Systems			115.26	84.6	63.7	-33.1
Hybrid RL ***	Information States		142.6	88.1	70.9	-16.4

Table 1: Testing the learned strategies after 50000 training dialogues, average reward achieved per dialogue over 1000 test dialogues. (a) = strategy trained using 4-gram and tested with 5-gram; (b) = strategy trained with 5-gram and tested with 4-gram; (av) = average; * significance level $p < 0.025$; ** significance level $p < 0.005$; *** Note: The Hybrid RL scores (here updated from (Henderson et al., 2005)) are not directly comparable since that system has a larger action set and fewer policy constraints.

the 5-gram, while the 'b' strategies were trained with the 5-gram user simulation and tested with the 4-gram. The table also shows average scores for the strategies. Column 2 contains the average reward obtained per dialogue by each strategy over 1000 test dialogues (computed using the function of section 2.2).

The 1000 test dialogues for each strategy were divided into 10 sets of 100. We carried out t-tests and found that in both the 'a' and 'b' cases, strategy 2 (UDM) performs significantly better than the RL baseline (significance levels $p < 0.005$ and $p < 0.025$), and strategy 3 (USDM) performs significantly better than strategy 2 (UDM) (significance level $p < 0.005$). With respect to average performance, strategy 2 (UDM) improves over the RL baseline by 4.9%, and strategy 3 (USDM) improves by 7.8%. Although there seem to be only negligible qualitative differences between strategies 2(b) and 3(b) and their 'a' equivalents, the former perform slightly better in testing. This suggests that the 4-gram simulation used for testing the 'b' strategies is a little more reliable in filling and confirming slot values than the 5-gram.

The 3rd column "HLG05" shows the average scores for the dialogues as computed by the reward function of (Henderson et al., 2005). This is done for comparison with that work but also with the COMMUNICATOR data baseline. Using the HLG05 reward function, strategy 3 (USDM) improves over the original COMMUNICATOR systems baseline by 65.9%. The components making up the reward are shown in the final 3 columns of table 1. Here we see that all of the RL strategies are able to fill and confirm all of the 4 slots when conversing with the simulated COMMUNICATOR users. The only variation is in the average length of dialogue required to confirm all four slots. The COMMUNICATOR systems were often unable to confirm or fill all of the user slots, and the dialogues were quite long on average. As stated in section 2.4.1, the n-gram simulations do not simulate the case of a particular user goal utterance being unrecognisable for the system. This was a problem that could be encountered by the real COMMUNICATOR systems.

Nevertheless, the performance of all the learned strategies compares very well to the COMMUNICATOR data baseline. For example, in an average dialogue, the RL strategies filled and confirmed all four slots with around 9 actions not including offering the flight, but the COMMUNICATOR systems took an average of around 33 actions per dialogue, and often failed to complete the task.

With respect to the hybrid RL result of (Henderson et al., 2005), shown in the final row of the table, Strategy 3 (USDM) shows a 34% improvement, though these results are not directly comparable because that system uses a larger action set and has fewer constraints (e.g. it can ask "how may I help you?" at any time, not just at the start of a dialogue).

Finally, let us note that the performance of the RL strategies is close to optimal, but that there is some room for improvement. With respect to the HLG05 metric, the optimal system score would be 197, but this would only be available in rare cases where the simulated user supplies all 4 slots in the

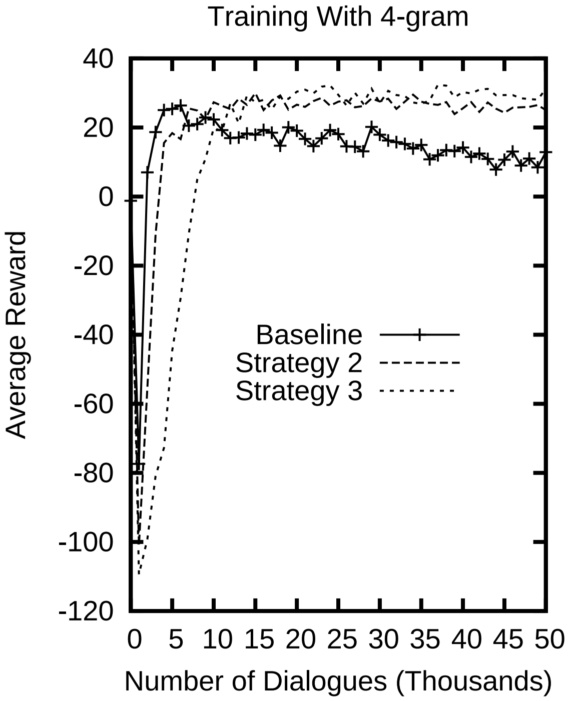


Figure 1: Training the dialogue strategies with the 4-gram user simulation

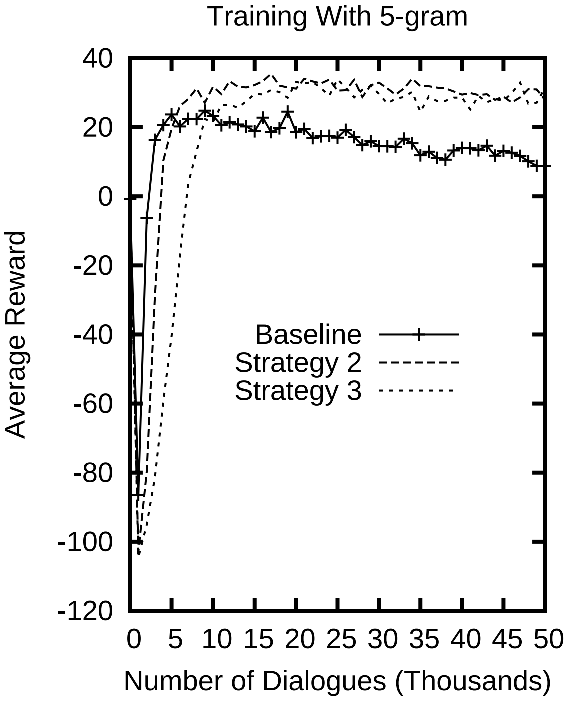


Figure 2: Training the dialogue strategies with the 5-gram user simulation

first utterance. With respect to the metric we have used here (with a -5 per system turn penalty), the optimal score is 85 (and we currently score an average of 55.57). Thus we expect that there are still further improvments that can be made to more fully exploit the dialogue context (see section 4.3).

4.1 Qualitative Analysis

Below are a list of general characteristics of the learned strategies:

1. The reinforcement learner learns to query the database only in states where all four slots have been confirmed.

2. With sufficient exploration, the reinforcement learner learns not to pass the call to a human operator in any state.

3. The learned strategies employ implicit confirmations wherever possible. This allows them to fill and confirm the slots in fewer turns than if they simply asked the slot values and then used explicit confirmation.

4. As a result of characteristic 3, which slots can be asked and implicitly confirmed at the same time influences the order in which the learned strategies attempt to fill and confirm each slot, e.g. if the status of the third slot is 'filled' and the others are 'empty', the learner learns to ask for the second or fourth slot rather than the first, since it can implicitly confirm the third while it asks for the second or fourth slots, but it cannot implicitly confirm the third while it asks for the first slot. This action is not available (see section 2.1).

4.2 Emergent behaviour

In testing the UDM strategy (2) filled and confirmed all of the slots in fewer turns on average than the RL baseline, and strategy 3 (USDM) did this in fewer turns than strategy 2 (UDM). What then were the qualitative differences between the three strategies? The behaviour of the three strategies only seems to really deviate when a user response fails to fill or confirm one or more slots. Then the baseline strategy's state has not changed and so it will repeat its last dialogue move, whereas the state for strategies 2 (UDM) and 3 (USDM) has changed and as a result, these may now try different actions. It is in such circumstances that the UDM strategy seems to be more effective than the baseline, and strategy 3 (USDM) more effective than the UDM strategy. In figure 3 we show illustrative state and learned action pairs for the different strategies. They relate to a situation where the first user response(s) in the dialogue has/have failed to fill a single slot value. NB: here 'emp' stands for 'empty' and 'fill' for 'filled' and they appear in the first four state variables, which stand for slot states. For strategy 2 (UDM), the fifth variable represents the user's last dialogue move, and the for strategy 3 (USDM), the fifth variable represents the system's last dialogue move, and the sixth, the user's last dialogue move.

```
BASELINE STRATEGY
State:
[emp,emp,emp,emp]
Action: askSlot2


STRATEGY 2 (UDM)
State:
[emp,emp,emp,emp,user(quiet)]
Action: askSlot3

State:
[emp,emp,emp,emp,user(null)]
Action: askSlot1


STRATEGY 3 (USDM)
State:
[emp,emp,emp,emp,askSlot3,user(quiet)]
Action: askSlot3

State:
[emp,emp,emp,emp,askSlot3,user(null)]
Action: giveHelp

State:
[emp,emp,emp,emp,giveHelp,user(quiet)]
Action: askSlot3

State:
[emp,emp,emp,emp,giveHelp,user(null)]
Action: askSlot3
```

Figure 3: Examples of the different learned strategies and emergent behaviours: focus switching (for UDM) and giving help (for USDM)

Here we can see that should the user responses continue to fail to provide a slot value, the baseline's state will be unchanged and so the strategy will simply ask for slot 2 again. The state for strategy 2 (UDM) does change however. This strategy switches focus between slots 3 and 1 depending on whether the user's last dialogue move was 'null' or 'quiet' NB. As stated in section 2.4, 'null' means out-of-domain or that there was no ASR hypothesis. Strategy 3 (USDM) is different again. Knowledge of the system's last dialogue move as well as the user's last move has enabled the learner to make effective use of the 'give help' action, rather than to rely on switching focus. When the user's last dialogue move is 'null' in response to the system move 'askSlot3', then the strategy uses the 'give help' action before returning to ask for slot 3 again. The example described here is not the only example of strategy 2 (UDM) employing focus switching while strategy 3 (USDM) prefers to use the 'give help' action when a user response fails to fill or confirm a slot. This kind of behaviour in strategies 2 and 3 is emergent dialogue behaviour that has been learned by the system rather than explicitly programmed.

4.3 Further possibilities for improvement over the RL baseline

Further improvements over the RL baseline might be possible with a wider set of system actions. Strategies 2 and 3 may learn to make more effective use of additional actions than the baseline e.g. additional actions that implicitly confirm one slot whilst asking another may allow more of the switching focus described in section 4.1. Other possible additional actions include actions that ask for or confirm two or more slots simultaneously.

In section 2.4.1, we highlighted the fact that the n-gram user simulations are not completely realistic and that this will make certain state features more or less important in learning a strategy. Thus had we been able to use even more realistic user simulations, including certain additional context features in the state might have enabled a greater improvement over the baseline. Dialogue length is an example of a feature that could have made a difference had the simulations been able to simulate the case of a particular goal utterance being unrecognisable for the system. The reinforcement learner may then be able to use the dialogue length feature to learn when to give up asking for a particular slot value and make a partially complete database query. This would of course require a reward function that gave some reward to partially complete database queries rather than the all-or-nothing reward function used here.

5 Conclusion and Future Work

We have used user simulations that are n-gram models learned from COMMUNICATOR data to explore reinforcement learning of full dialogue strategies with some "high-level" context information (the user and and system's last dialogue moves). Almost all previous work (e.g. (Singh et al., 2002; Pietquin, 2004; Scheffler and Young, 2001)) has included only low-level information in state representations. In contrast, the exploration of very large state spaces to date relies on a "hybrid" supervised/reinforcement learning technique, where the reinforcement learning element has not been shown to significantly improve policies over the purely supervised case (Henderson et al., 2005).

We presented our experimental environment, the reinforcement learner, the simulated users, and our methodology. In testing with the simulated COMMUNICATOR users, the new strategies learned with higher-level (i.e. dialogue move) information in the state outperformed the low-level RL baseline (only slot status information) by 7.8% and the original COMMUNICATOR systems by 65.9%. These strategies obtained more reward than the RL baseline by filling and confirming all of the slots with fewer system turns on average. Moreover, the learned strategies show interesting *emergent* dialogue behaviour such as making effective use of the 'give help' action and *switching focus* to different subtasks when the current subtask is proving problematic.

In future work, we plan to use even more realistic user simulations, for example those developed following (Georgila et al., 2005a), which incorporate elements of goal-directed user behaviour. We will continue to investigate whether we can maintain tractability and learn superior strategies as we add incrementally more high-level contextual information to the state. At some stage this may necessitate using a generalisation method such as linear function approximation (Henderson et al., 2005). We also intend to use feature selection techniques (e.g. CFS subset evaluation (Rieser and Lemon, 2006)) on in order to determine which contextual features this suggests are important. We will also carry out a more direct comparison with the hybrid strategies learned by (Henderson et al., 2005). In the slightly longer term, we will test our learned strategies on humans using a full spoken dialogue system. We hypothesize that the strategies which perform the best in terms of task completion and user satisfaction scores (Walker et al., 2000) will be those learned with high-level dialogue context information in the state.

Acknowledgements

This work is supported by the ESRC and the TALK project, `www.talk-project.org`.

References

John L. Austin. 1962. *How To Do Things With Words*. Oxford University Press.

Johan Bos, Ewan Klein, Oliver Lemon, and Tetsushi Oka. 2003. Dipper: Description and formalisation of an information-state update dialogue system architecture. In *4th SIGdial Workshop on Discourse and Dialogue*, Sapporo.

Herbert H. Clark. 1996. *Using Language*. Cambridge University Press.

Weiland Eckert, Esther Levin, and Roberto Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *IEEE Workshop on Automatic Speech Recognition and Understanding*.

Matthew Frampton and Oliver Lemon. 2005. Reinforcement Learning Of Dialogue Strategies Using The User's Last Dialogue Act. In *IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems*.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2005a. Learning User Simulations for Information State Update Dialogue Systems. In *Interspeech/Eurospeech: the 9th biennial conference of the International Speech Communication Association*.

Kallirroi Georgila, Oliver Lemon, and James Henderson. 2005b. Automatic annotation of COMMUNICATOR dialogue data for learning dialogue strategies and user simulations. In *Ninth Workshop on the Semantics and Pragmatics of Dialogue (SEMDIAL: DIALOR)*.

James Henderson, Oliver Lemon, and Kallirroi Georgila. 2005. Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data. In *IJCAI workshop on Knowledge and Reasoning in Practical Dialogue Systems,*.

Staffan Larsson and David Traum. 2000. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. *Natural Language Engineering*, 6(3-4):323–340.

Esther Levin and Roberto Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogue strategies. In *Eurospeech*, Rhodes,Greece.

Olivier Pietquin. 2004. *A Framework for Unsupervised Learning of Dialogue Strategies*. Presses Universitaires de Louvain, SIMILAR Collection.

Verena Rieser and Oliver Lemon. 2006. Using machine learning to explore human multimodal clarification strategies. In *Proc. ACL*.

Konrad Scheffler and Steve Young. 2001. Corpus-based dialogue simulation for automatic strategy learning and evaluation. In *NAACL-2001 Workshop on Adaptation in Dialogue Systems*, Pittsburgh, USA.

John R. Searle. 1969. *Speech Acts*. Cambridge University Press.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research (JAIR)*.

Richard Sutton and Andrew Barto. 1998. *Reinforcement Learning*. MIT Press.

Marilyn A. Walker, Candace A. Kamm, and Diane J. Litman. 2000. Towards Developing General Models of Usability with PARADISE. *Natural Language Engineering*, 6(3).

Marilyn A. Walker, Rebecca J. Passonneau, and Julie E. Boland. 2001. Quantitative and Qualitative Evaluation of Darpa Communicator Spoken Dialogue Systems. In *Meeting of the Association for Computational Linguistics*, pages 515–522.

Dependencies between Student State and Speech Recognition Problems in Spoken Tutoring Dialogues

Mihai Rotaru
University of Pittsburgh
Pittsburgh, USA
mrotaru@cs.pitt.edu

Diane J. Litman
University of Pittsburgh
Pittsburgh, USA
litman@cs.pitt.edu

Abstract

Speech recognition problems are a reality in current spoken dialogue systems. In order to better understand these phenomena, we study dependencies between speech recognition problems and several higher level dialogue factors that define our notion of student state: frustration/anger, certainty and correctness. We apply Chi Square (χ2) analysis to a corpus of speech-based computer tutoring dialogues to discover these dependencies both within and across turns. Significant dependencies are combined to produce interesting insights regarding speech recognition problems and to propose new strategies for handling these problems. We also find that tutoring, as a new domain for speech applications, exhibits interesting tradeoffs and new factors to consider for spoken dialogue design.

1 Introduction

Designing a spoken dialogue system involves many non-trivial decisions. One factor that the designer has to take into account is the presence of speech recognition problems (**SRP**). Previous work (Walker et al., 2000) has shown that the number of SRP is negatively correlated with overall user satisfaction. Given the negative impact of SRP, there has been a lot of work in trying to understand this phenomenon and its implications for building dialogue systems. Most of the previous work has focused on lower level details of SRP: identifying components responsible for SRP (acoustic model, language model, search algorithm (Chase, 1997)) or prosodic characterization of SRP (Hirschberg et al., 2004).

We extend previous work by analyzing the relationship between SRP and higher level dialogue factors. Recent work has shown that dialogue design can benefit from several higher level dialogue factors: dialogue acts (Frampton and Lemon, 2005; Walker et al., 2001), pragmatic plausibility (Gabsdil and Lemon, 2004). Also, it is widely believed that user emotions, as another example of higher level factor, interact with SRP but, currently, there is little hard evidence to support this intuition. We perform our analysis on three high level dialogue factors: frustration/anger, certainty and correctness. Frustration and anger have been observed as the most frequent emotional class in many dialogue systems (Ang et al., 2002) and are associated with a higher word error rate (Bulyko et al., 2005). For this reason, we use the presence of emotions like *frustration* and *anger* as our first dialogue factor.

Our other two factors are inspired by another contribution of our study: looking at *speech-based computer tutoring* dialogues instead of more commonly used information retrieval dialogues. Implementing spoken dialogue systems in a new domain has shown that many practices do not port well to the new domain (e.g. confirmation of long prompts (Kearns et al., 2002)). Tutoring, as a new domain for speech applications (Litman and Forbes-Riley, 2004; Pon-Barry et al., 2004), brings forward new factors that can be important for spoken dialogue design. Here we focus on *certainty* and *correctness*. Both factors have been shown to play an important role in the tutoring process (Forbes-Riley and Litman, 2005; Liscombe et al., 2005).

A common practice in previous work on emotion prediction (Ang et al., 2002; Litman and Forbes-Riley, 2004) is to transform an initial finer level emotion annotation (five or more labels) into a coarser level annotation (2-3 labels). We wanted to understand if this practice can im-

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 193–200,
Sydney, July 2006. ©2006 Association for Computational Linguistics

pact the dependencies we observe from the data. To test this, we combine our two emotion[1] factors (frustration/anger and certainty) into a binary emotional/non-emotional annotation.

To understand the relationship between SRP and our three factors, we take a three-step approach. In the first step, dependencies between SRP and our three factors are discovered using the Chi Square (χ2) test. Similar analyses on human-human dialogues have yielded interesting insights about human-human conversations (Forbes-Riley and Litman, 2005; Skantze, 2005). In the second step, significant dependencies are combined to produce interesting *insights* regarding SRP and to propose *strategies* for handling SRP. Validating these strategies is the purpose of the third step. In this paper, we focus on the first two steps; the third step is left as future work.

Our analysis produces several interesting insights and strategies which confirm the utility of the proposed approach. With respect to insights, we show that user emotions interact with SRP. We also find that incorrect/uncertain student turns have more SRP than expected. In addition, we find that the emotion annotation level affects the interactions we observe from the data, with finer-level emotions yielding more interactions and insights.

In terms of strategies, our data suggests that favoring misrecognitions over rejections (by lowering the rejection threshold) might be more beneficial for our tutoring task – at least in terms of reducing the number of emotional student turns. Also, as a general design practice in the spoken tutoring applications, we find an interesting tradeoff between the pedagogical value of asking difficult questions and the system's ability to recognize the student answer.

2 Corpus

The corpus analyzed in this paper consists of 95 experimentally obtained spoken tutoring dialogues between 20 students and our system ITSPOKE (Litman and Forbes-Riley, 2004), a speech-enabled version of the text-based WHY2 conceptual physics tutoring system (VanLehn et al., 2002). When interacting with ITSPOKE, students first type an essay answering a qualitative physics problem using a graphical user interface. ITSPOKE then engages the student in spoken dialogue (using speech-based input and output) to correct misconceptions and elicit more complete explanations, after which the student revises the essay, thereby ending the tutoring or causing another round of tutoring/essay revision. For recognition, we use the Sphinx2 speech recognizer with stochastic language models. Because speech recognition is imperfect, after the data was collected, each student utterance in our corpus was manually transcribed by a project staff member. An annotated excerpt from our corpus is shown in Figure 1 (punctuation added for clarity). The excerpts show both what the student said (the STD labels) and what ITSPOKE recognized (the ASR labels). The excerpt is also annotated with concepts that will be described next.

2.1 Speech Recognition Problems (SRP)

One form of SRP is the **Rejection**. Rejections occur when ITSPOKE is not confident enough in the recognition hypothesis and asks the student to repeat (Figure 1, $STD_{3,4}$). For our χ^2 analysis, we define the **REJ** variable with two values: **Rej** (a rejection occurred in the turn) and **noRej** (no rejection occurred in the turn). Not surprisingly, ITSPOKE also misrecognized some student turns. When ITSPOKE heard something different than what the student actually said but was confident in its hypothesis, we call this an **ASR Misrecognition** (a binary version of the commonly used Word Error Rate) (Figure 1, $STD_{1,2}$). Similarly, we define the **ASR MIS** variable with two values: **AsrMis** and **noAsrMis**.

Semantic accuracy is more relevant for dialogue evaluation, as it does not penalize for word errors that are unimportant to overall utterance interpretation. In the case of form-based information access spoken dialogue systems, computing semantic accuracy is straightforward (i.e. concept accuracy = percentage of correctly recognized concepts). In contrast, in the tutoring domain there are no clear forms with slots to be filled. We base our semantic accuracy on the "correctness" measure of the student turn. For each student turn, ITSPOKE interprets it and labels its correctness with regard to whether the student correctly answered the tutor question (see the labels between square brackets in Figure 1). We define **Semantic Misrecognition** as cases where ITSPOKE was confident in its recognition hypothesis and the correctness interpretation of the recognition hypothesis is different from the correctness interpretation of the manual transcript (Figure 1, STD_1). Similarly, we define the **SEM MIS** variable with two values: **SemMis** and **noSemMis**. The top part of Table 1 lists the distribution for our three SRP variables.

[1] We use the term "emotion" loosely to cover both affects and attitudes that can impact student learning.

3 Student State Annotation

Our definition of student state includes annotation on three dimensions (our three dialogue factors): Frustration/Anger/Hyperarticulation, Certainty and Correctness (see Table 1). The first two dimensions capture the student emotions and the last one the student correctness.

ITSPOKE: Good. Apply the same force to an object of mass M3, which is three times the mass of M1. What is the magnitude M3's acceleration in terms of A1 [acceleration of the object M1]?

STD$_1$: third of a one (***CERT=Uncertain, CRCT=C***) [*correct*]

ASR$_1$: they're it but a one (***AsrMis, SemMis***) [*incorrect*]

ITSPOKE: The acceleration of M3 [...] would be one-third the acceleration of M1. [...] In general for the same force what remains constant?

STD$_2$: weight (***CRCT=I***) [*incorrect*]

ASR$_2$: great (***AsrMis***) [*incorrect*]

ITSPOKE: Well... The product of the mass of a body and its acceleration remains constant for a given force. What is the relationship between a force acting on the box, the mass of the box, and its acceleration?

STD$_3$: newton's second law (***CERT=Uncertain, CRCT=PC***)

ASR$_3$: the cancel the fire (***Rej***)

ITSPOKE: Could you please repeat that?

STD$_4$: newton's second law (***FAH=FrAng, CRCT=PC***)

ASR$_4$: newton second long (***Rej***)

Figure 1: Human-Computer Dialogue Excerpt

The ***Frustration/Anger/Hyperarticulation*** dimension captures the perceived negative student emotional response to the interaction with the system. Three labels were used to annotate this dimension: frustration-anger, hyperarticulation and neutral. Similar to (Ang et al., 2002), because frustration and anger can be difficult to distinguish reliably, they were collapsed into a single label: frustration-anger (Figure 1, STD$_4$). Often, frustration and anger is prosodically marked and in many cases the prosody used is consistent with hyperarticulation (Ang et al., 2002). For this reason we included in this dimension the hyperarticulation label (even though hyperarticulation is not an emotion but a state). We used the hyperarticulation label for turns where no frustration or anger was perceived but nevertheless were hyperarticulated. For our interaction experiments we define the **FAH** variable with three values: **FrAng** (frustration-anger), **Hyp** (hyperarticulation) and **Neutral**.

The ***Certainty*** dimension captures the perceived student reaction to the questions asked by our computer tutor and her overall reaction to the tutoring domain (Liscombe et al., 2005). (Forbes-Riley and Litman, 2005) show that student certainty interacts with a human tutor's dialogue decision process (i.e. the choice of feedback). Four labels were used for this dimension: certain, uncertain (e.g. Figure 1, STD$_1$), mixed and neutral. In a small number of turns, both certainty and uncertainty were expressed and these turns were labeled as mixed (e.g. the student was certain about a concept, but uncertain about another concept needed to answer the tutor's question). For our interaction experiments we define the **CERT** variable with four values: **Certain**, **Uncertain**, **Mixed** and **Neutral**.

Variable	Values	Student turns (2334)
Speech recognition problems		
ASR MIS	AsrMis	25.4%
	noAsrMis	74.6%
SEM MIS	SemMis	5.7%
	noSemMis	94.3%
REJ	Rej	7.0%
	noRej	93.0%
Student state		
FAH	FrAng	9.9%
	Hyp	2.1%
	Neutral	88.0%
CERT	Certain	41.3%
	Uncertain	19.1%
	Mixed	2.4%
	Neutral	37.3%
CRCT	C	63.3%
	I	23.3%
	PC	6.2%
	UA	7.1%
EnE	Emotional	64.8%
	Neutral	35.2%

Table 1: Variable distributions in our corpus.

To test the impact of the emotion annotation level, we define the Emotional/Non-Emotional annotation based on our two emotional dimensions: neutral turns on both the FAH and the CERT dimension are labeled as neutral[2]; all other turns were labeled as emotional. Consequently, we define the **EnE** variable with two values: **Emotional** and **Neutral**.

Correctness is also an important factor of the student state. In addition to the correctness labels assigned by ITSPOKE (recall the definition of SEM MIS), each student turn was manually annotated by a project staff member in terms of their physics-related correctness. Our annotator used the human transcripts and his physics knowledge to label each student turn for various

[2] To be consistent with our previous work, we label hyperarticulated turns as emotional even though hyperarticulation is not an emotion.

degrees of correctness: correct, partially correct, incorrect and unable to answer. Our system can ask the student to provide multiple pieces of information in her answer (e.g. the question "Try to name the forces acting on the packet. Please, specify their directions." asks for both the names of the forces and their direction). If the student answer is correct and contains all pieces of information, it was labeled as correct (e.g. "gravity, down"). The partially correct label was used for turns where part of the answer was correct but the rest was either incorrect (e.g. "gravity, *up*") or omitted some information from the ideal correct answer (e.g. "gravity"). Turns that were completely incorrect (e.g. "no forces") were labeled as incorrect. Turns where the students did not answer the computer tutor's question were labeled as "unable to answer". In these turns the student used either variants of "I don't know" or simply did not say anything. For our interaction experiments we defined the **CRCT** variable with four values: **C** (correct), **I** (incorrect), **PC** (partially correct) and **UA** (unable to answer).

Please note that our definition of student state is from the tutor's perspective. As we mentioned before, our emotion annotation is for perceived emotions. Similarly, the notion of correctness is from the tutor's perspective. For example, the student might think she is correct but, in reality, her answer is incorrect. This correctness should be contrasted with the correctness used to define SEM MIS. The SEM MIS correctness uses ITSPOKE's language understanding module applied to recognition hypothesis or the manual transcript, while the student state's correctness uses our annotator's language understanding.

All our student state annotations are at the turn level and were performed manually by the same annotator. While an inter-annotator agreement study is the best way to test the ***reliability*** of our two emotional annotations (FAH and CERT), our experience with annotating student emotions (Litman and Forbes-Riley, 2004) has shown that this type of annotation can be performed reliably. Given the general importance of the student's uncertainty for tutoring, a second annotator has been commissioned to annotate our corpus for the presence or absence of uncertainty. This annotation can be directly compared with a binary version of CERT: Uncertain+Mixed versus Certain+Neutral. The comparison yields an agreement of 90% with a Kappa of 0.68. Moreover, if we rerun our study on the second annotation, we find similar dependencies. We are currently planning to perform a second annotation of the FAH dimension to validate its reliability.

We believe that our correctness annotation (CRCT) is reliable due to the simplicity of the task: the annotator uses his language understanding to match the human transcript to a list of correct/incorrect answers. When we compared this annotation with the correctness assigned by ITSPOKE on the human transcript, we found an agreement of 90% with a Kappa of 0.79.

4 Identifying dependencies using χ^2

To discover the dependencies between our variables, we apply the χ^2 test. We illustrate our analysis method on the interaction between certainty (CERT) and rejection (REJ). The χ^2 value assesses whether the differences between observed and expected counts are large enough to conclude a statistically significant dependency between the two variables (Table 2, last column). For Table 2, which has 3 degrees of freedom ((4-1)*(2-1)), the critical χ^2 value at a p<0.05 is 7.81. We thus conclude that there is a statistically significant dependency between the student certainty in a turn and the rejection of that turn.

Combination		Obs.	Exp.	χ^2
CERT – REJ				11.45
Certain – Rej	-	49	67	9.13
Uncertain – Rej	+	43	31	6.15

Table 2: CERT – REJ interaction.

If any of the two variables involved in a significant dependency has more than 2 possible values, we can look more deeply into this overall interaction by investigating how particular values interact with each other. To do that, we compute a binary variable for each variable's value in part and study dependencies between these variables. For example, for the value 'Certain' of variable CERT we create a binary variable with two values: 'Certain' and 'Anything Else' (in this case Uncertain, Mixed and Neutral). By studying the dependency between binary variables we can understand how the interaction works.

Table 2 reports in rows 3 and 4 all significant interactions between the values of variables CERT and REJ. Each row shows: 1) the value for each original variable, 2) the sign of the dependency, 3) the observed counts, 4) the expected counts and 5) the χ^2 value. For example, in our data there are 49 rejected turns in which the student was certain. This value is smaller than the expected counts (67); the dependency between Certain and Rej is significant with a χ^2 value of 9.13. A comparison of the observed counts and expected counts reveals the direction (sign) of the dependency. In our case we see that certain turns are rejected *less* than expected (row 3), while uncertain turns are rejected *more* than expected (row 4). On the other hand, there is no interaction between neutral turns and rejections or between mixed turns and rejections. Thus, the CERT – REJ interaction is explained only by the interaction between Certain and Rej and the interaction between Uncertain and Rej.

5 Results - dependencies

In this section we present all significant dependencies between SRP and student state both within and across turns. Within turn interactions analyze the contribution of the student state to the recognition of the turn. They were motivated by the widely believed intuition that emotion interacts with SRP. Across turn interactions look at the contribution of previous SRP to the current student state. Our previous work (Rotaru and Litman, 2005) had shown that certain SRP will correlate with emotional responses from the user. We also study the impact of the emotion annotation level (EnE versus FAH/CERT) on the interactions we observe. The implications of these dependencies will be discussed in Section 6.

5.1 Within turn interactions

For the FAH dimension, we find only one significant interaction: the interaction between the FAH student state and the rejection of the current turn (Table 3). By studying values' interactions, we find that turns where the student is frustrated or angry are rejected more than expected (34 instead of 16; Figure 1, STD_4 is one of them). Similarly, turns where the student response is hyperarticulated are also rejected more than expected (similar to observations in (Soltau and Waibel, 2000)). In contrast, neutral turns in the FAH dimension are rejected less than expected. Surprisingly, FrAng does not interact with AsrMis as observed in (Bulyko et al., 2005) but they use the full word error rate measure instead of the binary version used in this paper.

Combination		Obs.	Exp.	χ^2
FAH – REJ				77.92
FrAng – Rej	+	34	16	23.61
Hyp – Rej	+	16	3	50.76
Neutral – Rej	-	113	143	57.90

Table 3: FAH – REJ interaction.

Next we investigate how our second emotion annotation, CERT, interacts with SRP. All significant dependencies are reported in Tables 2 and 4. In contrast with the FAH dimension, here we see that the interaction direction depends on the valence. We find that 'Certain' turns have less SRP than expected (in terms of AsrMis and Rej). In contrast, 'Uncertain' turns have more SRP both in terms of AsrMis and Rej. 'Mixed' turns interact only with AsrMis, allowing us to conclude that the presence of uncertainty in the student turn (partial or overall) will result in ASR problems more than expected. Interestingly, on this dimension, neutral turns do not interact with any of our three SRP.

Combination		Obs.	Exp.	χ^2
CERT – ASRMIS				38.41
Certain – AsrMis	-	204	244	15.32
Uncertain – AsrMis	+	138	112	9.46
Mixed – AsrMis	+	29	13	22.27

Table 4: CERT – ASRMIS interaction.

Finally, we look at interactions between student correctness and SRP. Here we find significant dependencies with all types of SRP (see Table 5). In general, correct student turns have fewer SRP while incorrect, partially correct or UA turns have more SRP than expected. Partially correct turns have more AsrMis and SemMis problems than expected, but are rejected less than expected. Interestingly, UA turns interact only with rejections: these turns are rejected more than expected. An analysis of our corpus reveals that in most rejected UA turns the student does not say anything; in these cases, the system's recognition module thought the student said something but the system correctly rejects the recognition hypothesis.

Combination		Obs.	Exp.	χ^2
CRCT – ASRMIS				65.17
C – AsrMis	-	295	374	62.03
I – AsrMis	+	198	137	45.95
PC – AsrMis	+	50	37	5.9
CRCT – SEMMIS				20.44
C – SemMis	+	100	84	7.83
I – SemMis	-	14	31	13.09
PC – SemMis	+	15	8	5.62
CRCT – REJ				99.48
C – Rej	-	53	102	70.14
I – Rej	+	84	37	79.61
PC – Rej	-	4	10	4.39
UA – Rej	+	21	11	9.19

Table 5: Interactions between Correctness and SRP.

The only exception to the rule is SEM MIS. We believe that SEM MIS behavior is explained by the "catch-all" implementation in our system. In ITSPOKE, for each tutor question there is a list of anticipated answers. All other answers are treated as incorrect. Thus, it is less likely that a recognition problem in an incorrect turn will affect the correctness interpretation (e.g. Figure 1, STD_2: very unlikely to misrecognize the incorrect "weight" with the anticipated "the product of mass and acceleration"). In contrast, in correct turns recognition problems are more likely to screw up the correctness interpretation (e.g. misrecognizing "gravity down" as "gravity sound").

5.2 Across turn interactions

Next we look at the contribution of previous SRP – variable name or value followed by $_{(-1)}$ – to the current student state. Please note that there are two factors involved here: the presence of the SRP and the SRP handling strategy. In ITSPOKE, whenever a student turn is rejected, unless this is the third rejection in a row, the student is asked to repeat using variations of "Could you please repeat that?". In all other cases, ITSPOKE makes use of the available information ignoring any potential ASR errors.

Combination		Obs.	Exp.	χ^2
$ASRMIS_{(-1)}$ – FAH				7.64
$AsrMis_{(-1)}$ – FrAng	$-^t$	46	58	3.73
$AsrMis_{(-1)}$ – Hyp	$-^t$	7	12	3.52
$AsrMis_{(-1)}$ – Neutral	+	527	509	6.82
$REJ_{(-1)}$ – FAH				409.31
$Rej_{(-1)}$ – FrAng	+	36	16	28.95
$Rej_{(-1)}$ – Hyp	+	38	3	369.03
$Rej_{(-1)}$ – Neutral	-	88	142	182.9
$REJ_{(-1)}$ – CRCT				57.68
$Rej_{(-1)}$ – C	-	68	101	31.94
$Rej_{(-1)}$ – I	+	74	37	49.71
$Rej_{(-1)}$ – PC	-	3	10	6.25

Table 6: Interactions across turns (^{t} – trend, p<0.1).

Here we find only 3 interactions (Table 6). We find that after a non-harmful SRP (AsrMis) the student is less frustrated and hyperarticulated than expected. This result is not surprising since an AsrMis does not have any effect on the normal dialogue flow.

In contrast, after rejections we observe several negative events. We find a highly significant interaction between a previous rejection and the student FAH state, with student being more frustrated and more hyperarticulated than expected (e.g. Figure 1, STD_4). Not only does the system elicit an emotional reaction from the student after a rejection, but her subsequent response to the repetition request suffers in terms of the correctness. We find that after rejections student answers are correct or partially correct less than expected and incorrect more than expected. The $REJ_{(-1)}$ – CRCT interaction might be explained by the CRCT – REJ interaction (Table 5) if, in general, after a rejection the student repeats her previous turn. An annotation of responses to rejections as in (Swerts et al., 2000) (repeat, rephrase etc.) should provide additional insights.

We were surprised to see that a previous SemMis (more harmful than an AsrMis but less disruptive than a Rej) does not interact with the student state; also the student certainty does not interact with previous SRP.

5.3 Emotion annotation level

We also study the impact of the emotion annotation level on the interactions we can observe from our corpus. In this section, we look at interactions between SRP and our coarse-level emotion annotation (EnE) both within and across turns. Our results are similar with the results of our previous work (Rotaru and Litman, 2005) on a smaller corpus and a similar annotation scheme. We find again only one significant interaction: rejections are followed by more emotional turns than expected (Table 7). The strength of the interaction is smaller than in previous work, though the results can not be compared directly. No other dependencies are present.

Combination		Obs.	Exp.	χ^2
$REJ_{(-1)}$ – EnE				6.19
$Rej_{(-1)}$ – Emotional	+	119	104	6.19

Table 7: $REJ_{(-1)}$ – EnE interaction.

We believe that the $REJ_{(-1)}$ – EnE interaction is explained mainly by the FAH dimension. Not only is there no interaction between $REJ_{(-1)}$ and CERT, but the inclusion of the CERT dimension in the EnE annotation decreases the strength of the interaction between REJ and FAH (the χ^2 value decreases from 409.31 for FAH to a mere 6.19 for EnE). Collapsing emotional classes also prevents us from seeing any within turn interactions. These observations suggest that what is being counted as an emotion for a binary emotion annotation is critical its success. In our case, if we look at affect (FAH) or attitude (CERT) in isolation we find many interactions; in contrast, combining them offers little insight.

6 Results – insights & strategies

Our results put a spotlight on several interesting observations which we discuss below.

Emotions interact with SRP

The dependencies between FAH/CERT and various SRP (Tables 2-4) provide evidence that user's emotions interact with the system's ability to recognize the current turn. This is a widely believed intuition with little empirical support so far. Thus, our notion of student state can be a useful higher level information source for SRP predictors. Similar to (Hirschberg et al., 2004), we believe that peculiarities in the acoustic/prosodic profile of specific student states are responsible for their SRP. Indeed, previous work has shown that the acoustic/prosodic information plays an important role in characterizing and predicting both FAH (Ang et al., 2002; Soltau and Waibel, 2000) and CERT (Liscombe et al., 2005; Swerts and Krahmer, 2005).

The impact of the emotion annotation level
A comparison of the interactions yielded by various levels of emotion annotation shows the importance of the annotation level. When using a coarser level annotation (EnE) we find only one interaction. By using a finer level annotation, not only we can understand this interaction better but we also discover new interactions (five interactions with FAH and CERT). Moreover, various state annotations interact differently with SRP. For example, non-neutral turns in the FAH dimension (FrAng and Hyp) will be always rejected more than expected (Table 3); in contrast, interactions between non-neutral turns in the CERT dimension and rejections depend on the valence ('certain' turns will be rejected less than expected while 'uncertain' will be rejected more than expected; recall Table 2). We also see that the neutral turns interact with SRP depending on the dimension that defines them: FAH neutral turns interact with SRP (Table 3) while CERT neutral turns do not (Tables 2 and 4).

This insight suggests an interesting tradeoff between the practicality of collapsing emotional classes (Ang et al., 2002; Litman and Forbes-Riley, 2004) and the ability to observe meaningful interactions via finer level annotations.

Rejections: impact and a handling strategy
Our results indicate that rejections and ITSPOKE's current rejection-handling strategy are problematic. We find that rejections are followed by more emotional turns (Table 7). A similar effect was observed in our previous work (Rotaru and Litman, 2005). The fact that it generalizes across annotation scheme and corpus, emphasizes its importance. When a finer level annotation is used, we find that rejections are followed more than expected by a frustrated, angry and hyperarticulated user (Table 6). Moreover, these subsequent turns can result in additional rejections (Table 3). Asking to repeat after a rejection does not also help in terms of correctness: the subsequent student answer is actually incorrect more than expected (Table 6).

These interactions suggest an interesting strategy for our tutoring task: favoring misrecognitions over rejections (by lowering the rejection threshold). First, since rejected turns are more than expected incorrect (Table 5), the actual recognized hypothesis for such turns turn is very likely to be interpreted as incorrect. Thus, accepting a rejected turn instead of rejecting it will have the same outcome in terms of correctness: an incorrect answer. In this way, instead of attempting to acquire the actual student answer by asking to repeat, the system can skip these extra turn(s) and use the current hypothesis. Second, the other two SRP are less taxing in terms of eliciting FAH emotions (recall Table 6; note that a SemMis might activate an unwarranted and lengthy knowledge remediation subdialogue). This suggests that continuing the conversation will be more beneficial even if the system misunderstood the student. A similar behavior was observed in human-human conversations through a noisy speech channel (Skantze, 2005).

Correctness/certainty–SRP interactions
We also find an interesting interaction between correctness/certainty and system's ability to recognize that turn. In general correct/certain turns have less SRP while incorrect/uncertain turns have more SRP than expected. This observation suggests that the computer tutor should ask the right question (in terms of its difficulty) at the right time. Intuitively, asking a more complicated question when the student is not prepared to answer it will increase the likelihood of an incorrect or uncertain answer. But our observations show that the computer tutor has more trouble recognizing correctly these types of answers. This suggests an interesting tradeoff between the tutor's question difficulty and the system's ability to recognize the student answer. This tradeoff is similar in spirit to the initiative-SRP tradeoff that is well known when designing information-seeking systems (e.g. system initiative is often used instead of a more natural mixed initiative strategy, in order to minimize SRP).

7 Conclusions

In this paper we analyze the interactions between SRP and three higher level dialogue factors that define our notion of student state: frustration/anger/hyperarticulation, certainty and correctness. Our analysis produces several interesting insights and strategies which confirm the utility of the proposed approach. We show that user emotions interact with SRP and that the emotion annotation level affects the interactions we observe from the data, with finer-level emotions yielding more interactions and insights.

We also find that tutoring, as a new domain for speech applications, brings forward new important factors for spoken dialogue design: certainty and correctness. Both factors interact with SRP and these interactions highlight an interesting design practice in the spoken tutoring applications: the tradeoff between the pedagogical value of asking difficult questions and the system's ability to recognize the student answer (at least in our system). The particularities of the tutoring domain also suggest favoring misrecognitions over rejections to reduce the negative impact of asking to repeat after rejections.

In our future work, we plan to move to the third step of our approach: testing the strategies suggested by our results. For example, we will implement a new version of ITSPOKE that never rejects the student turn. Next, the current version and the new version will be compared with respect to users' emotional response. Similarly, to test the tradeoff hypothesis, we will implement a version of ITSPOKE that asks difficult questions first and then falls back to simpler questions. A comparison of the two versions in terms of the number of SRP can be used for validation.

While our results might be dependent on the tutoring system used in this experiment, we believe that our findings can be of interest to practitioners building similar voice-based applications. Moreover, our approach can be applied easily to studying other systems.

Acknowledgements

This work is supported by NSF Grant No. 0328431. We thank Dan Bohus, Kate Forbes-Riley, Joel Tetreault and our anonymous reviewers for their helpful comments.

References

J. Ang, R. Dhillon, A. Krupski, A. Shriberg and A. Stolcke. 2002. *Prosody-based automatic detection of annoyance and frustration in human-computer dialog.* In Proc. of ICSLP.

I. Bulyko, K. Kirchhoff, M. Ostendorf and J. Goldberg. 2005. Error-correction detection and response generation in a spoken dialogue system. *Speech Communication, 45*(3).

L. Chase. 1997. *Blame Assignment for Errors Made by Large Vocabulary Speech Recognizers.* In Proc. of Eurospeech.

K. Forbes-Riley and D. J. Litman. 2005. *Using Bigrams to Identify Relationships Between Student Certainness States and Tutor Responses in a Spoken Dialogue Corpus.* In Proc. of SIGdial.

M. Frampton and O. Lemon. 2005. *Reinforcement Learning of Dialogue Strategies using the User's Last Dialogue Act.* In Proc. of IJCAI Workshop on Know.&Reasoning in Practical Dialogue Systems.

M. Gabsdil and O. Lemon. 2004. *Combining Acoustic and Pragmatic Features to Predict Recognition Performance in Spoken Dialogue Systems.* In Proc. of ACL.

J. Hirschberg, D. Litman and M. Swerts. 2004. Prosodic and Other Cues to Speech Recognition Failures. *Speech Communication, 43*(1-2).

M. Kearns, C. Isbell, S. Singh, D. Litman and J. Howe. 2002. *CobotDS: A Spoken Dialogue System for Chat.* In Proc. of National Conference on Artificial Intelligence (AAAI).

J. Liscombe, J. Hirschberg and J. J. Venditti. 2005. *Detecting Certainness in Spoken Tutorial Dialogues.* In Proc. of Interspeech.

D. Litman and K. Forbes-Riley. 2004. *Annotating Student Emotional States in Spoken Tutoring Dialogues.* In Proc. of SIGdial Workshop on Discourse and Dialogue (SIGdial).

H. Pon-Barry, B. Clark, E. O. Bratt, K. Schultz and S. Peters. 2004. *Evaluating the effectiveness of Scot:a spoken conversational tutor.* In Proc. of ITS Workshop on Dialogue-based Intellig. Tutoring Systems.

M. Rotaru and D. Litman. 2005. *Interactions between Speech Recognition Problems and User Emotions.* In Proc. of Eurospeech.

G. Skantze. 2005. Exploring human error recovery strategies: Implications for spoken dialogue systems. *Speech Communication, 45*(3).

H. Soltau and A. Waibel. 2000. *Specialized acoustic models for hyperarticulated speech.* In Proc. of ICASSP.

M. Swerts and E. Krahmer. 2005. Audiovisual Prosody and Feeling of Knowing. *Journal of Memory and Language, 53.*

M. Swerts, D. Litman and J. Hirschberg. 2000. *Corrections in Spoken Dialogue Systems.* In Proc. of ICSLP.

K. VanLehn, P. W. Jordan, C. P. Rosé, et al. 2002. *The Architecture of Why2-Atlas: A Coach for Qualitative Physics Essay Writing.* In Proc. of Intelligent Tutoring Systems (ITS).

M. Walker, D. Litman, C. Kamm and A. Abella. 2000. Towards Developing General Models of Usability with PARADISE. *Natural Language Engineering.*

M. Walker, R. Passonneau and J. Boland. 2001. *Quantitative and Qualitative Evaluation of Darpa Communicator Spoken Dialogue Systems.* In Proc. of ACL.

Learning the Structure of Task-driven Human-Human Dialogs

Srinivas Bangalore
AT&T Labs-Research
180 Park Ave
Florham Park, NJ 07932
srini@research.att.com

Giuseppe Di Fabbrizio
AT&T Labs-Research
180 Park Ave
Florham Park, NJ 07932
pino@research.att.com

Amanda Stent
Dept of Computer Science
Stony Brook University
Stony Brook, NY
stent@cs.sunysb.edu

Abstract

Data-driven techniques have been used for many computational linguistics tasks. Models derived from data are generally more robust than hand-crafted systems since they better reflect the distribution of the phenomena being modeled. With the availability of large corpora of spoken dialog, dialog management is now reaping the benefits of data-driven techniques. In this paper, we compare two approaches to modeling subtask structure in dialog: a chunk-based model of subdialog sequences, and a parse-based, or hierarchical, model. We evaluate these models using customer agent dialogs from a catalog service domain.

1 Introduction

As large amounts of language data have become available, approaches to sentence-level processing tasks such as parsing, language modeling, named-entity detection and machine translation have become increasingly data-driven and empirical. Models for these tasks can be trained to capture the distributions of phenomena in the data resulting in improved robustness and adaptability. However, this trend has yet to significantly impact approaches to dialog management in dialog systems. Dialog managers (both plan-based and call-flow based, for example (Di Fabbrizio and Lewis, 2004; Larsson et al., 1999)) have traditionally been hand-crafted and consequently somewhat brittle and rigid. With the ability to record, store and process large numbers of human-human dialogs (e.g. from call centers), we anticipate that data-driven methods will increasingly influence approaches to dialog management.

A successful dialog system relies on the synergistic working of several components: speech recognition (ASR), spoken language understanding (SLU), dialog management (DM), language generation (LG) and text-to-speech synthesis (TTS). While data-driven approaches to ASR and SLU are prevalent, such approaches to DM, LG and TTS are much less well-developed. In ongoing work, we are investigating data-driven approaches for building all components of spoken dialog systems.

In this paper, we address one aspect of this problem – *inferring predictive models to structure task-oriented dialogs*. We view this problem as a first step in predicting the system state of a dialog manager and in predicting the system utterance during an incremental execution of a dialog. In particular, we learn models for predicting dialog acts of utterances, and models for predicting subtask structures of dialogs. We use three different dialog act tag sets for three different human-human dialog corpora. We compare a flat chunk-based model to a hierarchical parse-based model as models for predicting the task structure of dialogs.

The outline of this paper is as follows: In Section 2, we review current approaches to building dialog systems. In Section 3, we review related work in data-driven dialog modeling. In Section 4, we present our view of analyzing the structure of task-oriented human-human dialogs. In Section 5, we discuss the problem of segmenting and labeling dialog structure and building models for predicting these labels. In Section 6, we report experimental results on Maptask, Switchboard and a dialog data collection from a catalog ordering service domain.

2 Current Methodology for Building Dialog systems

Current approaches to building dialog systems involve several manual steps and careful crafting of different modules for a particular domain or application. The process starts with a small scale “Wizard-of-Oz” data collection where subjects talk to a machine driven by a human ‘behind the curtains’. A user experience (UE) engineer analyzes the collected dialogs, subject matter expert interviews, user testimonials and other evidences (e.g. customer care history records). This heterogeneous set of information helps the UE engineer to design some system functionalities, mainly: the

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 201–208,
Sydney, July 2006. ©2006 Association for Computational Linguistics

semantic scope (e.g. call-types in the case of call routing systems), the LG model, and the DM strategy. A larger automated data collection follows, and the collected data is transcribed and labeled by expert labelers following the UE engineer recommendations. Finally, the transcribed and labeled data is used to train both the ASR and the SLU.

This approach has proven itself in many commercial dialog systems. However, the initial UE requirements phase is an expensive and error-prone process because it involves non-trivial design decisions that can only be evaluated after system deployment. Moreover, scalability is compromised by the time, cost and high level of UE know-how needed to reach a consistent design.

The process of building speech-enabled automated contact center services has been formalized and cast into a scalable commercial environment in which dialog components developed for different applications are reused and adapted (Gilbert et al., 2005). However, we still believe that exploiting dialog data to train/adapt or complement hand-crafted components will be vital for robust and adaptable spoken dialog systems.

3 Related Work

In this paper, we discuss methods for automatically creating models of dialog structure using dialog act and task/subtask information. Relevant related work includes research on automatic dialog act tagging and stochastic dialog management, and on building hierarchical models of plans using task/subtask information.

There has been considerable research on statistical dialog act tagging (Core, 1998; Jurafsky et al., 1998; Poesio and Mikheev, 1998; Samuel et al., 1998; Stolcke et al., 2000; Hastie et al., 2002). Several disambiguation methods (n-gram models, hidden Markov models, maximum entropy models) that include a variety of features (cue phrases, speaker ID, word n-grams, prosodic features, syntactic features, dialog history) have been used. In this paper, we show that use of extended context gives improved results for this task.

Approaches to dialog management include AI-style plan recognition-based approaches (e.g. (Sidner, 1985; Litman and Allen, 1987; Rich and Sidner, 1997; Carberry, 2001; Bohus and Rudnicky, 2003)) and information state-based approaches (e.g. (Larsson et al., 1999; Bos et al., 2003; Lemon and Gruenstein, 2004)). In recent years, there has been considerable research on how to automatically learn models of both types from data. Researchers who treat dialog as a sequence of information states have used reinforcement learning and/or Markov decision processes to build stochastic models for dialog management that are evaluated by means of dialog simulations (Levin and Pieraccini, 1997; Scheffler and Young, 2002; Singh et al., 2002; Williams et al., 2005; Henderson et al., 2005; Frampton and Lemon, 2005). Most recently, Henderson et al. showed that it is possible to automatically learn good dialog management strategies from automatically labeled data over a large potential space of dialog states (Henderson et al., 2005); and Frampton and Lemon showed that the use of context information (the user's last dialog act) can improve the performance of learned strategies (Frampton and Lemon, 2005). In this paper, we combine the use of automatically labeled data and extended context for automatic dialog modeling.

Other researchers have looked at probabilistic models for plan recognition such as extensions of Hidden Markov Models (Bui, 2003) and probabilistic context-free grammars (Alexandersson and Reithinger, 1997; Pynadath and Wellman, 2000). In this paper, we compare hierarchical grammar-style and flat chunking-style models of dialog.

In recent research, Hardy (2004) used a large corpus of transcribed and annotated telephone conversations to develop the Amities dialog system. For their dialog manager, they trained separate task and dialog act classifiers on this corpus. For task identification they report an accuracy of 85% (true task is one of the top 2 results returned by the classifier); for dialog act tagging they report 86% accuracy.

4 Structural Analysis of a Dialog

We consider a task-oriented dialog to be the result of incremental creation of a shared plan by the participants (Lochbaum, 1998). The shared plan is represented as a single tree that encapsulates the task structure (dominance and precedence relations among tasks), dialog act structure (sequences of dialog acts), and linguistic structure of utterances (inter-clausal relations and predicate-argument relations within a clause), as illustrated in Figure 1. As the dialog proceeds, an utterance from a participant is accommodated into the tree in an incremental manner, much like an incremental syntactic parser accommodates the next word into a partial parse tree (Alexandersson and Reithinger, 1997). With this model, we can tightly couple language understanding and dialog management using a shared representation, which leads to improved accuracy (Taylor et al., 1998).

In order to infer models for predicting the structure of task-oriented dialogs, we label human-human dialogs with the hierarchical information shown in Figure 1 in several stages: utterance segmentation (Section 4.1), syntactic annotation (Section 4.2), dialog act tagging (Section 4.3) and subtask labeling (Section 5).

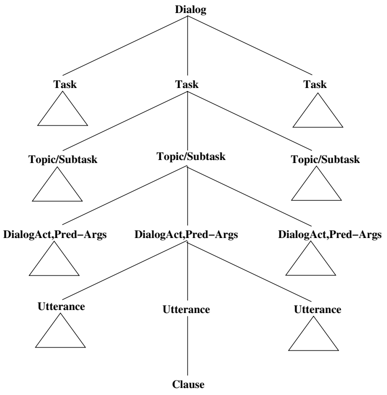


Figure 1: Structural analysis of a dialog

4.1 Utterance Segmentation

The task of ”cleaning up” spoken language utterances by detecting and removing speech repairs and dysfluencies and identifying sentence boundaries has been a focus of spoken language parsing research for several years (e.g. (Bear et al., 1992; Seneff, 1992; Shriberg et al., 2000; Charniak and Johnson, 2001)). We use a system that segments the ASR output of a user's utterance into clauses. The system annotates an utterance for sentence boundaries, restarts and repairs, and identifies coordinating conjunctions, filled pauses and discourse markers. These annotations are done using a cascade of classifiers, details of which are described in (Bangalore and Gupta, 2004).

4.2 Syntactic Annotation

We automatically annotate a user's utterance with supertags (Bangalore and Joshi, 1999). Supertags encapsulate predicate-argument information in a local structure. They are composed with each other using the substitution and adjunction operations of Tree-Adjoining Grammars (Joshi, 1987) to derive a dependency analysis of an utterance and its predicate-argument structure.

4.3 Dialog Act Tagging

We use a domain-specific dialog act tagging scheme based on an adapted version of DAMSL (Core, 1998). The DAMSL scheme is quite comprehensive, but as others have also found (Jurafsky et al., 1998), the multi-dimensionality of the scheme makes the building of models from DAMSL-tagged data complex. Furthermore, the generality of the DAMSL tags reduces their utility for natural language generation. Other tagging schemes, such as the Maptask scheme (Carletta et al., 1997), are also too general for our purposes. We were particularly concerned with obtaining sufficient discriminatory power between different types of statement (for generation), and to include an out-of-domain tag (for interpretation). We provide a sample list of our dialog act tags in Table 2. Our experiments in automatic dialog act tagging are described in Section 6.3.

5 Modeling Subtask Structure

Figure 2 shows the task structure for a sample dialog in our domain (catalog ordering). An *order placement* task is typically composed of the sequence of subtasks *opening, contact-information, order-item, related-offers, summary*. Subtasks can be nested; the nesting structure can be as deep as five levels. Most often the nesting is at the left or right frontier of the subtask tree.

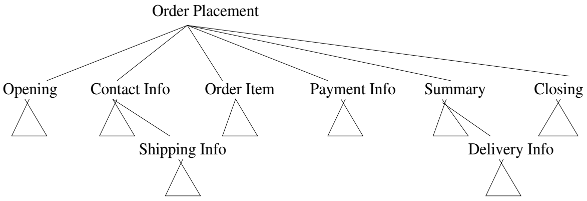


Figure 2: A sample task structure in our application domain.

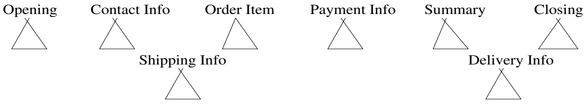


Figure 3: An example output of the chunk model's task structure

The goal of subtask segmentation is to predict if the current utterance in the dialog is part of the current subtask or starts a new subtask. We compare two models for recovering the subtask structure – a chunk-based model and a parse-based model. In the chunk-based model, we recover the precedence relations (sequence) of the subtasks but not dominance relations (subtask structure) among the subtasks. Figure 3 shows a sample output from the chunk model. In the parse model, we recover the complete task structure from the sequence of utterances as shown in Figure 2. Here, we describe our two models. We present our experiments on subtask segmentation and labeling in Section 6.4.

5.1 Chunk-based model

This model is similar to the second one described in (Poesio and Mikheev, 1998), except that we use tasks and subtasks rather than dialog games. We model the prediction problem as a classification task as follows: given a sequence of utterances u_i in a dialog $U = u_1, u_2, \ldots, u_n$ and a subtask label vocabulary ($st_i \in \mathcal{ST}$), we need to predict the best subtask label sequence $ST^* = st_1, st_2, \ldots, st_m$ as shown in equation 1.

$$ST^* = \underset{ST}{argmax}\, P(ST|U) \tag{1}$$

Each subtask has *begin*, *middle* (possibly absent) and *end* utterances. If we incorporate this information, the refined vocabulary of subtask labels is $\mathcal{ST}_r = \{st_i^b, st_i^m, st_i^e \mid st_i \in \mathcal{ST}\}$. In our experiments, we use a classifier to assign to each utterance a refined subtask label conditioned on a vector of local contextual features ($\mathbf{\Phi}$). In the interest of using an incremental left-to-right decoder, we restrict the contextual features to be from the preceding context only. Furthermore, the search is limited to the label sequences that respect precedence among the refined labels (*begin* $<$ *middle* $<$ *end*). This constraint is expressed in a grammar G encoded as a regular expression ($L(G) = \bigcup_i (st_i^b\ (st_i^m)^*\ st_i^e)^*$). However, in order to cope with the prediction errors of the classifier, we approximate $L(G)$ with an n-gram language model on sequences of the refined tag labels:

$$ST_r^* = \underset{\substack{ST_r \\ ST_r \in L(G)}}{argmax} P(ST_r|U) \tag{2}$$

$$\approx \underset{\substack{ST_r \\ ST_r \in L(G)}}{argmax} \prod_i^n P(st_i|\mathbf{\Phi}) \tag{3}$$

In order to estimate the conditional distribution $P(st_i|\mathbf{\Phi})$ we use the general technique of choosing the maximum entropy (maxent) distribution that properly estimates the average of each feature over the training data (Berger et al., 1996). This can be written as a Gibbs distribution parameterized with weights λ, where V is the size of the label set. Thus,

$$P(st_i|\mathbf{\Phi}) = \frac{e^{\lambda_{\mathbf{st_i}} \cdot \mathbf{\Phi}}}{\sum_{st=1}^{V} e^{\lambda_{\mathbf{st}} \cdot \mathbf{\Phi}}} \tag{4}$$

We use the machine learning toolkit LLAMA (Haffner, 2006) to estimate the conditional distribution using maxent. LLAMA encodes multiclass maxent as binary maxent, in order to increase the speed of training and to scale this method to large data sets. Each of the V classes in the set $\mathcal{ST}_r$ is encoded as a bit vector such that, in the vector for class i, the $i^t h$ bit is one and all other bits are zero. Then, V one-vs-other binary classifiers are used as follows.

$$P(y|\Phi) = 1 - P(\bar{y}|\Phi) = \frac{e^{\lambda_y \cdot \Phi}}{e^{\lambda_y \cdot \Phi} + e^{\lambda_{\bar{y}} \cdot \Phi}} = \frac{1}{1 + e^{-\lambda'_y \cdot \Phi}} \tag{5}$$

where $\lambda_{\bar{y}}$ is the parameter vector for the *anti-label* $\bar{y}$ and $\lambda'_{y} = \lambda_y - \lambda_{\bar{y}}$. In order to compute $P(st_i|\mathbf{\Phi})$, we use class independence assumption and require that $y_i = 1$ and for all $j \neq i$ $y_j = 0$.

$$P(st_i|\Phi) = P(y_i|\Phi) \prod_{j \neq i}^{V} P(y_j|\Phi)$$

5.2 Parse-based Model

As seen in Figure 3, the chunk model does not capture dominance relations among subtasks, which are important for resolving anaphoric references (Grosz and Sidner, 1986). Also, the chunk model is representationally inadequate for center-embedded nestings of subtasks, which do occur in our domain, although less frequently than the more prevalent "tail-recursive" structures.

In this model, we are interested in finding the most likely plan tree (PT) given the sequence of utterances:

$$PT^* = \underset{PT}{argmax}\, P(PT|U) \tag{6}$$

For real-time dialog management we use a top-down incremental parser that incorporates bottom-up information (Roark, 2001).

We rewrite equation (6) to exploit the subtask sequence provided by the chunk model as shown in Equation 7. For the purpose of this paper, we approximate Equation 7 using one-best (or *k-best*) chunk output.[1]

$$PT^* = \underset{PT}{argmax} \sum_{ST} P(ST|U)P(PT|ST) \tag{7}$$

$$\approx \underset{PT}{argmax}\, P(PT|ST^*) \tag{8}$$

where

$$ST^* = \underset{ST}{argmax}\, P(ST|U) \tag{9}$$

6 Experiments and Results

In this section, we present the results of our experiments for modeling subtask structure.

6.1 Data

As our primary data set, we used 915 telephone-based customer-agent dialogs related to the task of ordering products from a catalog. Each dialog was transcribed by hand; all numbers (telephone, credit card, etc.) were removed for privacy reasons. The average dialog lasted for 3.71

[1] However, it is conceivable to parse the multiple hypotheses of chunks (encoded as a weighted lattice) produced by the chunk model.

minutes and included 61.45 changes of speaker. A single customer-service representative might participate in several dialogs, but customers are represented by only one dialog each. Although the majority of the dialogs were on-topic, some were idiosyncratic, including: requests for order corrections, transfers to customer service, incorrectly dialed numbers, and long friendly out-of-domain asides. Annotations applied to these dialogs include: utterance segmentation (Section 4.1), syntactic annotation (Section 4.2), dialog act tagging (Section 4.3) and subtask segmentation (Section 5). The former two annotations are domain-independent while the latter are domain-specific.

6.2 Features

Offline natural language processing systems, such as part-of-speech taggers and chunkers, rely on both *static* and *dynamic* features. Static features are derived from the local context of the text being tagged. Dynamic features are computed based on previous predictions. The use of dynamic features usually requires a search for the globally optimal sequence, which is not possible when doing incremental processing. For dialog act tagging and subtask segmentation during dialog management, we need to predict incrementally since it would be unrealistic to wait for the entire dialog before decoding. Thus, in order to train the dialog act (DA) and subtask segmentation classifiers, we use only *static* features from the current and left context as shown in Table 1.[2] This obviates the need for constructing a search network and performing a dynamic programming search during decoding. In lieu of the dynamic context, we use larger static context to compute features – word trigrams and trigrams of words annotated with supertags computed from up to three previous utterances.

Label Type	Features
Dialog Acts	Speaker, word trigrams from current/previous utterance(s) supertagged utterance
Subtask	Speaker, word trigrams from current utterance, previous utterance(s)/turn

Table 1: Features used for the classifiers.

6.3 Dialog Act Labeling

For dialog act labeling, we built models from our corpus and from the Maptask (Carletta et al., 1997) and Switchboard-DAMSL (Jurafsky et al., 1998) corpora. From the files for the Maptask corpus, we extracted the moves, words and speaker information (follower/giver). Instead of using the raw move information, we augmented each move with speaker information, so that for example, the *instruct* move was split into *instruct-giver* and *instruct-follower*. For the Switchboard corpus, we clustered the original labels, removing most of the multidimensional tags and combining together tags with minimum training data as described in (Jurafsky et al., 1998). For all three corpora, non-sentence elements (e.g., dysfluencies, *discourse markers*, etc.) and restarts (with and without repairs) were kept; non-verbal content (e.g., laughs, background noise, etc.) was removed.

As mentioned in Section 4, we use a domain-specific tag set containing 67 dialog act tags for the catalog corpus. In Table 2, we give examples of our tags. We manually annotated 1864 clauses from 20 dialogs selected at random from our corpus and used a ten-fold cross-validation scheme for testing. In our annotation, a single utterance may have multiple dialog act labels. For our experiments with the Switchboard-DAMSL corpus, we used 42 dialog act tags obtained by clustering over the 375 unique tags in the data. This corpus has 1155 dialogs and 218,898 utterances; 173 dialogs, selected at random, were used for testing. The Maptask tagging scheme has 12 unique dialog act tags; augmented with speaker information, we get 24 tags. This corpus has 128 dialogs and 26181 utterances; ten-fold cross validation was used for testing.

Type	**Subtype**
Ask	Info
Explain	Catalog, CC_Related, Discount, Order_Info
	Order_Problem, Payment_Rel, Product_Info
	Promotions, Related_Offer, Shipping
Convers- -ational	Ack, Goodbye, Hello, Help, Hold, YoureWelcome, Thanks, Yes, No, Ack,
	Repeat, Not(Information)
Request	Code, Order_Problem, Address, Catalog,
	CC_Related, Change_Order, Conf, Credit,
	Customer_Info, Info, Make_Order, Name,
	Order_Info, Order_Status, Payment_Rel,
	Phone_Number, Product_Info, Promotions,
	Shipping, Store_Info
YNQ	Address, Email, Info, Order_Info,
	Order_Status,Promotions, Related_Offer

Table 2: Sample set of dialog act labels

Table 3 shows the error rates for automatic dialog act labeling using word trigram features from the current and previous utterance. We compare error rates for our tag set to those of Switchboard-DAMSL and Maptask using the same features and the same classifier learner. The error rates for the catalog and the Maptask corpus are an average of ten-fold cross-validation. We suspect that the larger error rate for our domain compared to Maptask and Switchboard might be due to the small size of our annotated corpus (about 2K utterances for our domain as against about 20K utterances for

[2] We could use dynamic contexts as well and adopt a greedy decoding algorithm instead of a viterbi search. We have not explored this approach in this paper.

Maptask and 200K utterances for DAMSL).

The error rates for the Switchboard-DAMSL data are significantly better than previously published results (28% error rate) (Jurafsky et al., 1998) with the same tag set. This improvement is attributable to the richer feature set we use and a discriminative modeling framework that supports a large number of features, in contrast to the generative model used in (Jurafsky et al., 1998). A similar obeservation applies to the results on Maptask dialog act tagging. Our model outperforms previously published results (42.8% error rate) (Poesio and Mikheev, 1998).

In labeling the Switchboard data, long utterances were split into *slash units* (Meteer et.al., 1995). A speaker's turn can be divided in one or more slash units and a slash unit can extend over multiple turns, for example:

sv B.64 utt3: C but, F uh –
b A.65 utt1: Uh-huh. /
+ B.66 utt1: – people want all of that /
sv B.66 utt2: C and not all of those are necessities. /
b A.67 utt1: Right . /

The labelers were instructed to label on the basis of the whole slash unit. This makes, for example, the dysfluency turn B.64 a Statement opinion (sv) rather than a non-verbal. For the purpose of discriminative learning, this could introduce noisy data since the context associated to the labeling decision shows later in the dialog. To address this issue, we compare 2 classifiers: the first (non-merged), simply propagates the same label to each continuation, cross turn slash unit; the second (merged) combines the units in one single utterance. Although the *merged* classifier breaks the regular structure of the dialog, the results in Table 3 show better overall performance.

Tagset	current utterance	+ stagged utterance	+ 3 previous (stagged) utterance
Catalog Domain	46.3	46.1	42.2
DAMSL (non-merged)	24.7	23.8	19.1
DAMSL (merged)	22.0	20.6	16.5
Maptask	34.3	33.9	30.3

Table 3: Error rates in dialog act tagging

6.4 Subtask Segmentation and Labeling

For subtask labeling, we used a random partition of 864 dialogs from our catalog domain as the training set and 51 dialogs as the test set. All the dialogs were annotated with subtask labels by hand. We used a set of 18 labels grouped as shown in Figure 4.

Type	**Subtask Labels**
1	opening, closing
2	contact-information, delivery-information, payment-information, shipping-address,summary
3	order-item, related-offer, order-problem discount, order-change, check-availability
4	call-forward, out-of-domain, misc-other, sub-call

Table 4: Subtask label set

6.4.1 Chunk-based Model

Table 5 shows error rates on the test set when predicting refined subtask labels using word n-gram features computed on different dialog contexts. The well-formedness constraint on the refined subtask labels significantly improves prediction accuracy. Utterance context is also very helpful; just one utterance of left-hand context leads to a 10% absolute reduction in error rate, with further reductions for additional context. While the use of trigram features helps, it is not as helpful as other contextual information. We used the dialog act tagger trained from Switchboard-DAMSL corpus to automatically annotate the catalog domain utterances. We included these tags as features for the classifier, however, we did not see an improvement in the error rates, probably due to the high error rate of the dialog act tagger.

Feature Context	Utterance Context		
	Current utt/with DA	+prev utt/with DA	+three prev utt/with DA
Unigram	42.9/42.4 (53.4/52.8)	33.6/34.1 (43.0/43.0)	30.0/30.3 (37.6/37.6)
Trigram	41.7/41.7 (52.5/52.0)	31.6/31.4 (42.9/42.7)	30.0/29.1 (37.6/37.4)

Table 5: Error rate for predicting the refined subtask labels. The error rates without the well-formedness constraint is shown in parenthesis. The error rates with dialog acts as features are separated by a slash.

6.4.2 Parsing-based Model

We retrained a top-down incremental parser (Roark, 2001) on the plan trees in the training dialogs. For the test dialogs, we used the k-best (k=50) refined subtask labels for each utterance as predicted by the chunk-based classifier to create a lattice of subtask label sequences. For each dialog we then created n-best sequences (100-best for these experiments) of subtask labels; these were parsed and (re-)ranked by the parser.[3] We combine the weights of the subtask label sequences assigned by the classifier with the parse score assigned by the parser and select the top

[3] Ideally, we would have parsed the subtask label lattice directly, however, the parser has to be reimplemented to parse such lattice inputs.

Features	Constraints		
	No Constraint	Sequence Constraint	Parser Constraint
Current Utt	54.4	42.0	41.5
+ DA	53.8	40.5	40.2
Current+Prev Utt	41.6	27.7	27.7
+DA	40.0	28.8	28.1
Current+3 Prev Utt	37.5	24.7	24.7
+DA	39.7	29.6	28.9

Table 6: Error rates for task structure prediction, with no constraints, sequence constraints and parser constraints

scoring sequence from the list for each dialog. The results are shown in Table 6. It can be seen that using the parsing constraint does not help the subtask label sequence prediction significantly. The chunk-based model gives almost the same accuracy, and is incremental and more efficient.

7 Discussion

The experiments reported in this section have been performed on transcribed speech. The audio for these dialogs, collected at a call center, were stored in a compressed format, so the speech recognition error rate is high. In future work, we will assess the performance of dialog structure prediction on recognized speech.

The research presented in this paper is but one step, albeit a crucial one, towards achieving the goal of inducing human-machine dialog systems using human-human dialogs. Dialog structure information is necessary for language generation (predicting the agents' response) and dialog state specific text-to-speech synthesis. However, there are several challenging problems that remain to be addressed.

The structuring of dialogs has another application in call center analytics. It is routine practice to monitor, analyze and mine call center data based on indicators such as the average length of dialogs, the task completion rate in order to estimate the efficiency of a call center. By incorporating structure to the dialogs, as presented in this paper, the analysis of dialogs can be performed at a more fine-grained (task and subtask) level.

8 Conclusions

In order to build a dialog manager using a data-driven approach, the following are necessary: a model for labeling/interpreting the user's current action; a model for identifying the current subtask/topic; and a model for predicting what the system's next action should be. Prior research in plan identification and in dialog act labeling has identified possible features for use in such models, but has not looked at the performance of different feature sets (reflecting different amounts of context and different views of dialog) across different domains (label sets). In this paper, we compared the performance of a dialog act labeler/predictor across three different tag sets: one using very detailed, domain-specific dialog acts usable for interpretation and generation; and two using general-purpose dialog acts and corpora available to the larger research community. We then compared two models for subtask labeling: a flat, chunk-based model and a hierarchical, parsing-based model. Findings include that simpler chunk-based models perform as well as hierarchical models for subtask labeling and that a dialog act feature is not helpful for subtask labeling.

In on-going work, we are using our best performing models for both DM and LG components (to predict the next dialog move(s), and to select the next system utterance). In future work, we will address the use of data-driven dialog management to improve SLU.

9 Acknowledgments

We thank Barbara Hollister and her team for their effort in annotating the dialogs for dialog acts and subtask structure. We thank Patrick Haffner for providing us with the LLAMA machine learning toolkit and Brian Roark for providing us with his top-down parser used in our experiments. We also thank Alistair Conkie, Mazin Gilbert, Narendra Gupta, and Benjamin Stern for discussions during the course of this work.

References

J. Alexandersson and N. Reithinger. 1997. Learning dialogue structures from a corpus. In *Proceedings of Eurospeech'97*.

S. Bangalore and N. Gupta. 2004. Extracting clauses in dialogue corpora : Application to spoken language understanding. *Journal Traitement Automatique des Langues (TAL)*, 45(2).

S. Bangalore and A. K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2).

J. Bear et al. 1992. Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog. In *Proceedings of ACL'92*.

A. Berger, S.D. Pietra, and V.D. Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

D. Bohus and A. Rudnicky. 2003. RavenClaw: Dialog management using hierarchical task decomposition and an expectation agenda. In *Proceedings of Eurospeech'03*.

J. Bos et al. 2003. DIPPER: Description and formalisation of an information-state update dialogue system architecture. In *Proceedings of SIGdial*.

H.H. Bui. 2003. A general model for online probabalistic plan recognition. In *Proceedings of IJCAI'03*.

S. Carberry. 2001. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1–2).

J. Carletta et al. 1997. The reliability of a dialog structure coding scheme. *Computational Linguistics*, 23(1).

E. Charniak and M. Johnson. 2001. Edit detection and parsing for transcribed speech. In *Proceedings of NAACL'01*.

M. Core. 1998. Analyzing and predicting patterns of DAMSL utterance tags. In *Proceedings of the AAAI spring symposium on Applying machine learning to discourse processing*.

M. Meteer et.al. 1995. Dysfluency annotation stylebook for the switchboard corpus. Distributed by LDC.

G. Di Fabbrizio and C. Lewis. 2004. Florence: a dialogue manager framework for spoken dialogue systems. In *ICSLP 2004, 8th International Conference on Spoken Language Processing*, Jeju, Jeju Island, Korea, October 4-8.

M. Frampton and O. Lemon. 2005. Reinforcement learning of dialogue strategies using the user's last dialogue act. In *Proceedings of the 4th IJCAI workshop on knowledge and reasoning in practical dialogue systems*.

M. Gilbert et al. 2005. Intelligent virtual agents for contact center automation. *IEEE Signal Processing Magazine*, 22(5), September.

B.J. Grosz and C.L. Sidner. 1986. Attention, intentions and the structure of discoursep. *Computational Linguistics*, 12(3).

P. Haffner. 2006. Scaling large margin classifiers for spoken language understanding. *Speech Communication*, 48(4).

H. Hardy et al. 2004. Data-driven strategies for an automated dialogue system. In *Proceedings of ACL'04*.

H. Wright Hastie et al. 2002. Automatically predicting dialogue structure using prosodic features. *Speech Communication*, 36(1–2).

J. Henderson et al. 2005. Hybrid reinforcement/supervised learning for dialogue policies from COMMUNICATOR data. In *Proceedings of the 4th IJCAI workshop on knowledge and reasoning in practical dialogue systems*.

A. K. Joshi. 1987. An introduction to tree adjoining grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam.

D. Jurafsky et al. 1998. Switchboard discourse language modeling project report. Technical Report Research Note 30, Center for Speech and Language Processing, Johns Hopkins University, Baltimore, MD.

S. Larsson et al. 1999. TrindiKit manual. Technical report, TRINDI Deliverable D2.2.

O. Lemon and A. Gruenstein. 2004. Multithreaded context for robust conversational interfaces: Context-sensitive speech recognition and interpretation of corrective fragments. *ACM Transactions on Computer-Human Interaction*, 11(3).

E. Levin and R. Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogue strategies. In *Proceedings of Eurospeech'97*.

D. Litman and J. Allen. 1987. A plan recognition model for subdialogs in conversations. *Cognitive Science*, 11(2).

K. Lochbaum. 1998. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4).

M. Poesio and A. Mikheev. 1998. The predictive power of game structure in dialogue act recognition: experimental results using maximum entropy estimation. In *Proceedings of ICSLP'98*.

D.V. Pynadath and M.P. Wellman. 2000. Probabilistic state-dependent grammars for plan recognition. In *In Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-2000)*.

C. Rich and C.L. Sidner. 1997. COLLAGEN: When agents collaborate with people. In *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*.

B. Roark. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 27(2).

K. Samuel et al. 1998. Computing dialogue acts from features with transformation-based learning. In *Proceedings of the AAAI spring symposium on Applying machine learning to discourse processing*.

K. Scheffler and S. Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of HLT'02*.

S. Seneff. 1992. A relaxation method for understanding spontaneous speech utterances. In *Proceedings of the Speech and Natural Language Workshop*, San Mateo, CA.

E. Shriberg et al. 2000. Prosody-based automatic segmentation of speech into sentences and topics. *Speech Communication*, 32, September.

C.L. Sidner. 1985. Plan parsing for intended response recognition in discourse. *Computational Intelligence*, 1(1).

S. Singh et al. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16.

A. Stolcke et al. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational Linguistics*, 26(3).

P. Taylor et al. 1998. Intonation and dialogue context as constraints for speech recognition. *Language and Speech*, 41(3).

J. Williams et al. 2005. Partially observable Markov decision processes with continuous observations for dialogue management. In *Proceedings of SIGdial*.

Semi-Supervised Conditional Random Fields for Improved Sequence Segmentation and Labeling

Feng Jiao
University of Waterloo

Shaojun Wang Chi-Hoon Lee
Russell Greiner Dale Schuurmans
University of Alberta

Abstract

We present a new semi-supervised training procedure for conditional random fields (CRFs) that can be used to train sequence segmentors and labelers from a combination of labeled and unlabeled training data. Our approach is based on extending the minimum entropy regularization framework to the structured prediction case, yielding a training objective that combines unlabeled conditional entropy with labeled conditional likelihood. Although the training objective is no longer concave, it can still be used to improve an initial model (e.g. obtained from supervised training) by iterative ascent. We apply our new training algorithm to the problem of identifying gene and protein mentions in biological texts, and show that incorporating unlabeled data improves the performance of the supervised CRF in this case.

1 Introduction

Semi-supervised learning is often touted as one of the most natural forms of training for language processing tasks, since unlabeled data is so plentiful whereas labeled data is usually quite limited or expensive to obtain. The attractiveness of semi-supervised learning for language tasks is further heightened by the fact that the models learned are large and complex, and generally even thousands of labeled examples can only sparsely cover the parameter space. Moreover, in complex *structured* prediction tasks, such as parsing or sequence modeling (part-of-speech tagging, word segmentation, named entity recognition, and so on), it is considerably more difficult to obtain labeled training data than for classification tasks (such as document classification), since hand-labeling individual words and word boundaries is much harder than assigning text-level class labels.

Many approaches have been proposed for semi-supervised learning in the past, including: generative models (Castelli and Cover 1996; Cohen and Cozman 2006; Nigam et al. 2000), self-learning (Celeux and Govaert 1992; Yarowsky 1995), co-training (Blum and Mitchell 1998), information-theoretic regularization (Corduneanu and Jaakkola 2006; Grandvalet and Bengio 2004), and graph-based transductive methods (Zhou et al. 2004; Zhou et al. 2005; Zhu et al. 2003). Unfortunately, these techniques have been developed primarily for single class label classification problems, or class label classification with a structured input (Zhou et al. 2004; Zhou et al. 2005; Zhu et al. 2003). Although still highly desirable, semi-supervised learning for structured classification problems like sequence segmentation and labeling have not been as widely studied as in the other semi-supervised settings mentioned above, with the sole exception of generative models.

With generative models, it is natural to include unlabeled data using an expectation-maximization approach (Nigam et al. 2000). However, generative models generally do not achieve the same accuracy as discriminatively trained models, and therefore it is preferable to focus on discriminative approaches. Unfortunately, it is far from obvious how unlabeled training data can be naturally incorporated into a discriminative training criterion. For example, unlabeled data simply cancels from the objective if one attempts to use a traditional conditional likelihood criterion. Nevertheless, recent progress has been made on incorporating unlabeled data in discriminative training procedures. For example, dependencies can be introduced between the labels of nearby instances and thereby have an effect on training (Zhu et al. 2003; Li and McCallum 2005; Altun et al. 2005). These models are trained to encourage nearby data points to have the same class label, and they can obtain impressive accuracy using a very small amount of labeled data. However, since they model pairwise similarities among data points, most of these approaches require joint inference over the whole data set at test time, which is not practical for large data sets.

In this paper, we propose a new semi-supervised training method for conditional random fields (CRFs) that incorporates both labeled and unlabeled sequence data to estimate a discriminative

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 209–216,
Sydney, July 2006. ©2006 Association for Computational Linguistics

structured predictor. CRFs are a flexible and powerful model for structured predictors based on undirected graphical models that have been globally conditioned on a set of input covariates (Lafferty et al. 2001). CRFs have proved to be particularly useful for sequence segmentation and labeling tasks, since, as conditional models of the labels given inputs, they relax the independence assumptions made by traditional generative models like hidden Markov models. As such, CRFs provide additional flexibility for using arbitrary overlapping features of the input sequence to define a structured conditional model over the output sequence, while maintaining two advantages: first, efficient dynamic program can be used for inference in both classification and training, and second, the training objective is concave in the model parameters, which permits global optimization.

To obtain a new semi-supervised training algorithm for CRFs, we extend the minimum entropy regularization framework of Grandvalet and Bengio (2004) to structured predictors. The resulting objective combines the likelihood of the CRF on labeled training data with its conditional entropy on unlabeled training data. Unfortunately, the maximization objective is no longer concave, but we can still use it to effectively improve an initial supervised model. To develop an effective training procedure, we first show how the derivative of the new objective can be computed from the covariance matrix of the features on the unlabeled data (combined with the labeled conditional likelihood). This relationship facilitates the development of an efficient dynamic programming for computing the gradient, and thereby allows us to perform efficient iterative ascent for training. We apply our new training technique to the problem of sequence labeling and segmentation, and demonstrate it specifically on the problem of identifying gene and protein mentions in biological texts. Our results show the advantage of semi-supervised learning over the standard supervised algorithm.

2 Semi-supervised CRF training

In what follows, we use the same notation as (Lafferty et al. 2001). Let $\mathbf{X}$ be a random variable over data sequences to be labeled, and $\mathbf{Y}$ be a random variable over corresponding label sequences. All components, $\mathbf{Y_i}$, of $\mathbf{Y}$ are assumed to range over a finite label alphabet $\mathcal{Y}$. For example, $\mathbf{X}$ might range over sentences and $\mathbf{Y}$ over part-of-speech taggings of those sentences; hence $\mathcal{Y}$ would be the set of possible part-of-speech tags in this case.

Assume we have a set of labeled examples, $\mathcal{D}^l = \Big((\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \cdots, (\mathbf{x}^{(N)}, \mathbf{y}^{(N)})\Big)$, and unlabeled examples, $\mathcal{D}^u = \Big(\mathbf{x}^{(N+1)}, \cdots, \mathbf{x}^{(M)}\Big)$. We would like to build a CRF model

$$\begin{aligned} p_\theta(\mathbf{y}|\mathbf{x}) &= \frac{1}{Z_\theta(\mathbf{x})} \exp\Big(\sum_{k=1}^{K} \theta_k f_k(\mathbf{x}, \mathbf{y})\Big) \\ &= \frac{1}{Z_\theta(\mathbf{x})} \exp\Big(\langle \theta, f(\mathbf{x}, \mathbf{y})\rangle\Big) \end{aligned}$$

over sequential input and output data $\mathbf{x}, \mathbf{y}$, where $\theta = (\theta_1, \cdots, \theta_K)^\top$, $f(\mathbf{x}, \mathbf{y}) = (f_1(\mathbf{x}, \mathbf{y}), \cdots, f_K(\mathbf{x}, \mathbf{y}))^\top$ and

$$Z_\theta(\mathbf{x}) = \sum_{\mathbf{y}} \exp\Big(\langle \theta, f(\mathbf{x}, \mathbf{y})\rangle\Big)$$

Our goal is to learn such a model from the combined set of labeled and unlabeled examples, $\mathcal{D}^l \cup \mathcal{D}^u$.

The standard supervised CRF training procedure is based upon maximizing the log conditional likelihood of the labeled examples in $\mathcal{D}^l$

$$CL(\theta) = \sum_{i=1}^{N} \log p_\theta(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) - U(\theta) \quad (1)$$

where $U(\theta)$ is any standard regularizer on θ, e.g. $U(\theta) = \|\theta\|^2/2$. Regularization can be used to limit over-fitting on rare features and avoid degeneracy in the case of correlated features. Obviously, (1) ignores the unlabeled examples in $\mathcal{D}^u$.

To make full use of the available training data, we propose a semi-supervised learning algorithm that exploits a form of *entropy regularization* on the unlabeled data. Specifically, for a semi-supervised CRF, we propose to maximize the following objective

$$\begin{aligned} RL(\theta) &= \sum_{i=1}^{N} \log p_\theta(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}) - U(\theta) \quad (2) \\ &+ \gamma \sum_{i=N+1}^{M} \sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) \log p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) \end{aligned}$$

where the first term is the penalized log conditional likelihood of the labeled data under the CRF, (1), and the second line is the negative conditional entropy of the CRF on the unlabeled data. Here, γ is a tradeoff parameter that controls the influence of the unlabeled data.

This approach resembles that taken by (Grandvalet and Bengio 2004) for single variable classification, but here applied to structured CRF training. The motivation is that minimizing conditional entropy over unlabeled data encourages the algorithm to find putative labelings for the unlabeled data that are mutually reinforcing with the supervised labels; that is, greater certainty on the putative labelings coincides with greater conditional likelihood on the supervised labels, and vice versa. For a single classification variable this criterion has been shown to effectively partition unlabeled data into clusters (Grandvalet and Bengio 2004; Roberts et al. 2000).

To motivate the approach in more detail, consider the overlap between the probability distribution over a label sequence y and the empirical distribution of $\tilde{p}(\mathbf{x})$ on the unlabeled data $\mathcal{D}^u$. The overlap can be measured by the Kullback-Leibler divergence $D(p_\theta(\mathbf{y}|\mathbf{x})\tilde{p}(\mathbf{x})\|\tilde{p}(\mathbf{x}))$. It is well known that Kullback-Leibler divergence (Cover and Thomas 1991) is positive and increases as the overlap between the two distributions decreases. In other words, maximizing Kullback-Leibler divergence implies that the overlap between two distributions is minimized. The total overlap over all possible label sequences can be defined as

$$\begin{aligned}
&\sum_{\mathbf{y}} D(p_\theta(\mathbf{y}|\mathbf{x})\tilde{p}(\mathbf{x})\|\tilde{p}(\mathbf{x})) \\
&= \sum_{\mathbf{y}} \sum_{\mathbf{x}\in\mathcal{D}^u} p_\theta(\mathbf{y}|\mathbf{x})\tilde{p}(\mathbf{x}) \log \frac{p_\theta(\mathbf{y}|\mathbf{x})\tilde{p}(\mathbf{x})}{\tilde{p}(\mathbf{x})} \\
&= \sum_{\mathbf{x}\in\mathcal{D}^u} \tilde{p}(\mathbf{x}) \sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}) \log p_\theta(\mathbf{y}|\mathbf{x})
\end{aligned}$$

which motivates the negative entropy term in (2).

The combined training objective (2) exploits unlabeled data to improve the CRF model, as we will see. However, one drawback with this approach is that the entropy regularization term is not concave. To see why, note that the entropy regularizer can be seen as a composition, $h(\theta) = f(g(\theta))$, where $f : \Re^{|\mathcal{Y}|} \to \Re$, $f(g) = \sum_{\mathbf{y}} g_{\mathbf{y}} \log g_{\mathbf{y}}$ and $g_{\mathbf{y}} : \Re^K \to \Re$, $g_{\mathbf{y}}(\theta) = \frac{1}{Z_\theta(\mathbf{x})} \exp\left(\sum_{k=1}^K \theta_k f_k(\mathbf{x},\mathbf{y})\right)$. For scalar θ, the second derivative of a composition, $h = f \circ g$, is given by (Boyd and Vandenberghe 2004)

$$h''(\theta) = g'(\theta)^\top \nabla^2 f(g(\theta)) g'(\theta) + \nabla f(g(\theta))^\top g''(\theta)$$

Although f and g_y are concave here, since f is not nondecreasing, h is not necessarily concave. So in general there are local maxima in (2).

3 An efficient training procedure

As (2) is not concave, many of the standard global maximization techniques do not apply. However, one can still use unlabeled data to improve a supervised CRF via iterative ascent. To derive an efficient iterative ascent procedure, we need to compute gradient of (2) with respect to the parameters θ. Taking derivative of the objective function (2) with respect to θ yields Appendix A for the derivation)

$$\begin{aligned}
&\frac{\partial}{\partial\theta} RL(\theta) \qquad (3)\\
&= \sum_{i=1}^N \left(f(\mathbf{x}^{(i)},\mathbf{y}^{(i)}) - \sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f(\mathbf{x}^{(i)},\mathbf{y}^{(i)}) \right) \\
&- \frac{\partial}{\partial\theta} U(\theta) \; + \; \gamma \sum_{i=N+1}^M \operatorname{cov}_{p_\theta(\mathbf{y}|\mathbf{x}^{(i)})} \left[f(\mathbf{x}^{(i)},\mathbf{y}) \right] \theta
\end{aligned}$$

The first three items on the right hand side are just the standard gradient of the CRF objective, $\partial CL(\theta)/\partial\theta$ (Lafferty et al. 2001), and the final item is the gradient of the entropy regularizer (the derivation of which is given in Appendix A.

Here, $\operatorname{cov}_{p_\theta(\mathbf{y}|\mathbf{x}^{(i)})}\left[f(\mathbf{x}^{(i)},\mathbf{y})\right]$ is the conditional covariance matrix of the features, $f_j(\mathbf{x},\mathbf{y})$, given sample sequence $\mathbf{x}^{(i)}$. In particular, the (j,k)th element of this matrix is given by

$$\begin{aligned}
&\operatorname{cov}_{p_\theta(\mathbf{y}|\mathbf{x})}\left[f_j(\mathbf{x},\mathbf{y}) f_k(\mathbf{x},\mathbf{y}) \right] \\
&= \mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(f_j(\mathbf{x},\mathbf{y}) f_k(\mathbf{x},\mathbf{y}) \Big) \\
&\quad - \mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(f_j(\mathbf{x},\mathbf{y}) \Big) \mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(f_k(\mathbf{x},\mathbf{y}) \Big) \\
&= \sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}) \Big(f_j(\mathbf{x},\mathbf{y}) f_k(\mathbf{x},\mathbf{y}) \Big) \qquad (4)\\
&- \Big(\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}) f_j(\mathbf{x},\mathbf{y}) \Big) \Big(\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}) f_k(\mathbf{x},\mathbf{y}) \Big)
\end{aligned}$$

To efficiently calculate the gradient, we need to be able to efficiently compute the expectations with respect to $\mathbf{y}$ in (3) and (4). However, this can pose a challenge in general, because there are exponentially many values for $\mathbf{y}$. Techniques for computing the *linear* feature expectations in (3) are already well known if $\mathbf{y}$ is sufficiently structured (e.g. $\mathbf{y}$ forms a Markov chain) (Lafferty et al. 2001). However, we now have to develop efficient techniques for computing the *quadratic* feature expectations in (4).

For the quadratic feature expectations, first note that the diagonal terms, $j = l$, are straightforward, since each feature is an indicator, we have that $f_j(\mathbf{x},\mathbf{y})^2 = f_j(\mathbf{x},\mathbf{y})$, and therefore the diagonal terms in the conditional covariance are just linear feature expectations $\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(f_j(\mathbf{x},\mathbf{y})^2\Big) = \mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(f_j(\mathbf{x},\mathbf{y})\Big)$ as before.

For the off diagonal terms, $j \neq l$, however, we need to develop a new algorithm. Fortunately, for structured label sequences, $\mathbf{Y}$, one can devise an efficient algorithm for calculating the quadratic expectations based on nested dynamic programming. To illustrate the idea, we assume that the dependencies of $\mathbf{Y}$, conditioned on $\mathbf{X}$, form a *Markov chain*.

Define one feature for each state pair (y', y), and one feature for each state-observation pair (y, x), which we express with indicator functions $f_{y',y}(\langle u,v\rangle, \mathbf{y}|_{\langle u,v\rangle}, \mathbf{x}) = \delta(\mathbf{y}_u, y')\delta(\mathbf{y}_v, y)$ and $g_{y,x}(v, \mathbf{y}|_v, \mathbf{x}) = \delta(\mathbf{y}_v, y)\delta(\mathbf{x}_v, x)$ respectively. Following (Lafferty et al. 2001), we also add special start and stop states, $\mathbf{Y}_0 = \text{start}$ and $\mathbf{Y}_{n+1} = \text{stop}$. The conditional probability of a label sequence can now be expressed concisely in a matrix form. For each position j in the observation sequence $\mathbf{x}$, define the $|\mathcal{Y}| \times |\mathcal{Y}|$ matrix random variable $M_j(\mathbf{x}) = [M_j(y', y|\mathbf{x})]$ by

$$\begin{aligned} M_j(y', y|\mathbf{x}) &= \exp(\Lambda_j(y', y|\mathbf{x})) \quad \text{where} \\ \Lambda_j(y', y|\mathbf{x}) &= \sum_k \lambda_k f_k\left(e_j, \mathbf{y}|_{e_j} = (y', y), \mathbf{x}\right) \\ &\quad + \sum_k \mu_k g_k\left(v_j, \mathbf{y}|_{v_j} = y, \mathbf{x}\right) \end{aligned}$$

Here e_j is the edge with labels $(\mathbf{Y}_{j-1}, \mathbf{Y}_j)$ and v_j is the vertex with label $\mathbf{Y}_j$.

For each index $j = 0, \cdots, n+1$ define the forward vectors $\alpha_j(\mathbf{x})$ with base case

$$\alpha_0(y|\mathbf{x}) = \begin{cases} 1 & \text{if } y = \text{start} \\ 0 & \text{otherwise} \end{cases}$$

and recurrence

$$\alpha_j(\mathbf{x}) = \alpha_{j-1}(\mathbf{x}) M_j(\mathbf{x})$$

Similarly, the backward vectors $\beta_j(\mathbf{x})$ are given by

$$\begin{aligned} \beta_{n+1}(y|\mathbf{x}) &= \begin{cases} 1 & \text{if } y = \text{stop} \\ 0 & \text{otherwise} \end{cases} \\ \beta_j(\mathbf{x}) &= M_{j+1}(\mathbf{x})\beta_{j+1}(\mathbf{x}) \end{aligned}$$

With these definitions, the expectation of the product of each pair of feature functions, $(f_j(\mathbf{x},\mathbf{y}), f_k(\mathbf{x},\mathbf{y}))$, $(f_j(\mathbf{x},\mathbf{y}), g_k(\mathbf{x},\mathbf{y}))$, and $(g_j(\mathbf{x},\mathbf{y}), g_k(\mathbf{x},\mathbf{y}))$, for $j,k = 1,\cdots,K$, $j \neq k$, can be recursively calculated.

First define the summary matrix

$$M_{s+1,t-1}(y, y'|\mathbf{x}) = \Big(\prod_{l=s+1}^{t-1} M_l(\mathbf{x})\Big)_{y,y'}$$

Then the quadratic feature expectations can be computed by the following recursion, where the two double sums in each expectation correspond to the two cases depending on which feature occurs first (e_s occuring before e_t).

$$\begin{aligned} &\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(f_j(\mathbf{x},\mathbf{y}) f_k(\mathbf{x},\mathbf{y})\Big) \\ &= \sum_{\mathbf{x}\in\mathcal{D}^u} \sum_{s,t=1,s<t}^{n+1} \sum_{y',y} f_j\left(e_s, \mathbf{y}|_{e_s} = (y', y), \mathbf{x}\right) \\ &\quad \sum_{y'',y'''} f_k\left(e_t, \mathbf{y}|_{e_t} = (y'', y'''), \mathbf{x}\right) \\ &\quad \alpha_{s-1}(y'|\mathbf{x}) M_s(y', y|\mathbf{x}) M_{s+1,t-1}(y, y''|\mathbf{x}) \\ &\quad M_t(y'', y'''|\mathbf{x}) \beta_t(y'''|\mathbf{x}) / Z_\theta(\mathbf{x}) \\ &\quad + \sum_{\mathbf{x}\in\mathcal{D}^u} \sum_{s,t=1,t<s}^{n+1} \sum_{y',y} f_j\left(e_t, \mathbf{y}|_{e_t} = (y', y), \mathbf{x}\right) \\ &\quad \sum_{y'',y'''} f_k\left(e_s, \mathbf{y}|_{e_s} = (y'', y'''), \mathbf{x}\right) \\ &\quad \alpha_{t-1}(y'''|\mathbf{x}) M_t(y''', y''|\mathbf{x}) M_{t+1,s-1}(y'', y'|\mathbf{x}) \\ &\quad M_s(y', y|\mathbf{x}) \beta_t(y|\mathbf{x}) / Z_\theta(\mathbf{x}) \end{aligned}$$

$$\begin{aligned} &\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(f_j(\mathbf{x},\mathbf{y}) g_k(\mathbf{x},\mathbf{y})\Big) \\ &= \sum_{\mathbf{x}\in\mathcal{D}^u} \sum_{s,t=1,s\leq t}^{n+1} \sum_{y',y} f_j\left(e_s, \mathbf{y}|_{e_s} = (y', y), \mathbf{x}\right) \\ &\sum_{y''} g_k\left(v_t, \mathbf{y}|_{v_t} = y'', \mathbf{x}\right) \alpha_{s-1}(y'|\mathbf{x}) M_s(y', y|\mathbf{x}) \\ &\quad M_{s+1,t-1}(y, y''|x) \beta_t(y''|\mathbf{x}) / Z_\theta(\mathbf{x}) \\ &\quad + \sum_{x\in\mathcal{D}^u} \sum_{s,t=1,t<s}^{n+1} \sum_{y',y} f_j\left(e_t, \mathbf{y}|_{e_t} = (y', y), \mathbf{x}\right) \\ &\quad \sum_{y''} g_k\left(v_s, \mathbf{y}|_{v_s} = y'', \mathbf{x}\right) \alpha_{t-1}(y''|\mathbf{x}) \\ &\quad M_{t+1,s-1}(y'', y'|x) M_s(y', y|\mathbf{x}) \beta_t(y|\mathbf{x}) / Z_\theta(\mathbf{x}) \end{aligned}$$

$$\begin{aligned} &\mathrm{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\Big(g_j(\mathbf{x},\mathbf{y}) g_k(\mathbf{x},\mathbf{y})\Big) \\ &= \sum_{\mathbf{x}\in\mathcal{D}^u} \sum_{s,t=1,s<t}^{n+1} \\ &\quad \sum_{y'} g_j\left(v_s, \mathbf{y}|_{v_s} = y', \mathbf{x}\right) \sum_{y} g_k\left(v_t, \mathbf{y}|_{v_t} = y, \mathbf{x}\right) \end{aligned}$$

$$\frac{\alpha_{s-1}(y'|\mathbf{x})M_{s+1,t-1}(y',y|\mathbf{x})\beta_t(y|\mathbf{x})}{Z_\theta(\mathbf{x})}$$

$$+\sum_{\mathbf{x}\in\mathcal{D}^u}\sum_{s,t=1,t<s}^{n+1}$$

$$\sum_{y'} g_j\left(v_t,\mathbf{y}|_{v_t}=y',\mathbf{x}\right)\sum_{y'} g_k\left(v_s,\mathbf{y}|_{v_s}=y,\mathbf{x}\right)$$

$$\frac{\alpha_{t-1}(y|\mathbf{x})M_{t+1,s-1}(y,y'|\mathbf{x})\beta_t(y'|\mathbf{x})}{Z_\theta(\mathbf{x})}$$

The computation of these expectations can be organized in a trellis, as illustrated in Figure 1.

Once we obtain the gradient of the objective function (2), we use limited-memory L-BFGS, a quasi-Newton optimization algorithm (McCallum 2002; Nocedal and Wright 2000), to find the local maxima with the initial value being set to be the optimal solution of the supervised CRF on labeled data.

4 Time and space complexity

The time and space complexity of the semi-supervised CRF training procedure is greater than that of standard supervised CRF training, but nevertheless remains a small degree polynomial in the size of the training data. Let

m_l = size of the labeled set
m_u = size of the unlabeled set
n_l = labeled sequence length
n_u = unlabeled sequence length
n_t = test sequence length
s = number of states
c = number of training iterations.

Then the time required to classify a test sequence is $O(n_t s^2)$, independent of training method, since the Viterbi decoder needs to access each path.

For training, supervised CRF training requires $O(cm_l n_l s^2)$ time, whereas semi-supervised CRF training requires $O(cm_l n_l s^2 + cm_u n_u^2 s^3)$ time. The additional cost for semi-supervised training arises from the extra nested loop required to calculated the quadratic feature expectations, which introduces in an additional $n_u s$ factor.

However, the space requirements of the two training methods are the same. That is, even though the covariance matrix has size $O(K^2)$, there is never any need to store the entire matrix in memory. Rather, since we only need to compute the product of the covariance with θ, the calculation can be performed iteratively without using extra space beyond that already required by supervised CRF training.

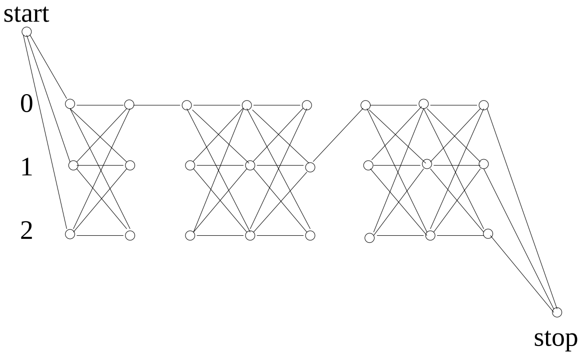


Figure 1: Trellis for computing the expectation of a feature product over a pair of feature functions, f_{00} vs f_{10}, where the feature f_{00} occurs first. This leads to one double sum.

5 Identifying gene and protein mentions

We have developed our new semi-supervised training procedure to address the problem of information extraction from biomedical text, which has received significant attention in the past few years. We have specifically focused on the problem of identifying explicit mentions of gene and protein names (McDonald and Pereira 2005). Recently, McDonald and Pereira (2005) have obtained interesting results on this problem by using a standard supervised CRF approach. However, our contention is that stronger results could be obtained in this domain by exploiting a large corpus of unannotated biomedical text to improve the quality of the predictions, which we now show.

Given a biomedical text, the task of identifying gene mentions can be interpreted as a tagging task, where each word in the text can be labeled with a tag that indicates whether it is the beginning of gene mention (B), the continuation of a gene mention (I), or outside of any gene mention (O). To compare the performance of different taggers learned by different mechanisms, one can measure the precision, recall and F-measure, given by

$$\begin{aligned}
\textit{precision} &= \frac{\textit{\# correct predictions}}{\textit{\# predicted gene mentions}}\\
\textit{recall} &= \frac{\textit{\# correct predictions}}{\textit{\# true gene mentions}}\\
\textit{F-measure} &= \frac{2\times\textit{precision}\times\textit{recall}}{\textit{precision}+\textit{recall}}
\end{aligned}$$

In our evaluation, we compared the proposed semi-supervised learning approach to the state of the art supervised CRF of McDonald and Pereira (2005), and also to self-training (Celeux and Govaert 1992; Yarowsky 1995), using the same feature set as (McDonald and Pereira 2005). The CRF training procedures, supervised and semi-supervised, were run with the same regularization function, $U(\theta) = \|\theta\|^2/2$, used in (McDonald and Pereira 2005).

First we evaluated the performance of the semi-supervised CRF in detail, by varying the ratio between the amount of labeled and unlabeled data, and also varying the tradeoff parameter γ. We choose a labeled training set A consisting of 5448 words, and considered alternative unlabeled training sets, B (5210 words), C (10,208 words), and D (25,145 words), consisting of the same, 2 times and 5 times as many sentences as A respectively. All of these sets were disjoint and selected randomly from the full corpus, the smaller one in (McDonald et al. 2005), consisting of 184,903 words in total. To determine sensitivity to the parameter γ we examined a range of discrete values $0, 0.1, 0.5, 1, 5, 10, 20, 50$.

In our first experiment, we train the CRF models using labeled set A and unlabeled sets B, C and D respectively. Then test the performance on the sets B, C and D respectively The results of our evaluation are shown in Table 1. The performance of the supervised CRF algorithm, trained only on the labeled set A, is given on the first row in Table 1 (corresponding to $\gamma = 0$). By comparison, the results obtained by the semi-supervised CRFs on the held-out sets B, C and D are given in Table 1 by increasing the value of γ.

The results of this experiment demonstrate quite clearly that in most cases the semi-supervised CRF obtains higher precision, recall and F-measure than the fully supervised CRF, yielding a 20% improvement in the best case.

In our second experiment, again we train the CRF models using labeled set A and unlabeled sets B, C and D respectively with increasing values of γ, but we test the performance on the held-out set E which is the full corpus minus the labeled set A and unlabeled sets B, C and D. The results of our evaluation are shown in Table 2 and Figure 2. The blue line in Figure 2 is the result of the supervised CRF algorithm, trained only on the labeled set A. In particular, by using the supervised CRF model, the system predicted 3334 out of 7472 gene mentions, of which 2435 were correct, resulting in a precision of 0.73, recall of 0.33 and F-measure of 0.45. The other curves are those of the semi-supervised CRFs.

The results of this experiment demonstrate quite clearly that the semi-supervised CRFs simultaneously increase both the number of predicted gene mentions and the number of correct predictions, thus the precision remains almost the same as the supervised CRF, and the recall increases significantly.

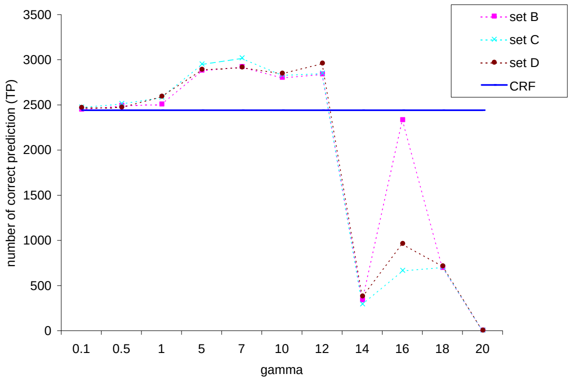


Figure 2: Performance of the supervised and semi-supervised CRFs. The sets B, C and D refer to the unlabeled training set used by the semi-supervised algorithm.

Both experiments as illustrated in Figure 2 and Tables 1 and 2 show that clearly better results are obtained by incorporating additional unlabeled training data, even when evaluating on disjoint testing data (Figure 2). The performance of the semi-supervised CRF is not overly sensitive to the tradeoff parameter γ, except that γ cannot be set too large.

5.1 Comparison to self-training

For completeness, we also compared our results to the self-learning algorithm, which has commonly been referred to as bootstrapping in natural language processing and originally popularized by the work of Yarowsky in word sense disambiguation (Abney 2004; Yarowsky 1995). In fact, similar ideas have been developed in pattern recognition under the name of the decision-directed algorithm (Duda and Hart 1973), and also traced back to 1970s in the EM literature (Celeux and Govaert 1992). The basic algorithm works as follows:

1. Given $\mathcal{D}^l$ and $\mathcal{D}^u$, begin with a seed set of labeled examples, $\mathcal{D}^{(0)}$, chosen from $\mathcal{D}^l$.
2. For $m = 0, 1, \cdots$
   (a) Train the supervised CRF on labeled examples $\mathcal{D}^{(m)}$, obtaining $\theta^{(m)}$.
   (b) For each sequence $\mathbf{x}^{(i)} \in \mathcal{D}^u$, find $\mathbf{y}^{(i)}_{(m)} = \arg\max_{\mathbf{y}} p_{\theta^{(m)}}(\mathbf{y}|\mathbf{x}^{(i)})$ via Viterbi decoding or other inference algorithm, and add the pair $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}_{(m)})$ to the set of labeled examples (replacing any previous label for $\mathbf{x}^{(i)}$ if present).

Table 1: Performance of the semi-supervised CRFs obtained on the held-out sets B, C and D

	Test Set B, Trained on A and B			Test Set C, Trained on A and C			Test Set D, Trained on A and D		
γ	Precision	Recall	F-Measure	Precision	Recall	F-Measure	Precision	Recall	F-Measure
0	0.80	0.36	0.50	0.77	0.29	0.43	0.74	0.30	0.43
0.1	0.82	0.4	0.54	0.79	0.32	0.46	0.74	0.31	0.44
0.5	0.82	0.4	0.54	0.79	0.33	0.46	0.74	0.31	0.44
1	0.82	0.4	0.54	0.77	0.34	0.47	0.73	0.33	0.45
5	0.84	0.45	0.59	0.78	0.38	0.51	0.72	0.36	0.48
10	0.78	0.46	0.58	0.66	0.38	0.48	0.66	0.38	0.47

Table 2: Performance of the semi-supervised CRFs trained by using unlabeled sets B, C and D

	Test Set E, Trained on A and B		Test Set E, Trained on A and C		Test Set E, Trained on A and D	
γ	# predicted	# correct prediction	# predicted	# correct prediction	# predicted	# correct prediction
0.1	3345	2446	3376	2470	3366	2466
0.5	3413	2489	3450	2510	3376	2469
1	3446	2503	3588	2580	3607	2590
5	4089	2878	4206	2947	4165	2888
10	4450	2799	4762	2827	4778	2845

(c) If for each $\mathbf{x}^{(i)} \in \mathcal{D}^u$, $\mathbf{y}^{(i)}_{(m)} = \mathbf{y}^{(i)}_{(m-1)}$, stop; otherwise $m = m + 1$, iterate.

We implemented this self training approach and tried it in our experiments. Unfortunately, we were not able to obtain any improvements over the standard supervised CRF with self-learning, using the sets $\mathcal{D}^l = A$, and $\mathcal{D}^u \in \{B, C, D\}$. The semi-supervised CRF remains the best of the approaches we have tried on this problem.

6 Conclusions and further directions

We have presented a new semi-supervised training algorithm for CRFs, based on extending minimum conditional entropy regularization to the structured prediction case. Our approach is motivated by the information-theoretic argument (Grandvalet and Bengio 2004; Roberts et al. 2000) that unlabeled examples can provide the most benefit when classes have small overlap. An iterative ascent optimization procedure was developed for this new criterion, which exploits a nested dynamic programming approach to efficiently compute the covariance matrix of the features.

We applied our new approach to the problem of identifying gene name occurrences in biological text, exploiting the availability of auxiliary unlabeled data to improve the performance of the state of the art supervised CRF approach in this domain. Our semi-supervised CRF approach shares all of the benefits of the standard CRF training, including the ability to exploit arbitrary features of the inputs, while obtaining improved accuracy through the use of unlabeled data. The main drawback is that training time is increased because of the extra nested loop needed to calculate feature covariances. Nevertheless, the algorithm is sufficiently efficient to be trained on unlabeled data sets that yield a notable improvement in classification accuracy over standard supervised training. To further accelerate the training process of our semi-supervised CRFs, we may apply stochastic gradient optimization method with adaptive gain adjustment as proposed by Vishwanathan et al. (2006).

Acknowledgments

Research supported by Genome Alberta, Genome Canada, and the Alberta Ingenuity Centre for Machine Learning.

References

S. Abney. (2004). Understanding the Yarowsky algorithm. *Computational Linguistics*, 30(3):365-395.

Y. Altun, D. McAllester and M. Belkin. (2005). Maximum margin semi-supervised learning for structured variables. *Advances in Neural Information Processing Systems 18*.

A. Blum and T. Mitchell. (1998). Combining labeled and unlabeled data with co-training. *Proceedings of the Workshop on Computational Learning Theory*, 92-100.

S. Boyd and L. Vandenberghe. (2004). *Convex Optimization*. Cambridge University Press.

V. Castelli and T. Cover. (1996). The relative value of labeled and unlabeled samples in pattern recognition with an unknown mixing parameter. *IEEE Trans. on Information Theory*, 42(6):2102-2117.

G. Celeux and G. Govaert. (1992). A classification EM algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*, 14:315-332.

I. Cohen and F. Cozman. (2006). Risks of semi-supervised learning. *Semi-Supervised Learning*, O. Chapelle, B. Scholköpf and A. Zien, (Editors), 55-70, MIT Press.

A. Corduneanu and T. Jaakkola. (2006). Data dependent regularization. *Semi-Supervised Learning*, O. Chapelle, B. Scholköpf and A. Zien, (Editors), 163-182, MIT Press.

T. Cover and J. Thomas, (1991). *Elements of Information Theory*, John Wiley & Sons.

R. Duda and P. Hart. (1973). *Pattern Classification and Scene Analysis*, John Wiley & Sons.

Y. Grandvalet and Y. Bengio. (2004). Semi-supervised learning by entropy minimization, *Advances in Neural Information Processing Systems*, 17:529-536.

J. Lafferty, A. McCallum and F. Pereira. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. *Proceedings of the 18th International Conference on Machine Learning*, 282-289.

W. Li and A. McCallum. (2005). Semi-supervised sequence modeling with syntactic topic models. *Proceedings of Twentieth National Conference on Artificial Intelligence*, 813-818.

A. McCallum. (2002). MALLET: A machine learning for language toolkit. [http://mallet.cs.umass.edu]

R. McDonald, K. Lerman and Y. Jin. (2005). Conditional random field biomedical entity tagger. [http://www.seas.upenn.edu/~sryantm/software/BioTagger/]

R. McDonald and F. Pereira. (2005). Identifying gene and protein mentions in text using conditional random fields. *BMC Bioinformatics 2005*, 6(Suppl 1):S6.

K. Nigam, A. McCallum, S. Thrun and T. Mitchell. (2000). Text classification from labeled and unlabeled documents using EM. *Machine learning*. 39(2/3):135-167.

J. Nocedal and S. Wright. (2000). *Numerical Optimization*, Springer.

S. Roberts, R. Everson and I. Rezek. (2000). Maximum certainty data partitioning. *Pattern Recognition*, 33(5):833-839.

S. Vishwanathan, N. Schraudolph, M. Schmidt and K. Murphy. (2006). Accelerated training of conditional random fields with stochastic meta-descent. *Proceedings of the 23th International Conference on Machine Learning*.

D. Yarowsky. (1995). Unsupervised word sense disambiguation rivaling supervised methods. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, 189-196.

D. Zhou, O. Bousquet, T. Navin Lal, J. Weston and B. Schölkopf. (2004). Learning with local and global consistency. *Advances in Neural Information Processing Systems*, 16:321-328.

D. Zhou, J. Huang and B. Schölkopf. (2005). Learning from labeled and unlabeled data on a directed graph. *Proceedings of the 22nd International Conference on Machine Learning*, 1041-1048.

X. Zhu, Z. Ghahramani and J. Lafferty. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. *Proceedings of the 20th International Conference on Machine Learning*, 912-919.

A Deriving the gradient of the entropy

We wish to show that

$$\begin{aligned}
&\frac{\partial}{\partial\theta}\left(\sum_{i=N+1}^{M}\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\log p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\right) \\
&\quad = \sum_{i=N+1}^{M}\mathrm{cov}_{p_\theta(\mathbf{y}|\mathbf{x}^{(i)})}\left[f(\mathbf{x}^{(i)},\mathbf{y})\right]\theta \qquad (5)
\end{aligned}$$

First, note that some simple calculation yields

$$\frac{\partial \log Z_\theta(\mathbf{x}^{(i)})}{\partial\theta_j} = \sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})$$

and

$$\begin{aligned}
\frac{\partial p_\theta(\mathbf{y}|\mathbf{x}^{(i)})}{\partial\theta_j} &= \frac{\partial}{\partial\theta_j}\left(\frac{\exp\left(\langle\theta, f(\mathbf{x}^{(i)},\mathbf{y})\rangle\right)}{Z_\theta(\mathbf{x}^{(i)})}\right) \\
&= p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y}) \\
&\quad - p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})
\end{aligned}$$

Therefore

$$\begin{aligned}
&\frac{\partial}{\partial\theta_j}\left(\sum_{i=N+1}^{M}\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\log p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\right) \\
&= \sum_{i=N+1}^{M}\frac{\partial}{\partial\theta_j}\left(\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\langle\theta, f(\mathbf{x}^{(i)},\mathbf{y})\rangle\right. \\
&\qquad\qquad \left. - \log Z_\theta(\mathbf{x}^{(i)})\right) \\
&= \sum_{i=N+1}^{M}\left(\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})\right. \\
&\qquad + \sum_{\mathbf{y}}\frac{\partial p_\theta(\mathbf{y}|\mathbf{x}^{(i)})}{\partial\theta_j}\langle\theta, f(\mathbf{x}^{(i)},\mathbf{y})\rangle \\
&\qquad \left. - \sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})\right) \\
&= \sum_{i=N+1}^{M}\left(\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})\langle\theta, f(\mathbf{x}^{(i)},\mathbf{y})\rangle\right. \\
&\qquad - [\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)})\langle\theta, f(\mathbf{x}^{(i)},\mathbf{y})\rangle] \\
&\qquad \left. [\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})]\right) \\
&= \sum_{i=N+1}^{M}\left(\sum_{k}\theta_k\left[\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y}) f_k(\mathbf{x}^{(i)},\mathbf{y})\right.\right. \\
&\qquad - [\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_k(\mathbf{x}^{(i)},\mathbf{y})] \\
&\qquad \left.\left. [\sum_{\mathbf{y}} p_\theta(\mathbf{y}|\mathbf{x}^{(i)}) f_j(\mathbf{x}^{(i)},\mathbf{y})]\right]\right)
\end{aligned}$$

In the vector form, this can be written as (5)

Training Conditional Random Fields with Multivariate Evaluation Measures

Jun Suzuki, Erik McDermott and Hideki Isozaki
NTT Communication Science Laboratories, NTT Corp.
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
{jun, mcd, isozaki}@cslab.kecl.ntt.co.jp

Abstract

This paper proposes a framework for training Conditional Random Fields (CRFs) to optimize multivariate evaluation measures, including non-linear measures such as F-score. Our proposed framework is derived from an error minimization approach that provides a simple solution for directly optimizing *any* evaluation measure. Specifically focusing on sequential segmentation tasks, i.e. text chunking and named entity recognition, we introduce a loss function that closely reflects the target evaluation measure for these tasks, namely, segmentation F-score. Our experiments show that our method performs better than standard CRF training.

1 Introduction

Conditional random fields (CRFs) are a recently introduced formalism (Lafferty et al., 2001) for representing a conditional model $p(\boldsymbol{y}|\boldsymbol{x})$, where both a set of inputs, $\boldsymbol{x}$, and a set of outputs, $\boldsymbol{y}$, display non-trivial interdependency. CRFs are basically defined as a discriminative model of Markov random fields conditioned on inputs (observations) $\boldsymbol{x}$. Unlike generative models, CRFs model only the output $\boldsymbol{y}$'s distribution over $\boldsymbol{x}$. This allows CRFs to use flexible features such as complicated functions of multiple observations. The modeling power of CRFs has been of great benefit in several applications, such as shallow parsing (Sha and Pereira, 2003) and information extraction (McCallum and Li, 2003).

Since the introduction of CRFs, intensive research has been undertaken to boost their effectiveness. The first approach to estimating CRF parameters is the *maximum likelihood (ML)* criterion over conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ itself (Lafferty et al., 2001). The ML criterion, however, is prone to over-fitting the training data, especially since CRFs are often trained with a very large number of correlated features. The *maximum a posteriori (MAP)* criterion over parameters, $\boldsymbol{\lambda}$, given $\boldsymbol{x}$ and $\boldsymbol{y}$ is the natural choice for reducing over-fitting (Sha and Pereira, 2003). Moreover, the Bayes approach, which optimizes both MAP and the prior distribution of the parameters, has also been proposed (Qi et al., 2005). Furthermore, large margin criteria have been employed to optimize the model parameters (Taskar et al., 2004; Tsochantaridis et al., 2005).

These training criteria have yielded excellent results for various tasks. However, real world tasks are evaluated by task-specific evaluation measures, including non-linear measures such as F-score, while all of the above criteria achieve optimization based on the linear combination of average accuracies, or error rates, rather than a given task-specific evaluation measure. For example, *sequential segmentation tasks (SSTs)*, such as text chunking and named entity recognition, are generally evaluated with the *segmentation F-score*. This inconsistency between the objective function during training and the task evaluation measure might produce a suboptimal result.

In fact, to overcome this inconsistency, an SVM-based multivariate optimization method has recently been proposed (Joachims, 2005). Moreover, an F-score optimization method for logistic regression has also been proposed (Jansche, 2005). In the same spirit as the above studies, we first propose a generalization framework for CRF training that allows us to optimize directly not only the error rate, but also any evaluation measure. In other words, our framework can incorporate any evaluation measure of interest into the loss function and then optimize this loss function as the training objective function. Our proposed framework is fundamentally derived from an approach to (smoothed) error rate minimization well

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 217–224,
Sydney, July 2006. ©2006 Association for Computational Linguistics

known in the speech and pattern recognition community, namely the *Minimum Classification Error (MCE)* framework (Juang and Katagiri, 1992). The framework of MCE criterion training supports the theoretical background of our method. The approach proposed here subsumes the conventional ML/MAP criteria training of CRFs, as described in the following.

After describing the new framework, as an example of optimizing multivariate evaluation measures, we focus on SSTs and introduce a segmentation F-score loss function for CRFs.

2 CRFs and Training Criteria

Given an input (observation) $\boldsymbol{x} \in \mathcal{X}$ and parameter vector $\boldsymbol{\lambda} = \{\lambda_1, \ldots, \lambda_M\}$, CRFs define the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ of a particular output $\boldsymbol{y} \in \mathcal{Y}$ as being proportional to a product of potential functions on the cliques of a graph, which represents the interdependency of $\boldsymbol{y}$ and $\boldsymbol{x}$. That is:

$$p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda}) = \frac{1}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x})} \prod_{c \in C(\boldsymbol{y},\boldsymbol{x})} \Phi_c(\boldsymbol{y},\boldsymbol{x};\boldsymbol{\lambda})$$

where $\Phi_c(\boldsymbol{y},\boldsymbol{x};\boldsymbol{\lambda})$ is a non-negative real value potential function on a clique $c \in C(\boldsymbol{y},\boldsymbol{x})$. $Z_{\boldsymbol{\lambda}}(\boldsymbol{x}) = \sum_{\tilde{\boldsymbol{y}} \in \mathcal{Y}} \prod_{c \in C(\tilde{\boldsymbol{y}},\boldsymbol{x})} \Phi_c(\tilde{\boldsymbol{y}},\boldsymbol{x};\boldsymbol{\lambda})$ is a normalization factor over all output values, $\mathcal{Y}$.

Following the definitions of (Sha and Pereira, 2003), a log-linear combination of weighted features, $\Phi_c(\boldsymbol{y},\boldsymbol{x};\boldsymbol{\lambda}) = \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}_c(\boldsymbol{y},\boldsymbol{x}))$, is used as individual potential functions, where $\boldsymbol{f}_c$ represents a feature vector obtained from the corresponding clique c. That is, $\prod_{c \in C(\boldsymbol{y},\boldsymbol{x})} \Phi_c(\boldsymbol{y},\boldsymbol{x}) = \exp(\boldsymbol{\lambda} \cdot F(\boldsymbol{y},\boldsymbol{x}))$, where $F(\boldsymbol{y},\boldsymbol{x}) = \sum_c \boldsymbol{f}_c(\boldsymbol{y},\boldsymbol{x})$ is the CRF's global feature vector for $\boldsymbol{x}$ and $\boldsymbol{y}$.

The most probable output $\hat{\boldsymbol{y}}$ is given by $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y} \in \mathcal{Y}} p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda})$. However $Z_{\boldsymbol{\lambda}}(\boldsymbol{x})$ never affects the decision of $\hat{\boldsymbol{y}}$ since $Z_{\boldsymbol{\lambda}}(\boldsymbol{x})$ does not depend on $\boldsymbol{y}$. Thus, we can obtain the following discriminant function for CRFs:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y} \in \mathcal{Y}} \boldsymbol{\lambda} \cdot F(\boldsymbol{y},\boldsymbol{x}). \tag{1}$$

The maximum (log-)likelihood (ML) of the conditional probability $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda})$ of training data $\{(\boldsymbol{x}^k, \boldsymbol{y}^{*k})\}_{k=1}^N$ w.r.t. parameters $\boldsymbol{\lambda}$ is the most basic CRF training criterion, that is, $\arg\max_{\boldsymbol{\lambda}} \sum_k \log p(\boldsymbol{y}^{*k}|\boldsymbol{x}^k;\boldsymbol{\lambda})$, where $\boldsymbol{y}^{*k}$ is the correct output for the given $\boldsymbol{x}^k$. Maximizing the conditional log-likelihood given by CRFs is equivalent to minimizing the log-loss function, $\sum_k -\log p(\boldsymbol{y}^{*k}|\boldsymbol{x}^k;\boldsymbol{\lambda})$. We minimize the following loss function for the ML criterion training of CRFs:

$$\mathcal{L}^{\text{ML}}_{\boldsymbol{\lambda}} = \sum_k \Big[-\boldsymbol{\lambda} \cdot F(\boldsymbol{y}^{*k},\boldsymbol{x}^k) + \log Z_{\boldsymbol{\lambda}}(\boldsymbol{x}^k) \Big].$$

To reduce over-fitting, the Maximum a Posteriori (MAP) criterion of parameters $\boldsymbol{\lambda}$, that is, $\arg\max_{\boldsymbol{\lambda}} \sum_k \log p(\boldsymbol{\lambda}|\boldsymbol{y}^{*k},\boldsymbol{x}^k) \propto \sum_k \log p(\boldsymbol{y}^{*k}|\boldsymbol{x}^k;\boldsymbol{\lambda}) p(\boldsymbol{\lambda})$, is now the most widely used CRF training criterion. Therefore, we minimize the following loss function for the MAP criterion training of CRFs:

$$\mathcal{L}^{\text{MAP}}_{\boldsymbol{\lambda}} = \mathcal{L}^{\text{ML}}_{\boldsymbol{\lambda}} - \log p(\boldsymbol{\lambda}). \tag{2}$$

There are several possible choices when selecting a prior distribution $p(\boldsymbol{\lambda})$. This paper only considers L_ϕ-norm prior, $p(\boldsymbol{\lambda}) \propto \exp(-||\boldsymbol{\lambda}||^\phi/\phi C)$, which becomes a Gaussian prior when ϕ=2. The essential difference between ML and MAP is simply that MAP has this prior term in the objective function. This paper sometimes refers to the ML and MAP criterion training of CRFs as ML/MAP.

In order to estimate the parameters $\boldsymbol{\lambda}$, we seek a zero of the gradient over the parameters $\boldsymbol{\lambda}$:

$$\begin{aligned} \nabla \mathcal{L}^{\text{MAP}}_{\boldsymbol{\lambda}} = -\nabla \log p(\boldsymbol{\lambda}) + \sum_k \Bigg[-F(\boldsymbol{y}^{*k},\boldsymbol{x}^k) \\ + \sum_{\boldsymbol{y} \in \mathcal{Y}^k} \frac{\exp(\boldsymbol{\lambda} \cdot F(\boldsymbol{y},\boldsymbol{x}^k))}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x}^k)} \cdot F(\boldsymbol{y},\boldsymbol{x}^k) \Bigg]. \end{aligned} \tag{3}$$

The gradient of ML is Eq. 3 without the gradient term of the prior, $-\nabla \log p(\boldsymbol{\lambda})$.

The details of actual optimization procedures for linear chain CRFs, which are typical CRF applications, have already been reported (Sha and Pereira, 2003).

3 MCE Criterion Training for CRFs

The Minimum Classification Error (MCE) framework first arose out of a broader family of approaches to pattern classifier design known as *Generalized Probabilistic Descent (GPD)* (Katagiri et al., 1991). The MCE criterion minimizes an empirical loss corresponding to a smooth approximation of the classification error. This MCE loss is itself defined in terms of a *misclassification measure* derived from the *discriminant functions* of a given task. Via the smoothing parameters, the MCE loss function can be made arbitrarily close to the binary classification error. An important property of this framework is that it makes it possible in principle to achieve the optimal Bayes error even under *incorrect* modeling assumptions. It is easy to extend the MCE framework to use evaluation measures other than the classification error, namely the linear combination of error rates. Thus, it is possible to optimize directly a variety of (smoothed) evaluation measures. This is the approach proposed in this article.

We first introduce a framework for MCE criterion training, focusing only on error rate optimization. Sec. 4 then describes an example of minimizing a different multivariate evaluation measure using MCE criterion training.

3.1 Brief Overview of MCE

Let $\boldsymbol{x} \in \mathcal{X}$ be an input, and $\boldsymbol{y} \in \mathcal{Y}$ be an output. The *Bayes decision rule* decides the most probable output $\hat{\boldsymbol{y}}$ for $\boldsymbol{x}$, by using the maximum a posteriori probability, $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}\in\mathcal{Y}} p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda})$. In general, $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda})$ can be replaced by a more general *discriminant function*, that is,

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}\in\mathcal{Y}} g(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\lambda}). \quad (4)$$

Using the discriminant functions for the possible output of the task, the *misclassification measure* $d()$ is defined as follows:

$$d(\boldsymbol{y}^*,\boldsymbol{x},\boldsymbol{\lambda}) = -g(\boldsymbol{y}^*,\boldsymbol{x},\boldsymbol{\lambda}) + \max_{\boldsymbol{y}\in\mathcal{Y}\setminus\boldsymbol{y}^*} g(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\lambda}). \quad (5)$$

where $\boldsymbol{y}^*$ is the correct output for $\boldsymbol{x}$. Here it can be noted that, for a given $\boldsymbol{x}$, $d() \geq 0$ indicates misclassification. By using $d()$, the minimization of the error rate can be rewritten as the minimization of the sum of 0-1 (step) losses of the given training data. That is, $\arg\min_{\boldsymbol{\lambda}} \mathcal{L}_{\boldsymbol{\lambda}}$ where

$$\mathcal{L}_{\boldsymbol{\lambda}} = \sum_k \delta(d(\boldsymbol{y}^{*k},\boldsymbol{x}^k,\boldsymbol{\lambda})). \quad (6)$$

$\delta(r)$ is a step function returning 0 if $r<0$ and 1 otherwise. That is, δ is 0 if the value of the discriminant function of the correct output $g(\boldsymbol{y}^{*k},\boldsymbol{x}^k,\boldsymbol{\lambda})$ is greater than that of the maximum *incorrect* output $g(\boldsymbol{y}^k,\boldsymbol{x}^k,\boldsymbol{\lambda})$, and δ is 1 otherwise.

Eq. 5 is not an appropriate function for optimization since it is a discontinuous function w.r.t. the parameters $\boldsymbol{\lambda}$. One choice of continuous misclassification measure consists of substituting 'max' with 'soft-max', $\max_k r_k \approx \log \sum_k \exp(r_k)$. As a result

$$d(\boldsymbol{y}^*,\boldsymbol{x},\boldsymbol{\lambda}) = -g^* + \log\left[\mathcal{A}\sum_{\boldsymbol{y}\in\mathcal{Y}\setminus\boldsymbol{y}^*}\exp(\psi g)\right]^{\frac{1}{\psi}}, \quad (7)$$

where $g^* = g(\boldsymbol{y}^*,\boldsymbol{x},\boldsymbol{\lambda})$, $g = g(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\lambda})$, and $\mathcal{A} = \frac{1}{|\mathcal{Y}|-1}$. ψ is a positive constant that represents L_ψ-norm. When ψ approaches ∞, Eq. 7 converges to Eq. 5. Note that we can design any misclassification measure, including non-linear measures for $d()$. Some examples are shown in the Appendices.

Of even greater concern is the fact that the step function δ is discontinuous; minimization of Eq. 6 is therefore NP-complete. In the MCE formalism, $\delta()$ is replaced with an approximated 0-1 loss function, $l()$, which we refer to as a *smoothing function*. A typical choice for $l()$ is the *sigmoid function*, $l^{\text{sig}}()$, which is differentiable and provides a good approximation of the 0-1 loss when the hyper-parameter α is large (see Eq. 8). Another choice is the *(regularized) logistic function*, $l^{\text{log}}()$, that gives the upper bound of the 0-1 loss. Logistic loss is used as a conventional CRF loss function and provides convexity while the sigmoid function does not. These two smoothing functions can be written as follows:

$$\begin{aligned} l^{\text{sig}} &= (1+\exp(-\alpha\cdot d(\boldsymbol{y}^*,\boldsymbol{x},\boldsymbol{\lambda})-\beta))^{-1} \\ l^{\text{log}} &= \alpha^{-1}\cdot\log(1+\exp(\alpha\cdot d(\boldsymbol{y}^*,\boldsymbol{x},\boldsymbol{\lambda})+\beta)), \end{aligned} \quad (8)$$

where α and β are the hyper-parameters of the training.

We can introduce a regularization term to reduce over-fitting, which is derived using the same sense as in MAP, Eq. 2. Finally, the objective function of the MCE criterion with the regularization term can be rewritten in the following form:

$$\mathcal{L}^{\text{MCE}}_{\boldsymbol{\lambda}} = \mathcal{F}_{l,d,g,\boldsymbol{\lambda}}\left[\{(\boldsymbol{x}^k,\boldsymbol{y}^{*k})\}_{k=1}^N\right] + \frac{||\boldsymbol{\lambda}||^\phi}{\phi C}. \quad (9)$$

Then, the objective function of the MCE criterion that minimizes the error rate is Eq. 9 and

$$\mathcal{F}^{\text{MCE}}_{l,d,g,\boldsymbol{\lambda}} = \frac{1}{N}\sum_{k=1}^{N} l(d(\boldsymbol{y}^{*k},\boldsymbol{x}^k,\boldsymbol{\lambda})) \quad (10)$$

is substituted for $\mathcal{F}_{l,d,g,\boldsymbol{\lambda}}$. Since N is constant, we can eliminate the term $1/N$ in actual use.

3.2 Formalization

We simply substitute the discriminant function of the CRFs into that of the MCE criterion:

$$g(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\lambda}) = \log p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\lambda}) \propto \boldsymbol{\lambda}\cdot F(\boldsymbol{y},\boldsymbol{x}) \quad (11)$$

Basically, CRF training with the MCE criterion optimizes Eq. 9 with Eq. 11 after the selection of an appropriate misclassification measure, $d()$, and smoothing function, $l()$. Although there is no restriction on the choice of $d()$ and $l()$, in this work we select sigmoid or logistic functions for $l()$ and Eq. 7 for $d()$.

The gradient of the loss function Eq. 9 can be decomposed by the following chain rule:

$$\nabla \mathcal{L}^{\text{MCE}}_{\boldsymbol{\lambda}} = \frac{\partial \mathcal{F}()}{\partial l()} \cdot \frac{\partial l()}{\partial d()} \cdot \frac{\partial d()}{\partial \boldsymbol{\lambda}} + \frac{||\lambda||^{\phi-1}}{C}.$$

The derivatives of $l()$ w.r.t. $d()$ given in Eq. 8 are written as: $\partial l^{\text{sig}} / \partial d = \alpha \cdot l^{\text{sig}} \cdot (1 - l^{\text{sig}})$ and $\partial l^{\text{log}} / \partial d = l^{\text{sig}}$.

The derivative of $d()$ of Eq. 7 w.r.t. parameters $\boldsymbol{\lambda}$ is written in this form:

$$\begin{aligned} \frac{\partial d()}{\partial \boldsymbol{\lambda}} = & -\frac{Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \psi)}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \psi) - \exp(\psi g^*)} \cdot F(\boldsymbol{y}^*, \boldsymbol{x}) \\ & + \sum_{\boldsymbol{y} \in \mathcal{Y}} \left[\frac{\exp(\psi g)}{Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \psi) - \exp(\psi g^*)} \cdot F(\boldsymbol{y}, \boldsymbol{x}) \right] \end{aligned} \quad (12)$$

where $g = \boldsymbol{\lambda} \cdot F(\boldsymbol{y}, \boldsymbol{x})$, $g^* = \boldsymbol{\lambda} \cdot F(\boldsymbol{y}^*, \boldsymbol{x})$, and $Z_{\boldsymbol{\lambda}}(\boldsymbol{x}, \psi) = \sum_{\boldsymbol{y} \in \mathcal{Y}} \exp(\psi g)$.

Note that we can obtain exactly the same loss function as ML/MAP with appropriate choices of $\mathcal{F}()$, $l()$ and $d()$. The details are provided in the Appendices. Therefore, ML/MAP can be seen as one special case of the framework proposed here. In other words, our method provides a generalized framework of CRF training.

3.3 Optimization Procedure

With linear chain CRFs, we can calculate the objective function, Eq. 9 combined with Eq. 10, and the gradient, Eq. 12, by using the variant of the forward-backward and Viterbi algorithm described in (Sha and Pereira, 2003). Moreover, for the parameter optimization process, we can simply exploit gradient descent or quasi-Newton methods such as L-BFGS (Liu and Nocedal, 1989) as well as ML/MAP optimization.

If we select $\psi = \infty$ for Eq. 7, we only need to evaluate the correct and the maximum incorrect output. As we know, the maximum output can be efficiently calculated with the Viterbi algorithm, which is the same as calculating Eq. 1. Therefore, we can find the maximum incorrect output by using the A* algorithm (Hart et al., 1968), if the maximum output is the correct output, and by using the Viterbi algorithm otherwise. It may be feared that since the objective function is not differentiable everywhere for $\psi = \infty$, problems for optimization would occur. However, it has been shown (Le Roux and McDermott, 2005) that even simple gradient-based (first-order) optimization methods such as GPD and (approximated) second-order methods such as *QuickProp* (Fahlman, 1988) and BFGS-based methods have yielded good experimental optimization results.

4 Multivariate Evaluation Measures

Thus far, we have discussed the error rate version of MCE. Unlike ML/MAP, the framework of MCE criterion training allows the embedding of not only a linear combination of error rates, but also any evaluation measure, including non-linear measures.

Several non-linear objective functions, such as F-score for text classification (Gao et al., 2003), and BLEU-score and some other evaluation measures for statistical machine translation (Och, 2003), have been introduced with reference to the framework of MCE criterion training.

4.1 Sequential Segmentation Tasks (SSTs)

Hereafter, we focus solely on CRFs in sequences, namely the linear chain CRF. We assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ have the same length: $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$. In a linear chain CRF, y_i depends only on y_{i-1}.

Sequential segmentation tasks (SSTs), such as text chunking (Chunking) and named entity recognition (NER), which constitute the shared tasks of the *Conference of Natural Language Learning (CoNLL)* 2000, 2002 and 2003, are typical CRF applications. These tasks require the extraction of pre-defined segments, referred to as *target segments*, from given texts. Fig. 1 shows typical examples of SSTs. These tasks are generally treated as *sequential labeling problems* incorporating the IOB tagging scheme (Ramshaw and Marcus, 1995). The IOB tagging scheme, where we only consider the *IOB2* scheme, is also shown in Fig. 1. B-X, I-X and O indicate that the word in question is the beginning of the tag 'X', inside the tag 'X', and outside any target segment, respectively. Therefore, a segment is defined as a sequence of a few outputs.

4.2 Segmentation F-score Loss for SSTs

The standard evaluation measure of SSTs is the *segmentation F-score* (Sang and Buchholz, 2000):

$$\text{F}_\gamma = \frac{(\gamma^2 + 1) \cdot TP}{\gamma^2 \cdot FN + FP + (\gamma^2 + 1) \cdot TP} \quad (13)$$

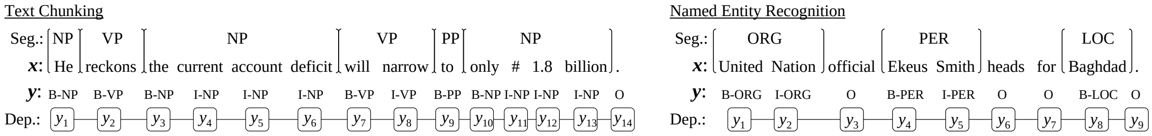

Text Chunking
Seg.: NP VP NP VP PP NP
x: He reckons the current account deficit will narrow to only # 1.8 billion.
y: B-NP B-VP B-NP I-NP I-NP I-NP B-VP I-VP B-PP B-NP I-NP I-NP I-NP O
Dep.: y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8 y_9 y_{10} y_{11} y_{12} y_{13} y_{14}
Named Entity Recognition
Seg.: ORG PER LOC
x: United Nation official Ekeus Smith heads for Baghdad.
y: B-ORG I-ORG O B-PER I-PER O O B-LOC O
Dep.: y_1 y_2 y_3 y_4 y_5 y_6 y_7 y_8 y_9

Figure 1: Examples of sequential segmentation tasks (SSTs): text chunking (Chunking) and named entity recognition (NER).

where TP, FP and FN represent true positive, false positive and false negative counts, respectively.

The individual evaluation units used to calculate TP, FN and PN, are not individual outputs y_i or output sequences $\boldsymbol{y}$, but rather segments. We need to define a *segment-wise loss*, in contrast to the standard CRF loss, which is sometimes referred to as an *(entire) sequential loss* (Kakade et al., 2002; Altun et al., 2003). First, we consider the point-wise decision w.r.t. Eq. 1, that is, $\hat{y}_i = \arg\max_{y_i \in \mathcal{Y}_1} g(\boldsymbol{y}, \boldsymbol{x}, i, \boldsymbol{\lambda})$. The point-wise discriminant function can be written as follows:

$$g(\boldsymbol{y}, \boldsymbol{x}, i, \boldsymbol{\lambda}) = \max_{\boldsymbol{y}' \in \mathcal{Y}_{|\boldsymbol{y}|}[y_i]} \boldsymbol{\lambda} \cdot F(\boldsymbol{y}', \boldsymbol{x}) \qquad (14)$$

where $\mathcal{Y}_j$ represents a set of all $\boldsymbol{y}$ whose length is j, and $\mathcal{Y}[y_i]$ represents a set of all $\boldsymbol{y}$ that contain y_i in the i'th position. Note that the same output $\hat{\boldsymbol{y}}$ can be obtained with Eqs. 1 and 14, that is, $\hat{\boldsymbol{y}} = (\hat{y}_1, \ldots, \hat{y}_n)$. This point-wise discriminant function is different from that described in (Kakade et al., 2002; Altun et al., 2003), which is calculated based on marginals.

Let $\boldsymbol{y}_{s_j}$ be an output sequence corresponding to the j-th segment of $\boldsymbol{y}$, where s_j represents a sequence of indices of $\boldsymbol{y}$, that is, $s_j = (s_{j,1}, \ldots, s_{j,|s_j|})$. An example of the Chunking data shown in Fig. 1, $\boldsymbol{y}_{s_4}$ is (B-VP, I-VP) where $s_4 = (7, 8)$. Let $\mathcal{Y}[\boldsymbol{y}_{s_j}]$ be a set of all outputs whose positions from $s_{j,1}$ to $s_{j,|s_j|}$ are $\boldsymbol{y}_{s_j} = (y_{s_{j,1}}, \ldots, y_{s_{j,|s_j|}})$. Then, we can define a segment-wise discriminant function w.r.t. Eq. 1. That is,

$$g(\boldsymbol{y}, \boldsymbol{x}, s_j, \boldsymbol{\lambda}) = \max_{\boldsymbol{y}' \in \mathcal{Y}_{|\boldsymbol{y}|}[\boldsymbol{y}_{s_j}]} \boldsymbol{\lambda} \cdot F(\boldsymbol{y}', \boldsymbol{x}). \qquad (15)$$

Note again that the same output $\hat{\boldsymbol{y}}$ can be obtained using Eqs. 1 and 15, as with the piece-wise discriminant function described above. This property is needed for evaluating segments since we do not know the correct segments of the test data; we can maintain consistency even if we use Eq. 1 for testing and Eq. 15 for training. Moreover, Eq. 15 obviously reduces to Eq. 14 if the length of all segments is 1. Then, the segment-wise misclassification measure $d(\boldsymbol{y}^*, \boldsymbol{x}, s_j, \boldsymbol{\lambda})$ can be obtained simply by replacing the discriminant function of the entire sequence $g(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\lambda})$ with that of segment-wise $g(\boldsymbol{y}, \boldsymbol{x}, s_j, \boldsymbol{\lambda})$ in Eq. 7.

Let $\boldsymbol{s}^{*k}$ be a segment sequence corresponding to the correct output $\boldsymbol{y}^{*k}$ for a given $\boldsymbol{x}^k$, and $\mathcal{S}(\boldsymbol{x}^k)$ be all possible segments for a given $\boldsymbol{x}^k$. Then, approximated evaluation functions of TP, FP and FN can be defined as follows:

$$\begin{aligned}
TP_l &= \sum_k \sum_{s_j^* \in \boldsymbol{s}^{*k}} \Big[1 - l(d(\boldsymbol{y}^{*k}, \boldsymbol{x}^k, s_j^*, \boldsymbol{\lambda}))\Big] \cdot \delta(s_j^*) \\
FP_l &= \sum_k \sum_{s'_j \in \mathcal{S}(\boldsymbol{x}^k) \setminus \boldsymbol{s}^{*k}} l(d(\boldsymbol{y}^{*k}, \boldsymbol{x}^k, s'_j, \boldsymbol{\lambda})) \cdot \delta(s'_j) \\
FN_l &= \sum_k \sum_{s_j^* \in \boldsymbol{s}^{*k}} l(d(\boldsymbol{y}^{*k}, \boldsymbol{x}^k, s_j^*, \boldsymbol{\lambda})) \cdot \delta(s_j^*)
\end{aligned}$$

where $\delta(s_j)$ returns 1 if segment s_j is a target segment, and returns 0 otherwise. For the NER data shown in Fig. 1, 'ORG', 'PER' and 'LOC' are the target segments, while segments that are labeled 'O' in $\mathbf{y}$ are not. Since TP_l should not have a value of less than zero, we select sigmoid loss as the smoothing function $l()$.

The second summation of TP_l and FN_l performs a summation over correct segments $\boldsymbol{s}^*$. In contrast, the second summation in FP_l takes all possible segments into account, but excludes the correct segments $\boldsymbol{s}^*$. Although an efficient way to evaluate all possible segments has been proposed in the context of semi-Markov CRFs (Sarawagi and Cohen, 2004), we introduce a simple alternative method. If we select $\psi = \infty$ for $d()$ in Eq. 7, we only need to evaluate the segments corresponding to the maximum incorrect output $\tilde{\boldsymbol{y}}$ to calculate FP_l. That is, $s'_j \in \mathcal{S}(\boldsymbol{x}^k) \setminus \boldsymbol{s}^{*k}$ can be reduced to $s'_j \in \tilde{\boldsymbol{s}}^k$, where $\tilde{\boldsymbol{s}}^k$ represents segments corresponding to the maximum incorrect output $\tilde{\boldsymbol{y}}$. In practice, this reduces the calculation cost and so we used this method for our experiments described in the next section.

Maximizing the segmentation F$_\gamma$-score, Eq. 13, is equivalent to minimizing $\frac{\gamma^2 \cdot FN+FP}{(\gamma^2+1)\cdot TP}$, since Eq. 13 can also be written as F$_\gamma = \frac{1}{1+\frac{\gamma^2 \cdot FN+FP}{(\gamma^2+1)\cdot TP}}$. Thus, an objective function closely reflecting the segmentation F$_\gamma$-score based on the MCE criterion can be written as Eq. 9 while replacing $\mathcal{F}_{l,d,g,\boldsymbol{\lambda}}$ with:

$$\mathcal{F}^{\text{MCE-F}}_{l,d,g,\boldsymbol{\lambda}} = \frac{\gamma^2 \cdot FN_l + FP_l}{(\gamma^2+1)\cdot TP_l}. \tag{16}$$

The derivative of Eq. 16 w.r.t. $l()$ is given by the following equation:

$$\frac{\partial \mathcal{F}^{\text{MCE-F}}_{l,d,g,\boldsymbol{\lambda}}}{\partial l()} = \begin{cases} \frac{\gamma^2}{Z_D} + \frac{(\gamma^2+1)\cdot Z_N}{Z_D^2}, & \text{if } \delta(s_j^*) = 1 \\ \frac{1}{Z_D}, & \text{otherwise} \end{cases}$$

where Z_N and Z_D represent the numerator and denominator of Eq. 16, respectively.

In the optimization process of the segmentation F-score objective function, we can efficiently calculate Eq. 15 by using the forward and backward Viterbi algorithm, which is almost the same as calculating Eq. 3 with a variant of the forward-backward algorithm (Sha and Pereira, 2003). The same numerical optimization methods described in Sec. 3.3 can be employed for this optimization.

5 Experiments

We used the same Chunking and 'English' NER task data used for the shared tasks of CoNLL-2000 (Sang and Buchholz, 2000) and CoNLL-2003 (Sang and De Meulder, 2003), respectively.

Chunking data was obtained from the Wall Street Journal (WSJ) corpus: sections 15-18 as training data (8,936 sentences and 211,727 tokens), and section 20 as test data (2,012 sentences and 47,377 tokens), with 11 different chunk-tags, such as NP and VP plus the 'O' tag, which represents the outside of any target chunk (segment).

The English NER data was taken from the Reuters Corpus2[1]. The data consists of 203,621, 51,362 and 46,435 tokens from 14,987, 3,466 and 3,684 sentences in training, development and test data, respectively, with four named entity tags, PERSON, LOCATION, ORGANIZATION and MISC, plus the 'O' tag.

5.1 Comparison Methods and Parameters

For ML and MAP, we performed exactly the same training procedure described in (Sha and Pereira, 2003) with L-BFGS optimization. For MCE, we only considered $d()$ with $\psi = \infty$ as described in Sec. 4.2, and used QuickProp optimization[2].

For MAP, MCE and MCE-F, we used the L_2-norm regularization. We selected a value of C from 1.0×10^n where n takes a value from -5 to 5 in intervals 1 by development data[3]. The tuning of smoothing function hyper-parameters is not considered in this paper; that is, α=1 and β=0 were used for all the experiments.

We evaluated the performance by Eq. 13 with $\gamma = 1$, which is the evaluation measure used in CoNLL-2000 and 2003. Moreover, we evaluated the performance by using the average sentence accuracy, since the conventional ML/MAP objective function reflects this sequential accuracy.

5.2 Features

As regards the basic feature set for Chunking, we followed (Kudo and Matsumoto, 2001), which is the same feature set that provided the best result in CoNLL-2000. We expanded the basic features by using bigram combinations of the same types of features, such as words and part-of-speech tags, within window size 5.

In contrast to the above, we used the original feature set for NER. We used features derived only from the data provided by CoNLL-2003 with the addition of character-level regular expressions of uppercases [A-Z], lowercases [a-z], digits [0-9] or others, and prefixes and suffixes of one to four letters. We also expanded the above basic features by using bigram combinations within window size 5. Note that we never used features derived from external information such as the Web, or a dictionary, which have been used in many previous studies but which are difficult to employ for validating the experiments.

5.3 Results and Discussion

Our experiments were designed to investigate the impact of eliminating the inconsistency between objective functions and evaluation measures, that is, to compare ML/MAP and MCE-F.

Table 1 shows the results of Chunking and NER. The F$_{\gamma=1}$ and 'Sent' columns show the performance evaluated using segmentation F-score and

[1] http://trec.nist.gov/data/reuters/reuters.html

[2] In order to realize faster convergence, we applied online GPD optimization for the first ten iterations.

[3] Chunking has no common development set. We first train the systems with all but the last 2000 sentences in the training data as a development set to obtain C, and then retrain them with all the training data.

Table 1: Performance of text chunking and named entity recognition data (CoNLL-2000 and 2003)

		Chunking			NER		
	$l()$	n	$\mathrm{F}_{\gamma=1}$	Sent	n	$\mathrm{F}_{\gamma=1}$	Sent
MCE-F	(sig)	5	**93.96**	**60.44**	4	**84.72**	**78.72**
MCE	(log)	3	93.92	60.19	3	84.30	78.02
MCE	(sig)	3	93.85	60.14	3	83.82	77.52
MAP		0	93.71	59.15	0	83.79	77.39
ML		-	93.19	56.26	-	82.39	75.71

sentence accuracy, respectively. MCE-F refers to the results obtained from optimizing Eq. 9 based on Eq. 16. In addition, we evaluated the error rate version of MCE. MCE(log) and MCE(sig) indicate that logistic and sigmoid functions are selected for $l()$, respectively, when optimizing Eq. 9 based on Eq. 10. Moreover, MCE(log) and MCE(sig) used $d()$ based on $\psi{=}\infty$, and were optimized using QuickProp; these are the same conditions as used for MCE-F. We found that MCE-F exhibited the best results for both Chunking and NER. There is a significant difference ($p < 0.01$) between MCE-F and ML/MAP with the McNemar test, in terms of the correctness of both individual outputs, y_i^k, and sentences, $\boldsymbol{y}^k$.

NER data has 83.3% (170524/204567) and 82.6% (38554/46666) of 'O' tags in the training and test data, respectively while the corresponding values of the Chunking data are only 13.1% (27902/211727) and 13.0% (6180/47377). In general, such an imbalanced data set is unsuitable for accuracy-based evaluation. This may be one reason why MCE-F improved the NER results much more than the Chunking results.

The only difference between MCE(sig) and MCE-F is the objective function. The corresponding results reveal the effectiveness of using an objective function that is consistent as the evaluation measure for the target task. These results show that minimizing the error rate is not optimal for improving the segmentation F-score evaluation measure. Eliminating the inconsistency between the task evaluation measure and the objective function during the training can improve the overall performance.

5.3.1 Influence of Initial Parameters

While ML/MAP and MCE(log) is convex w.r.t. the parameters, neither the objective function of MCE-F, nor that of MCE(sig), is convex. Therefore, initial parameters can affect the optimization results, since QuickProp as well as L-BFGS can only find local optima.

Table 2: Performance when initial parameters are derived from MAP

		Chunking			NER		
	$l()$	n	$\mathrm{F}_{\gamma=1}$	Sent	n	$\mathrm{F}_{\gamma=1}$	Sent
MCE-F	(sig)	5	**94.03**	**60.74**	4	**85.29**	**79.26**
MCE	(sig)	3	93.97	60.59	3	84.57	77.71

The previous experiments were only performed with all parameters initialized at zero. In this experiment, the parameters obtained by the MAP-trained model were used as the initial values of MCE-F and MCE(sig). This evaluation setting appears to be similar to *reranking*, although we used exactly the same model and feature set.

Table 2 shows the results of Chunking and NER obtained with this parameter initialization setting. When we compare Tables 1 and 2, we find that the initialization with the MAP parameter values further improves performance.

6 Related Work

Various loss functions have been proposed for designing CRFs (Kakade et al., 2002; Altun et al., 2003). This work also takes the design of the loss functions for CRFs into consideration. However, we proposed a general framework for designing these loss function that included non-linear loss functions, which has not been considered in previous work.

With Chunking, (Kudo and Matsumoto, 2001) reported the best F-score of 93.91 with the voting of several models trained by *Support Vector Machine* in the same experimental settings and with the same feature set. MCE-F with the MAP parameter initialization achieved an F-score of 94.03, which surpasses the above result without manual parameter tuning.

With NER, we cannot make a direct comparison with previous work in the same experimental settings because of the different feature set, as described in Sec. 5.2. However, MCE-F showed the better performance of 85.29 compared with (McCallum and Li, 2003) of 84.04, which used the MAP training of CRFs with a feature selection architecture, yielding similar results to the MAP results described here.

7 Conclusions

We proposed a framework for training CRFs based on optimization criteria directly related to target multivariate evaluation measures. We first provided a general framework of CRF training based on MCE criterion. Then, specifically focusing on SSTs, we introduced an approximate segmentation F-score objective function. Experimental results showed that eliminating the inconsistency between the task evaluation measure and the objective function used during training improves the overall performance in the target task without any change in feature set or model.

Appendices

Misclassification measures

Another type of misclassification measure using soft-max is (Katagiri et al., 1991):

$$d(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\lambda}) = -g^* + \left[\mathcal{A}\sum_{y\in\mathcal{Y}\setminus y^*} g^{\psi}\right]^{\frac{1}{\psi}}.$$

Another $d()$, for g in the range $[0,\infty)$:

$$d(\boldsymbol{y},\boldsymbol{x},\boldsymbol{\lambda}) = \left[\mathcal{A}\textstyle\sum_{y\in\mathcal{Y}\setminus y^*} g^{\psi}\right]^{\frac{1}{\psi}}/g^*.$$

Comparison of ML/MAP and MCE

If we select $l^{\log}()$ with $\alpha = 1$ and $\beta = 0$, and use Eq. 7 with $\psi = 1$ and without the term $\mathcal{A}$ for $d()$. We can obtain the same loss function as ML/MAP:

$$\begin{aligned}
&\log\left(1+\exp(-g^* + \log(Z_{\boldsymbol{\lambda}} - \exp(g^*)))\right) \\
&= \log\left(\frac{\exp(g^*) + (Z_{\boldsymbol{\lambda}} - \exp(g^*))}{\exp(g^*)}\right) \\
&= -g^* + \log(Z_{\boldsymbol{\lambda}}).
\end{aligned}$$

References

Y. Altun, M. Johnson, and T. Hofmann. 2003. Investigating Loss Functions and Optimization Methods for Discriminative Learning of Label Sequences. In *Proc. of EMNLP-2003*, pages 145–152.

S. E. Fahlman. 1988. An Empirical Study of Learning Speech in Backpropagation Networks. In *Technical Report CMU-CS-88-162, Carnegie Mellon University*.

S. Gao, W. Wu, C.-H. Lee, and T.-S. Chua. 2003. A Maximal Figure-of-Merit Approach to Text Categorization. In *Proc. of SIGIR'03*, pages 174–181.

P. E. Hart, N. J. Nilsson, and B. Raphael. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. on Systems Science and Cybernetics*, SSC-4(2):100–107.

M. Jansche. 2005. Maximum Expected F-Measure Training of Logistic Regression Models. In *Proc. of HLT/EMNLP-2005*, pages 692–699.

T. Joachims. 2005. A Support Vector Method for Multivariate Performance Measures. In *Proc. of ICML-2005*, pages 377–384.

B. H. Juang and S. Katagiri. 1992. Discriminative Learning for Minimum Error Classification. *IEEE Trans. on Signal Processing*, 40(12):3043–3053.

S. Kakade, Y. W. Teh, and S. Roweis. 2002. An Alternative Objective Function for Markovian Fields. In *Proc. of ICML-2002*, pages 275–282.

S. Katagiri, C. H. Lee, and B.-H. Juang. 1991. New Discriminative Training Algorithms based on the Generalized Descent Method. In *Proc. of IEEE Workshop on Neural Networks for Signal Processing*, pages 299–308.

T. Kudo and Y. Matsumoto. 2001. Chunking with Support Vector Machines. In *Proc. of NAACL-2001*, pages 192–199.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proc. of ICML-2001*, pages 282–289.

D. C. Liu and J. Nocedal. 1989. On the Limited Memory BFGS Method for Large-scale Optimization. *Mathematic Programming*, (45):503–528.

A. McCallum and W. Li. 2003. Early Results for Named Entity Recognition with Conditional Random Fields Feature Induction and Web-Enhanced Lexicons. In *Proc. of CoNLL-2003*, pages 188–191.

F. J. Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of ACL-2003*, pages 160–167.

Y. Qi, M. Szummer, and T. P. Minka. 2005. Bayesian Conditional Random Fields. In *Proc. of AI & Statistics 2005*.

L. A. Ramshaw and M. P. Marcus. 1995. Text Chunking using Transformation-based Learning. In *Proc. of VLC-1995*, pages 88–94.

J. Le Roux and E. McDermott. 2005. Optimization Methods for Discriminative Training. In *Proc. of Eurospeech 2005*, pages 3341–3344.

E. F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 Shared Task: Chunking. In *Proc. of CoNLL/LLL-2000*, pages 127–132.

E. F. Tjong Kim Sang and F. De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proc. of CoNLL-2003*, pages 142–147.

S. Sarawagi and W. W. Cohen. 2004. Semi-Markov Conditional Random Fields for Information Extraction. In *Proc of NIPS-2004*.

F. Sha and F. Pereira. 2003. Shallow Parsing with Conditional Random Fields. In *Proc. of HLT/NAACL-2003*, pages 213–220.

B. Taskar, C. Guestrin, and D. Koller. 2004. Max-Margin Markov Networks. In *Proc. of NIPS-2004*.

I. Tsochantaridis, T. Joachims and T. Hofmann, and Y. Altun. 2005. Large Margin Methods for Structured and Interdependent Output Variables. *JMLR*, 6:1453–1484.

Approximation Lasso Methods for Language Modeling

Jianfeng Gao
Microsoft Research
One Microsoft Way
Redmond WA 98052 USA
jfgao@microsoft.com

Hisami Suzuki
Microsoft Research
One Microsoft Way
Redmond WA 98052 USA
hisamis@microsoft.com

Bin Yu
Department of Statistics
University of California
Berkeley., CA 94720 U.S.A.
binyu@stat.berkeley.edu

Abstract

Lasso is a regularization method for parameter estimation in linear models. It optimizes the model parameters with respect to a loss function subject to model complexities. This paper explores the use of lasso for statistical language modeling for text input. Owing to the very large number of parameters, directly optimizing the penalized lasso loss function is impossible. Therefore, we investigate two approximation methods, the boosted lasso (BLasso) and the forward stagewise linear regression (FSLR). Both methods, when used with the exponential loss function, bear strong resemblance to the boosting algorithm which has been used as a discriminative training method for language modeling. Evaluations on the task of Japanese text input show that BLasso is able to produce the best approximation to the lasso solution, and leads to a significant improvement, in terms of character error rate, over boosting and the traditional maximum likelihood estimation.

1 Introduction

Language modeling (LM) is fundamental to a wide range of applications. Recently, it has been shown that a linear model estimated using discriminative training methods, such as the boosting and perceptron algorithms, outperforms significantly a traditional word trigram model trained using maximum likelihood estimation (MLE) on several tasks such as speech recognition and Asian language text input (Bacchiani et al. 2004; Roark et al. 2004; Gao et al. 2005; Suzuki and Gao 2005).

The success of discriminative training methods is largely due to fact that unlike the traditional approach (e.g., MLE) that maximizes the function (e.g., likelihood of training data) that is loosely associated with error rate, discriminative training methods aim to directly minimize the error rate on training data even if they reduce the likelihood. However, given a finite set of training samples, discriminative training methods could lead to an arbitrary complex model for the purpose of achieving zero training error. It is well-known that complex models exhibit high variance and perform poorly on unseen data. Therefore some regularization methods have to be used to control the complexity of the model.

Lasso is a regularization method for parameter estimation in linear models. It optimizes the model parameters with respect to a loss function subject to model complexities. The basic idea of lasso is originally proposed by Tibshirani (1996). Recently, there have been several implementations and experiments of lasso on multi-class classification tasks where only a small number of features need to be handled and the lasso solution can be directly computed via numerical methods. To our knowledge, this paper presents the first empirical study of lasso for a realistic, large scale task: LM for Asian language text input. Because the task utilizes millions of features and training samples, directly optimizing the penalized lasso loss function is impossible. Therefore, two approximation methods, the boosted lasso (BLasso, Zhao and Yu 2004) and the forward stagewise linear regression (FSLR, Hastie et al. 2001), are investigated. Both methods, when used with the exponential loss function, bear strong resemblance to the boosting algorithm which has been used as a discriminative training method for LM. Evaluations on the task of Japanese text input show that BLasso is able to produce the best approximation to the lasso solution, and leads to a significant improvement, in terms of character error rate, over the boosting algorithm and the traditional MLE.

2 LM Task and Problem Definition

This paper studies LM on the application of Asian language (e.g. Chinese or Japanese) text input, a standard method of inputting Chinese or Japanese text by converting the input phonetic symbols into the appropriate word string. In this paper we call the task IME, which stands for

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 225–232,
Sydney, July 2006. ©2006 Association for Computational Linguistics

input method editor, based on the name of the commonly used Windows-based application.

Performance on IME is measured in terms of the character error rate (CER), which is the number of characters wrongly converted from the phonetic string divided by the number of characters in the correct transcript.

Similar to speech recognition, IME is viewed as a Bayes decision problem. Let A be the input phonetic string. An IME system's task is to choose the most likely word string W^* among those candidates that could be converted from A:

$$W^* = \arg\max_{W \in \mathbf{GEN}(A)} P(W \mid A) = \arg\max_{W \in \mathbf{GEN}(A)} P(W)P(A \mid W) \quad (1)$$

where $\mathbf{GEN}(A)$ denotes the candidate set given A. Unlike speech recognition, however, there is no acoustic ambiguity as the phonetic string is inputted by users. Moreover, we can assume a unique mapping from W and A in IME as words have unique readings, i.e. $P(A|W) = 1$. So the decision of Equation (1) depends solely upon $P(W)$, making IME an ideal evaluation test bed for LM.

In this study, the LM task for IME is formulated under the framework of linear models (e.g., Duda et al. 2001). We use the following notation, adapted from Collins and Koo (2005):

- Training data is a set of example input/output pairs. In LM for IME, training samples are represented as $\{A_i, W_i^R\}$, for $i = 1 \ldots M$, where each A_i is an input phonetic string and W_i^R is the reference transcript of A_i.
- We assume some way of generating a set of candidate word strings given A, denoted by $\mathbf{GEN}(A)$. In our experiments, $\mathbf{GEN}(A)$ consists of top n word strings converted from A using a baseline IME system that uses only a word trigram model.
- We assume a set of D+1 features $f_d(W)$, for $d = 0 \ldots D$. The features could be arbitrary functions that map W to real values. Using vector notation, we have $\mathbf{f}(W) \in \Re^{D+1}$, where $\mathbf{f}(W) = [f_0(W), f_1(W), \ldots, f_D(W)]^{\mathrm{T}}$. $f_0(W)$ is called the base feature, and is defined in our case as the log probability that the word trigram model assigns to W. Other features ($f_d(W)$, for $d = 1 \ldots D$) are defined as the counts of word n-grams (n = 1 and 2 in our experiments) in W.
- Finally, the parameters of the model form a vector of D+1 dimensions, each for one feature function, $\boldsymbol{\lambda} = [\lambda_0, \lambda_1, \ldots, \lambda_D]$. The score of a word string W can be written as

$$Score(W, \boldsymbol{\lambda}) = \boldsymbol{\lambda}\mathbf{f}(W) = \sum_{d=0}^{D} \lambda_d f_d(W) \,. \quad (2)$$

The decision rule of Equation (1) is rewritten as

$$W^*(A, \boldsymbol{\lambda}) = \arg\max_{W \in \mathbf{GEN}(A)} Score(W, \boldsymbol{\lambda}) \,. \quad (3)$$

```
1  Set $\lambda_0$ = argmin$_{\lambda_0}$ExpLoss($\boldsymbol{\lambda}$); and $\lambda_d$ = 0 for $d$=1…$D$
2  Select a feature $f_{k^*}$ which has largest estimated
   impact on reducing ExpLoss of Eq. (6)
3  Update $\lambda_{k^*} \leftarrow \lambda_{k^*} + \delta^*$, and return to Step 2
```

Figure 1: The boosting algorithm

Equation (3) views IME as a ranking problem, where the model gives the ranking score, not probabilities. We therefore do not evaluate the model via perplexity.

Now, assume that we can measure the number of conversion errors in W by comparing it with a reference transcript W^R using an error function Er(W^R,W), which is the string edit distance function in our case. We call the sum of error counts over the training samples *sample risk*. Our goal then is to search for the best parameter set $\boldsymbol{\lambda}$ which minimizes the sample risk, as in Equation (4):

$$\boldsymbol{\lambda}_{MSR} \overset{def}{=} \arg\min_{\boldsymbol{\lambda}} \sum_{i=1 \ldots M} \mathrm{Er}(W_i^R, W_i^*(A_i, \boldsymbol{\lambda})) \,. \quad (4)$$

However, (4) cannot be optimized easily since Er(.) is a piecewise constant (or step) function of $\boldsymbol{\lambda}$ and its gradient is undefined. Therefore, discriminative methods apply different approaches that optimize it approximately. The boosting algorithm described below is one of such approaches.

3 Boosting

This section gives a brief review of the boosting algorithm, following the description of some recent work (e.g., Schapire and Singer 1999; Collins and Koo 2005).

The boosting algorithm uses an exponential loss function (ExpLoss) to approximate the sample risk in Equation (4). We define the margin of the pair (W^R, W) with respect to the model $\boldsymbol{\lambda}$ as

$$M(W^R, W) = Score(W^R, \boldsymbol{\lambda}) - Score(W, \boldsymbol{\lambda}) \quad (5)$$

Then, ExpLoss is defined as

$$\mathrm{ExpLoss}(\boldsymbol{\lambda}) = \sum_{i=1 \ldots M} \sum_{W_i \in \mathbf{GEN}(A_i)} \exp(-M(W_i^R, W_i)) \quad (6)$$

Notice that ExpLoss is convex so there is no problem with local minima when optimizing it. It is shown in Freund et al. (1998) and Collins and Koo (2005) that there exist gradient search procedures that converge to the right solution.

Figure 1 summarizes the boosting algorithm we used. After initialization, Steps 2 and 3 are repeated N times; at each iteration, a feature is chosen and its weight is updated as follows.

First, we define Upd($\boldsymbol{\lambda}$, k, δ) as an updated model, with the same parameter values as $\boldsymbol{\lambda}$ with the exception of λ_k, which is incremented by δ

$$\text{Upd}(\boldsymbol{\lambda}, k, \delta) = \{\lambda_0, \lambda_1, ..., \lambda_k + \delta, ..., \lambda_D\}$$

Then, Steps 2 and 3 in Figure 1 can be rewritten as Equations (7) and (8), respectively.

$$(k^*, \delta^*) = \arg\min_{k,\delta} \text{ExpLoss}(\text{Upd}(\boldsymbol{\lambda}, k, \delta)) \quad (7)$$

$$\boldsymbol{\lambda}^t = \text{Upd}(\boldsymbol{\lambda}^{t-1}, k^*, \delta^*) \quad (8)$$

The boosting algorithm can be too greedy: Each iteration usually reduces the ExpLoss(.) on training data, so for the number of iterations large enough this loss can be made arbitrarily small. However, fitting training data too well eventually leads to overfiting, which degrades the performance on unseen test data (even though in boosting overfitting can happen very slowly).

Shrinkage is a simple approach to dealing with the overfitting problem. It scales the incremental step δ by a small constant ν, $\nu \in (0, 1)$. Thus, the update of Equation (8) with shrinkage is

$$\boldsymbol{\lambda}^t = \text{Upd}(\boldsymbol{\lambda}^{t-1}, k^*, \nu\delta^*) \quad (9)$$

Empirically, it has been found that smaller values of ν lead to smaller numbers of test errors.

4 Lasso

Lasso is a regularization method for estimation in linear models (Tibshirani 1996). It regularizes or shrinks a fitted model through an L_1 penalty or constraint.

Let $T(\boldsymbol{\lambda})$ denote the L_1 penalty of the model, i.e., $T(\boldsymbol{\lambda}) = \sum_{d=0...D} |\lambda_d|$. We then optimize the model $\boldsymbol{\lambda}$ so as to minimize a regularized loss function on training data, called *lasso loss* defined as

$$\text{LassoLoss}(\boldsymbol{\lambda}, \alpha) = \text{ExpLoss}(\boldsymbol{\lambda}) + \alpha T(\boldsymbol{\lambda}) \quad (10)$$

where $T(\boldsymbol{\lambda})$ generally penalizes larger models (or complex models), and the parameter α controls the amount of regularization applied to the estimate. Setting $\alpha = 0$ reverses the LassoLoss to the unregularized ExpLoss; as α increases, the model coefficients all shrink, each ultimately becoming zero. In practice, α should be adaptively chosen to minimize an estimate of expected loss, e.g., α decreases with the increase of the number of iterations.

Computation of the solution to the lasso problem has been studied for special loss functions. For least square regression, there is a fast algorithm LARS to find the whole lasso path for different α' s (Obsborn et al. 2000a; 2000b; Efron et al. 2004); for 1-norm SVM, it can be transformed into a linear programming problem with a fast algorithm similar to LARS (Zhu et al. 2003). However, the solution to the lasso problem for a general convex loss function and an adaptive α remains open. More importantly for our purposes, directly minimizing lasso function of Equation (10) with respect to $\boldsymbol{\lambda}$ is not possible when a very large number of model parameters are employed, as in our task of LM for IME. Therefore we investigate below two methods that closely approximate the effect of the lasso, and are very similar to the boosting algorithm.

It is also worth noting the difference between L_1 and L_2 penalty. The classical Ridge Regression setting uses an L_2 penalty in Equation (10) i.e., $T(\boldsymbol{\lambda}) = \sum_{d=0...D}(\lambda_d)^2$, which is much easier to minimize (for least square loss but not for ExpLoss). However, recent research (Donoho et al. 1995) shows that the L_1 penalty is better suited for sparse situations, where there are only a small number of features with nonzero weights among all candidate features. We find that our task is indeed a sparse situation: among 860,000 features, in the resulting linear model only around 5,000 features have nonzero weights. We then focus on the L_1 penalty. We leave the empirical comparison of the L_1 and L_2 penalty on the LM task to future work.

4.1 Forward Stagewise Linear Regression (FSLR)

The first approximation method we used is FSLR, described in (Algorithm 10.4, Hastie et al. 2001), where Steps 2 and 3 in Figure 1 are performed according to Equations (7) and (11), respectively.

$$(k^*, \delta^*) = \arg\min_{k,\delta} \text{ExpLoss}(\text{Upd}(\boldsymbol{\lambda}, k, \delta)) \quad (7)$$

$$\boldsymbol{\lambda}^t = \text{Upd}(\boldsymbol{\lambda}^{t-1}, k^*, \varepsilon \times \text{sign}(\delta^*)) \quad (11)$$

Notice that FSLR is very similar to the boosting algorithm with shrinkage in that at each step, the feature f_{k^*} that has largest estimated impact on reducing ExpLoss is selected. The only difference is that FSLR updates the weight of f_{k^*} by a small fixed step size ε. By taking such small steps, FSLR imposes some implicit regularization, and can closely approximate the effect of the lasso in a local sense (Hastie et al. 2001). Empirically, we find that the performance of the boosting algorithm with shrinkage closely resembles that of FSLR, with the learning rate parameter ν corresponding to ε.

4.2 Boosted Lasso (BLasso)

The second method we used is a modified version of the BLasso algorithm described in Zhao and Yu (2004). There are two major differences between BLasso and FSLR. At each iteration, BLasso can take either a *forward step* or a *backward step*. Similar to the boosting algorithm and FSLR, at each forward step, a feature is selected and its weight is updated according to Equations (12) and (13).

$$(k^*, \delta^*) = \arg\min_{k,\delta=\pm\varepsilon} \mathrm{ExpLoss}(\mathrm{Upd}(\boldsymbol{\lambda}, k, \delta)) \qquad (12)$$

$$\boldsymbol{\lambda}^t = \mathrm{Upd}(\boldsymbol{\lambda}^{t-1}, k^*, \varepsilon \times \mathrm{sign}(\delta^*)) \qquad (13)$$

However, there is an important difference between Equations (12) and (7). In the boosting algorithm with shrinkage and FSLR, as shown in Equation (7), a feature is selected by its impact on reducing the loss with its optimal update δ^*. In contract, in BLasso, as shown in Equation (12), the optimization over δ is removed, and for each feature, its loss is calculated with an update of either $+\varepsilon$ or $-\varepsilon$, i.e., the grid search is used for feature selection. We will show later that this seemingly trivial difference brings a significant improvement.

The backward step is unique to BLasso. In each iteration, a feature is selected and its weight is updated backward if and only if it leads to a decrease of the lasso loss, as shown in Equations (14) and (15):

$$k^* = \arg\min_{k,\lambda_k \neq 0} \mathrm{ExpLoss}(\mathrm{Upd}(\boldsymbol{\lambda}, k, -\mathrm{sign}(\lambda_k) \times \varepsilon) \qquad (14)$$

$$\boldsymbol{\lambda}^t = \mathrm{Upd}(\boldsymbol{\lambda}^{t-1}, k^*, -\mathrm{sign}(\lambda_{k^*}) \times \varepsilon) \qquad (15)$$

$$\text{if } \mathrm{LassoLoss}(\boldsymbol{\lambda}^{t-1}, \alpha^{t-1}) - \mathrm{LassoLoss}(\boldsymbol{\lambda}^t, \alpha^t) > \theta$$

where θ is a tolerance parameter.

Figure 2 summarizes the BLasso algorithm we used. After initialization, Steps 4 and 5 are repeated N times; at each iteration, a feature is chosen and its weight is updated either backward or forward by a fixed amount ε. Notice that the value of α is adaptively chosen according to the reduction of ExpLoss during training. The algorithm starts with a large initial α, and then at each forward step the value of α decreases until the ExpLoss stops decreasing. This is intuitively desirable: It is expected that most highly effective features are selected in early stages of training, so the reduction of ExpLoss at each step in early stages are more substantial than in later stages. These early steps coincide with the boosting steps most of the time. In other words, the effect of backward steps is more visible at later stages.

Our implementation of BLasso differs slightly from the original algorithm described in Zhao and Yu (2004). Firstly, because the value of the base feature f_0 is the log probability (assigned by a word trigram model) and has a different range from that of other features as in Equation (2), λ_0 is set to optimize ExpLoss in the initialization step (Step 1 in Figure 2) and remains fixed during training. As suggested by Collins and Koo (2005), this ensures that the contribution of the log-likelihood feature f_0 is well-calibrated with respect to ExpLoss. Secondly, when updating a feature weight, if the size of the optimal update step (computed via Equation (7)) is smaller than ε, we use the optimal step to update the feature. Therefore, in our implementation BLasso does not always take a fixed step; it may take steps whose size is smaller than ε. In our initial experiments we found that both changes (also used in our implementations of boosting and FSLR) were crucial to the performance of the methods.

```
1  Initialize $\boldsymbol{\lambda}^0$: set $\lambda_0 = \mathrm{argmin}_{\lambda_0}\mathrm{ExpLoss}(\boldsymbol{\lambda})$, and $\lambda_d = 0$
   for $d=1\ldots D$.
2  Take a forward step according to Eq. (12) and (13),
   and the updated model is denoted by $\boldsymbol{\lambda}^1$
3  Initialize $\alpha = (\mathrm{ExpLoss}(\boldsymbol{\lambda}^0)\text{-}\mathrm{ExpLoss}(\boldsymbol{\lambda}^1))/\varepsilon$
4  Take a backward step if and only if it leads to a
   decrease of LassoLoss according to Eq. (14) and
   (15), where $\theta = 0$; otherwise
5  Take a forward step according to Eq. (12) and (13);
   update $\alpha = \min(\alpha, (\mathrm{ExpLoss}(\boldsymbol{\lambda}^{t-1})\text{-}\mathrm{ExpLoss}(\boldsymbol{\lambda}^t))/\varepsilon)$;
   and return to Step 4.
```

Figure 2: The BLasso algorithm

(Zhao and Yu 2004) provides theoretical justifications for BLasso. It has been proved that (1) it guarantees that it is safe for BLasso to start with an initial α which is the largest α that would allow an ε step away from 0 (i.e., larger α's correspond to $T(\boldsymbol{\lambda})=0$); (2) for each value of α, BLasso performs coordinate descent (i.e., reduces ExpLoss by updating the weight of a feature) until there is no descent step; and (3) for each step where the value of α decreases, it guarantees that the lasso loss is reduced. As a result, it can be proved that for a finite number of features and $\theta = 0$, the BLasso algorithm shown in Figure 2 converges to the lasso solution when $\varepsilon \rightarrow 0$.

5 Evaluation

5.1 Settings

We evaluated the training methods described above in the so-called cross-domain language model adaptation paradigm, where we adapt a model trained on one domain (which we call the *background* domain) to a different domain (*adaptation* domain), for which only a small amount of training data is available.

The data sets we used in our experiments came from five distinct sources of text. A 36-million-word Nikkei Newspaper corpus was used as the background domain, on which the word trigram model was trained. We used four adaptation domains: Yomiuri (newspaper corpus), TuneUp (balanced corpus containing newspapers and other sources of text), Encarta (encyclopedia) and Shincho (collection of novels). All corpora have been pre-word-segmented using a lexicon containing 167,107 entries. For each of the four domains, we created training data consisting of 72K sentences (0.9M~1.7M words) and test data of 5K sentences (65K~120K words) from each adaptation domain. The first 800 and 8,000 sentences of each adaptation training data were also used to show how different sizes of training data affected the performances of various adaptation methods. Another 5K-sentence subset was used as held-out data for each domain.

We created the training samples for discriminative learning as follows. For each phonetic string A in adaptation training data, we produced a lattice of candidate word strings W using the baseline system described in (Gao et al. 2002), which uses a word trigram model trained via MLE on the Nikkei Newspaper corpus. For efficiency, we kept only the best 20 hypotheses in its candidate conversion set **GEN**(A) for each training sample for discriminative training. The oracle best hypothesis, which gives the minimum number of errors, was used as the reference transcript of A.

We used unigrams and bigrams that occurred more than once in the training set as features in the linear model of Equation (2). The total number of candidate features we used was around 860,000.

5.2 Main Results

Table 1 summarizes the results of various model training (adaptation) methods in terms of CER (%) and CER reduction (in parentheses) over comparing models. In the first column, the numbers in parentheses next to the domain name indicates the number of training sentences used for adaptation.

Baseline, with results shown in Column 3, is the word trigram model. As expected, the CER correlates very well the similarity between the background domain and the adaptation domain, where domain similarity is measured in terms of cross entropy (Yuan et al. 2005) as shown in Column 2.

MAP (maximum *a posteriori*), with results shown in Column 4, is a traditional LM adaptation method where the parameters of the background model are adjusted in such a way that maximizes the likelihood of the adaptation data. Our implementation takes the form of linear interpolation as described in Bacchiani et al. (2004): $P(w_i|h) = \lambda P_b(w_i|h) + (1\text{-}\lambda)P_a(w_i|h)$, where P_b is the probability of the background model, P_a is the probability trained on adaptation data using MLE and the history h corresponds to two preceding words (i.e. P_b and P_a are trigram probabilities). λ is the interpolation weight optimized on held-out data.

Boosting, with results shown in Column 5, is the algorithm described in Figure 1. In our implementation, we use the shrinkage method suggested by Schapire and Singer (1999) and Collins and Koo (2005). At each iteration, we used the following update for the kth feature

$$\delta_k = \frac{1}{2}\log\frac{C_k^+ + \varepsilon Z}{C_k^- + \varepsilon Z} \qquad (16)$$

where C_k^+ is a value increasing exponentially with the sum of margins of (W^R, W) pairs over the set where f_k is seen in W^R but not in W; C_k^- is the value related to the sum of margins over the set where f_k is seen in W but not in W^R. ε is a smoothing factor (whose value is optimized on held-out data) and Z is a normalization constant (whose value is the ExpLoss(.) of training data according to the current model). We see that εZ in Equation (16) plays the same role as ν in Equation (9).

BLasso, with results shown in Column 6, is the algorithm described in Figure 2. We find that the performance of BLasso is not very sensitive to the selection of the step size ε across training sets of different domains and sizes. Although small ε is preferred in theory as discussed earlier, it would lead to a very slow convergence. Therefore, in our experiments, we always use a large step (ε = 0.5) and use the so-called early stopping strategy, i.e., the number of iterations before stopping is optimized on held-out data.

In the task of LM for IME, there are millions of features and training samples, forming an extremely large and sparse matrix. We therefore applied the techniques described in Collins and Koo (2005) to speed up the training procedure. The resulting algorithms run in around 15 and 30 minutes respectively for Boosting and BLasso to converge on an XEON™ MP 1.90GHz machine when training on an 8K-sentnece training set.

The results in Table 1 give rise to several observations. First of all, both discriminative training methods (i.e., Boosting and BLasso) outperform MAP substantially. The improvement margins are larger when the background and adaptation domains are more similar. The phenomenon is attributed to the underlying difference between the two adaptation methods: MAP aims to improve the likelihood of a distribution, so if the adaptation domain is very similar to the background domain, the difference between the two underlying distributions is so small that MAP cannot adjust the model effectively. Discriminative methods, on the other hand, do not have this limitation for they aim to reduce errors directly. Secondly, BLasso outperforms Boosting significantly (p-value < 0.01) on all test sets. The improvement margins vary with the training sets of different domains and sizes. In general, in cases where the adaptation domain is less similar to the background domain and larger training set is used, the improvement of BLasso is more visible.

Note that the CER results of FSLR are not included in Table 1 because it achieves very similar results to the boosting algorithm with shrinkage if the controlling parameters of both algorithms are optimized via cross-validation. We shall discuss their difference in the next section.

5.3 Dicussion

This section investigates what components of BLasso bring the improvement over Boosting. Comparing the algorithms in Figures 1 and 2, we notice three differences between BLasso and Boosting: (i) the use of backward steps in BLasso; (ii) BLasso uses the grid search (fixed step size) for feature selection in Equation (12) while Boosting uses the continuous search (optimal step size) in Equation (7); and (iii) BLasso uses a fixed step size for feature update in Equation (13) while Boosting uses an optimal step size in Equation (8). We then investigate these differences in turn.

To study the impact of backward steps, we compared BLasso with the boosting algorithm with a fixed step search and a fixed step update, henceforth referred to as **F-Boosting**. F-Boosting was implemented as Figure 2, by setting a large value to θ in Equation (15), i.e., $\theta = 10^3$, to prohibit backward steps. We find that although the training error curves of BLasso and F-Boosting are almost identical, the $T(\boldsymbol{\lambda})$ curves grow apart with iterations, as shown in Figure 3. The results show that with backward steps, BLasso achieves a better approximation to the true lasso solution: It leads to a model with similar training errors but less complex (in terms of L_1 penalty). In our experiments we find that the benefit of using backward steps is only visible in later iterations when BLasso's backward steps kick in. A typical example is shown in Figure 4. The early steps fit to highly effective features and in these steps BLasso and F-Boosting agree. For later steps, fine-tuning of features is required. BLasso with backward steps provides a better mechanism than F-Boosting to revise the previously chosen features to accommodate this fine level of tuning. Consequently we observe the superior performance of BLasso at later stages as shown in our experiments.

As well-known in linear regression models, when there are many strongly correlated features, model parameters can be poorly estimated and exhibit high variance. By imposing a model size constraint, as in lasso, this phenomenon is alleviated. Therefore, we speculate that a better approximation to lasso, as BLasso with backward steps, would be superior in eliminating the negative effect of strongly correlated features in model estimation. To verify our speculation, we performed the following experiments. For each training set, in addition to word unigram and bigram features, we introduced a new type of features called *headword bigram*.

As described in Gao et al. (2002), headwords are defined as the content words of the sentence. Therefore, headword bigrams constitute a special type of skipping bigrams which can capture dependency between two words that may not be adjacent. In reality, a large portion of headword bigrams are identical to word bigrams, as two headwords can occur next to each other in text. In the adaptation test data we used, we find that headword bigram features are for the most part either *completely overlapping* with the word bigram features (i.e., all instances of headword bigrams also count as word bigrams) or *not overlapping at all* (i.e., a headword bigram feature is not observed as a word bigram feature) – less than 20% of headword bigram features displayed a variable degree of overlap with word bigram features. In our data, the rate of completely overlapping features is 25% to 47% depending on the adaptation domain. From this, we can say that the headword bigram features show moderate to high degree of correlation with the word bigram features.

We then used BLasso and F-Boosting to train the linear language models including both word bigram and headword bigram features. We find that although the CER reduction by adding headword features is overall very small, the difference between the two versions of BLasso is more visible in all four test sets. Comparing Figures 5 – 8 with Figure 4, it can be seen that BLasso with backward steps outperforms the one without backward steps in much earlier stages of training with a larger margin. For example, on Encarta data sets, BLasso outperforms F-Boosting after around 18,000 iterations with headword features (Figure 7), as opposed to 25,000 iterations without headword features (Figure 4). The results seem to corroborate our speculation that BLasso is more robust in the presence of highly correlated features.

To investigate the impact of using the grid search (fixed step size) versus the continuous search (optimal step size) for feature selection, we compared F-Boosting with FSLR since they differs only in their search methods for feature selection. As shown in Figures 5 to 8, although FSLR is robust in that its test errors do not increase after many iterations, F-Boosting can reach a much lower error rate on three out of four test sets. Therefore, in the task of LM for IME where CER is the most important metric, the grid search for feature selection is more desirable.

To investigate the impact of using a fixed versus an optimal step size for feature update, we compared FSLR with Boosting. Although both algorithms achieve very similar CER results, the performance of FSLR is much less sensitive to the selected fixed step size. For example, we can select any value from 0.2 to 0.8, and in most settings FSLR achieves the very similar lowest CER after 20,000 iterations, and will stay there for many iterations. In contrast, in Boosting, the optimal value of ε in Equation (16) varies with the sizes and domains of training data, and has to be tuned carefully. We thus conclude that in our task FSLR is more robust against different training settings and a fixed step size for feature update is more preferred.

6 Conclusion

This paper investigates two approximation lasso methods for LM applied to a realistic task with a very large number of features with sparse feature space. Our results on Japanese text input are promising. BLasso outperforms the boosting algorithm significantly in terms of CER reduction on all experimental settings.

We have shown that this superior performance is a consequence of BLasso's backward step and its fixed step size in both feature selection and feature weight update. Our experimental results in Section 5 show that the use of backward step is vital for model fine-tuning after major features are selected and for coping with strongly correlated features; the fixed step size of BLasso is responsible for the improvement of CER and the robustness of the results. Experiments on other data sets and theoretical analysis are needed to further support our findings in this paper.

References

Bacchiani, M., Roark, B., and Saraclar, M. 2004. Language model adaptation with MAP estimation and the perceptron algorithm. In *HLT-NAACL 2004*. 21-24.

Collins, Michael and Terry Koo 2005. Discriminative reranking for natural language parsing. *Computational Linguistics* 31(1): 25-69.

Duda, Richard O, Hart, Peter E. and Stork, David G. 2001. *Pattern classification*. John Wiley & Sons, Inc.

Donoho, D., I. Johnstone, G. Kerkyachairan, and D. Picard. 1995. Wavelet shrinkage; asymptopia? (with discussion), *J. Royal. Statist. Soc.* 57: 201-337.

Efron, B., T. Hastie, I. Johnstone, and R. Tibshirani. 2004. Least angle regression. *Ann. Statist.* 32, 407-499.

Freund, Y, R. Iyer, R. E. Schapire, and Y. Singer. 1998. An efficient boosting algorithm for combining preferences. In *ICML'98*.

Hastie, T., R. Tibshirani and J. Friedman. 2001. *The elements of statistical learning*. Springer-Verlag, New York.

Gao, Jianfeng, Hisami Suzuki and Yang Wen. 2002. Exploiting headword dependency and predictive clustering for language modeling. In *EMNLP 2002*.

Gao. J., Yu, H., Yuan, W., and Xu, P. 2005. Minimum sample risk methods for language modeling. In *HLT/EMNLP 2005*.

Osborne, M.R. and Presnell, B. and Turlach B.A. 2000a. A new approach to variable selection in least squares problems. *Journal of Numerical Analysis*, 20(3).

Osborne, M.R. and Presnell, B. and Turlach B.A. 2000b. On the lasso and its dual. *Journal of Computational and Graphical Statistics*, 9(2): 319-337.

Roark, Brian, Murat Saraclar and Michael Collins. 2004. Corrective language modeling for large vocabulary ASR with the perceptron algorithm. In *ICASSP 2004*.

Schapire, Robert E. and Yoram Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3): 297-336.

Suzuki, Hisami and Jianfeng Gao. 2005. A comparative study on language model adaptation using new evaluation metrics. In *HLT/EMNLP 2005*.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *J. R. Statist. Soc. B*, 58(1): 267-288.

Yuan, W., J. Gao and H. Suzuki. 2005. An Empirical Study on Language Model Adaptation Using a Metric of Domain Similarity. In *IJCNLP 05*.

Zhao, P. and B. Yu. 2004. Boosted lasso. *Tech Report*, Statistics Department, U. C. Berkeley.

Zhu, J. S. Rosset, T. Hastie, and R. Tibshirani. 2003. 1-norm support vector machines. *NIPS 16*. MIT Press.

Table 1. CER (%) and CER reduction (%) (Y=Yomiuri; T=TuneUp; E=Encarta; S=-Shincho)

Domain	Entropy vs.Nikkei	Baseline	MAP (over Baseline)	Boosting (over MAP)	BLasso (over MAP/Boosting)
Y (800)	7.69	3.70	3.70 (+0.00)	3.13 (+15.41)	3.01 (+18.65/+3.83)
Y (8K)	7.69	3.70	3.69 (+0.27)	2.88 (+21.95)	2.85 (+22.76/+1.04)
Y (72K)	7.69	3.70	3.69 (+0.27)	2.78 (+24.66)	2.73 (+26.02/+1.80)
T (800)	7.95	5.81	5.81 (+0.00)	5.69 (+2.07)	5.63 (+3.10/+1.05)
T (8K)	7.95	5.81	5.70 (+1.89)	5.48 (+5.48)	5.33 (+6.49/+2.74)
T (72K)	7.95	5.81	5.47 (+5.85)	5.33 (+2.56)	5.05 (+7.68/+5.25)
E (800)	9.30	10.24	9.60 (+6.25)	9.82 (-2.29)	9.18 (+4.38/+6.52)
E (8K)	9.30	10.24	8.64 (+15.63)	8.54 (+1.16)	8.04 (+6.94/+5.85)
E (72K)	9.30	10.24	7.98 (+22.07)	7.53 (+5.64)	7.20 (+9.77/+4.38)
S (800)	9.40	12.18	11.86 (+2.63)	11.91 (-0.42)	11.79 (+0.59/+1.01)
S (8K)	9.40	12.18	11.15 (+8.46)	11.09 (+0.54)	10.73 (+3.77/+3.25)
S (72K)	9.40	12.18	10.76 (+11.66)	10.25 (+4.74)	9.64 (+10.41/+5.95)

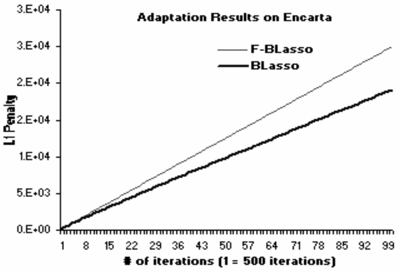


Figure 3. L_1 curves: models are trained on the E(8K) dataset.

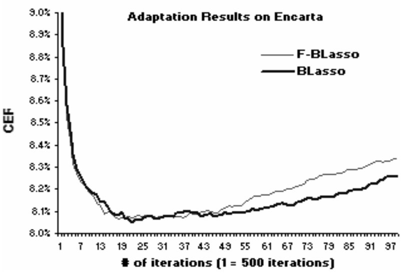


Figure 4. Test error curves: models are trained on the E(8K) dataset.

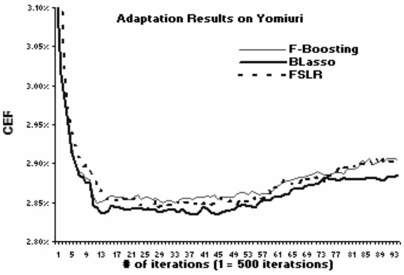


Figure 5. Test error curves: models are trained on the Y(8K) dataset, including headword bigram features.

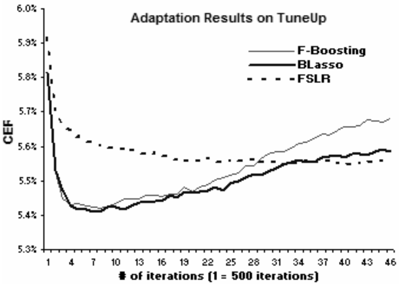


Figure 6. Test error curves: models are trained on the T(8K) dataset, including headword bigram features.

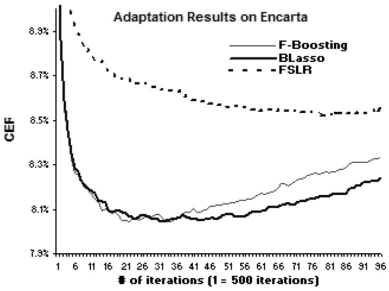


Figure 7. Test error curves: models are trained on the E(8K) dataset, including headword bigram features.

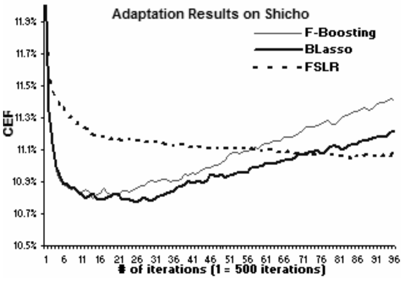


Figure 8. Test error curves: models are trained on the S(8K) dataset, including headword bigram features.

Automated Japanese Essay Scoring System based on Articles Written by Experts

Tsunenori Ishioka
Research Division
The National Center for
University Entrance Examinations
Tokyo 153-8501, Japan
tunenori@rd.dnc.ac.jp

Masayuki Kameda
Software Research Center
Ricoh Co., Ltd.
Tokyo 112-0002, Japan
masayuki.kameda@nts.ricoh.co.jp

Abstract

We have developed an automated Japanese essay scoring system called Jess. The system needs expert writings rather than expert raters to build the evaluation model. By detecting statistical outliers of predetermined aimed essay features compared with many professional writings for each prompt, our system can evaluate essays. The following three features are examined: (1) rhetoric — syntactic variety, or the use of various structures in the arrangement of phases, clauses, and sentences, (2) organization — characteristics associated with the orderly presentation of ideas, such as rhetorical features and linguistic cues, and (3) content — vocabulary related to the topic, such as relevant information and precise or specialized vocabulary. The final evaluation score is calculated by deducting from a perfect score assigned by a learning process using editorials and columns from the Mainichi Daily News newspaper. A diagnosis for the essay is also given.

1 Introduction

When giving an essay test, the examiner expects a written essay to reflect the writing ability of the examinee. A variety of factors, however, can affect scores in a complicated manner. Cooper (1984) states that “various factors including the writer, topic, mode, time limit, examination situation, and rater can introduce error into the scoring of essays used to measure writing ability.” Most of these factors are present in giving tests, and the human rater, in particular, is a major error factor in the scoring of essays.

In fact, many other factors influence the scoring of essay tests, as listed below, and much research has been devoted.

- Handwriting skill (handwriting quality, spelling) (Chase, 1979; Marshall and Powers, 1969)
- Serial effects of rating (the order in which essay answers are rated) (Hughes et al., 1983)
- Topic selection (how should essays written on different topics be rated?) (Meyer, 1939)
- Other error factors (writer’s gender, ethnic group, etc.) (Chase, 1986)

In recent years, with the aim of removing these error factors and establishing fairness, considerable research has been performed on computer-based automated essay scoring (AES) systems (Burstein et al., 1998; Foltz et al., 1999; Page et al., 1997; Powers et al., 2000; Rudner and Liang, 2002).

The AES systems provide the users with prompt feedback to improve their writings. Therefore, many practical AES systems have been used. E-rater (Burstein et al., 1998), developed by the Educational Testing Service, began being used for operational scoring of the Analytical Writing Assessment in the Graduate Management Admission Test (GMAT), an entrance examination for business graduate schools, in February 1999, and it has scored approximately 360,000 essays per year. The system includes several independent NLP-based modules for identifying features relevant to the scoring guide from three categories: syntax, discourse, and topic. Each of the feature-recognition modules correlate the essay scores with assigned by human readers. E-rater uses a model-building module to select and weight predictive features for essay scoring. Project Essay

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 233–240,
Sydney, July 2006.

Grade (PEG), which was the first automated essay scorer, uses a regression model like e-rater (Page et al., 1997). IntelliMetric (Elliot, 2003) was first commercially released by Vantage Learning in January 1998 and was the first AI-based essay-scoring tool available to educational agencies. The system internalizes the pooled wisdom of many expert scorers. The Intelligent Essay Assessor (IEA) is a set of software tools for scoring the quality of the conceptual content of essays based on latent semantic analysis (Foltz et al., 1999). The Bayesian Essay Test Scoring sYstem (BETSY) is a windows-based program that classifies text based on trained material. The features include multi-nomial and Bernoulli Naive Bayes models (Rudner and Liang, 2002).

Note that all above-mentioned systems are based on the assumption that the true quality of essays must be defined by human judges. However, Bennet and Bejar (1998) have criticized the overreliance on human ratings as the sole criterion for evaluating computer performance because ratings are typically based as a constructed rubric that may ultimately achieve acceptable reliability at the cost of validity. In addition, Friedman, in research during the 1980s, found that holistic ratings by human raters did not award particularly high marks to professionally written essays mixed in with student productions. This is a kind of negative halo effect: create a bad impression, and you will be scored low on everything. Thus, Bereiter (2003) insists that another approach to doing better than ordinary human raters would be to use expert writers rather than expert raters. Reputable professional writers produce sophisticated and easy-to-read essays. The use of professional writings as the criterion, whether the evaluation is based on holistic or trait rating, has an advantage, described below.

The methods based on expert rater evaluations require much effort to set up the model for each prompt. For example, e-rater and PEG use some sort of regression approaches in setting up the statistical models. Depending on how many variables are involved, these models may require thousands of cases to derive stable regression weights. BETSY requires the Bayesian rules, and IntelliMetric, the AI-based rules. Thus, the methodology limits the grader's practical utility to large-scale testing operations in which such data collection is feasible. On the other hand, a method based on professional writings can overcome this; i.e., in our system, we need not set up a model simulating a human rater because thousands of articles by professional writers can easily be obtained via various electronic media. By detecting a statistical outlier to predetermined essay features compared with many professional writings for each prompt, our system can evaluate essays.

In Japan, it is possible to obtain complete articles from the Mainichi Daily News newspaper up to 2005 from Nichigai Associates, Inc. and from the Nihon Keizai newspaper up to 2004 from Nikkei Books and Software, Inc. for purposes of linguistic study. In short, it is relatively easy to collect editorials and columns (e.g., "Yoroku") on some form of electronic media for use as essay models. Literary works in the public domain can be accessed from Aozora Bunko (http://www.aozora.gr.jp/). Furthermore, with regard to morphological analysis, the basis of Japanese natural language processing, a number of free Japanese morphological analyzers are available. These include JUMAN (http://www-lab25.kuee.kyoto-u.ac.jp/nlresource/juman.html), developed by the Language Media Laboratory of Kyoto University, and ChaSen (http://chasen.aist-nara.ac.jp/, used in this study) from the Matsumoto Laboratory of the Nara Institute of Science and Technology.

Likewise, for syntactic analysis, free resources are available such as KNP (http://www-lab25.kuee.kyoto-u.ac.jp/nlresource/knp.html) from Kyoto University, SAX and BUP (http://cactus.aist-nara.ac.jp/lab/nlt/{sax,bup}.html) from the Nara Institute of Science and Technology, and the MSLR parser (http://tanaka-www.cs.titech.ac.jp/pub/mslr/index-j.html) from the Tanaka Tokunaga Laboratory of the Tokyo Institute of Technology. With resources such as these, we prepared tools for computer processing of the articles and columns that we collect as essay models.

In addition, for the scoring of essays, where it is essential to evaluate whether content is suitable, i.e., whether a written essay responds appropriately to the essay prompt, it is becoming possible for us to use semantic search technologies not based on pattern matching as used by search engines on the Web. The methods for implementing such technologies are explained in detail by Ishioka and Kameda (1999) and elsewhere. We believe that this statistical outlier detection approach to using published professional essays and columns as models makes it possible to develop a system essentially superior to other AES systems.

We have named this automated Japanese essay scoring system "Jess." This system evaluates essays based on three features : (1) rhetoric, (2) organization, and (3) content, which are basically the same as the structure, organization, and content used by e-rater. Jess also allows the user to designate weights (allotted points) for each of these essay features. If the user does not explicitly specify the point allotment, the default weights are 5, 2, and 3 for structure, organization, and content, respectively, for a total of 10 points. (Incidentally, a perfect score in e-rater is 6.) This default point allotment in which "rhetoric" is weighted higher than "organization" and "content" is based on the work of Watanabe et al. (1988). In that research, 15 criteria were given for scoring essays: (1) wrong/omitted characters, (2) strong vocabulary, (3) character usage, (4) grammar, (5) style, (6) topic relevance, (7) ideas, (8) sentence structure, (9) power of expression, (10) knowledge, (11) logic/consistency, (12) power of thinking/judgment, (13) complacency, (14) nuance, and (15) affinity. Here, correlation coefficients were given to reflect the evaluation value of each of these criteria. For example, (3) character usage, which is deeply related to "rhetoric," turned out to have the highest correlation coefficient at 0.58, and (1) wrong/omitted characters was relatively high at 0.36. In addition, (8) sentence structure and (11) logic/consistency, which is deeply related to "organization," had correlation coefficients of 0.32 and 0.26, respectively, both lower than that of "rhetoric," and (6) topic relevance and (14) nuance, which are though to be deeply related to "content," had correlation coefficients of 0.27 and 0.32, respectively.

Our system, Jess, is the first automated Japanese essay scorer and has become most famous in Japan, since it was introduced in February 2005 in a headline in the Asahi Daily News, which is well known as the most reliable and most representative newspaper of Japan.

The following sections describe the scoring criteria of Jess in detail. Sections 2, 3, and 4 examine rhetoric, organization, and content, respectively. Section 5 presents an application example and associated operation times, and section 6 concludes the paper.

2 Rhetoric

As metrics to portray rhetoric, Jess uses (1) ease of reading, (2) diversity of vocabulary, (3) percentage of big words (long, difficult words), and (4) percentage of passive sentences, in accordance with Maekawa (1995) and Nagao (1996). These metrics are broken down further into various statistical quantities in the following sections. The distributions of these statistical quantities were obtained from the editorials and columns stored on the Mainichi Daily News CD-ROMs.

Though most of these distributions are asymmetrical (skewed), they are each treated as a distribution of an ideal essay. In the event that a score (obtained statistical quantity) turns out to be an outlier value with respect to such an ideal distribution, that score is judged to be "inappropriate" for that metric. The points originally allotted to the metric are then reduced, and a comment to that effect is output. An "outlier" is an item of data more than 1.5 times the interquartile range. (In a box-and-whisker plot, whiskers are drawn up to the maximum and minimum data points within 1.5 times the interquartile range.) In scoring, the relative weights of the broken-down metrics are equivalent with the exception of "diversity of vocabulary," which is given a weight twice that of the others because we consider it an index contributing to not only "rhetoric" but to "content" as well.

2.1 Ease of reading

The following items are considered indexes of "ease of reading."

1. Median and maximum sentence length

   Shorter sentences are generally assumed to make for easier reading (Knuth et al., 1988). Many books on writing in the Japanese language, moreover, state that a sentence should be no longer than 40 or 50 characters. Median and maximum sentence length can therefore be treated as an index. The reason the median value is used as opposed to the average is that sentence-length distributions are skewed in most cases. The relative weight used in the evaluation of median and maximum sentence length is equivalent to that of the indexes described below. Sentence length is also known to be quite effective for determining style.

2. Median and maximum clause length

In addition to periods (.), commas (,) can also contribute to ease of reading. Here, text between commas is called a "clause." The number of characters in a clause is also an evaluation index.

3. Median and maximum number of phrases in clauses

   A human being cannot understand many things at one time. The limit of human short-term memory is said to be seven things in general, and that is thought to limit the length of clauses. Actually, on surveying the number of phrases in clauses from editorials in the Mainichi Daily News, we found it to have a median of four, which is highly compatible with the short-term memory maximum of seven things.

4. Kanji/kana ratio

   To simplify text and make it easier to read, a writer will generally reduce kanji (Chinese characters) intentionally. In fact, an appropriate range for the kanji/kana ratio in essays is thought to exist, and this range is taken to be an evaluation index. The kanji/kana ratio is also thought to be one aspect of style.

5. Number of attributive declined or conjugated words (embedded sentences)

   The declined or conjugated forms of attributive modifiers indicate the existence of "embedded sentences," and their quantity is thought to affect ease of understanding.

6. Maximum number of consecutive infinitive-form or conjunctive-particle clauses

   Consecutive infinitive-form or conjunctive-particle clauses, if many, are also thought to affect ease of understanding. Note that not this "average size" but "maximum number" of consecutive infinitive-form or conjunctive-particle clauses holds significant meaning as an indicator of the depth of dependency affecting ease of understanding.

2.2 Diversity of vocabulary

Yule (1944) used a variety of statistical quantities in his analysis of writing. The most famous of these is an index of vocabulary concentration called the K characteristic value. The value of K is non-negative, increases as vocabulary becomes more concentrated, and conversely, decreases as vocabulary becomes more diversified. The median values of K for editorials and columns in the Mainichi Daily News were found to be 87.3 and 101.3, respectively. Incidentally, other characteristic values indicating vocabulary concentration have been proposed. See Tweedie et al. (1998), for example.

2.3 Percentage of big words

It is thought that the use of big words, to whatever extent, cannot help but impress the reader. On investigating big words in Japanese, however, care must be taken because simply measuring the length of a word may lead to erroneous conclusions. While "big word" in English is usually synonymous with "long word," a word expressed in kanji becomes longer when expressed in kana characters. That is to say, a "small word" in Japanese may become a big word simply due to notation. The number of characters in a word must therefore be counted after converting it to kana characters (i.e., to its "reading") to judge whether that word is big or small. In editorials from the Mainichi Daily News, the median number of characters in nouns after conversion to kana was found to be 4, with 5 being the 3rd quartile (upper 25%). We therefore assumed for the time being that nouns having readings of 6 or more characters were big words, and with this as a guideline, we again measured the percentage of nouns in a document that were big words. Since the number of characters in a reading is an integer value, this percentage would not necessarily be 25%, but a distribution that takes a value near that percentage on average can be obtained.

2.4 Percentage of passive sentences

It is generally felt that text should be written in active voice as much as possible, and that text with many passive sentences is poor writing (Knuth et al., 1988). For this reason, the percentage of passive sentences is also used as an index of rhetoric. Grammatically speaking, passive voice is distinguished from active voice in Japanese by the auxiliary verbs "reru" and "rareru". In addition to passivity, however, these two auxiliary verbs can also indicate respect, possibility, and spontaneity. In fact, they may be used to indicate respect even in the case of active voice. This distinction, however, while necessary in analysis at the semantic level, is not used in morphological analysis and syntactic analysis. For example, in the case that the object of respect is “teacher” (sensei) or “your husband” (goshujin), the use of “reru” and “rareru” auxiliary verbs here would certainly indicate respect. This meaning, however, belongs entirely to the world of semantics. We can assume that such an indication of respect would not be found in essays required for tests, and consequently, that the use of “reru” and “rareru” in itself would indicate the passive voice in such an essay.

3 Organization

Comprehending the flow of a discussion is essential to understanding the connection between various assertions. To help the reader to catch this flow, the frequent use of conjunctive expressions is useful. In Japanese writing, however, the use of conjunctive expressions tends to alienate the reader, and such expressions, if used at all, are preferably vague. At times, in fact, presenting multiple descriptions or posing several questions seeped in ambiguity can produce interesting effects and result in a beautiful passage (Noya, 1997). In essays tests, however, examinees are not asked to come up with “beautiful passages.” They are required, rather, to write logically while making a conscious effort to use conjunctive expressions. We therefore attempt to determine the logical structure of a document by detecting the occurrence of conjunctive expressions. In this effort, we use a method based on cue words as described in Quirk et al. (1985) for measuring the organization of a document. This method, which is also used in e-rater, the basis of our system, looks for phrases like “in summary” and “in conclusion” that indicate summarization, and words like “perhaps” and “possibly” that indicate conviction or thinking when examining a matter in depth, for example. Now, a conjunctive relationship can be broadly divided into “forward connection” and “reverse connection.” “Forward connection” has a rather broad meaning indicating a general conjunctive structure that leaves discussion flow unchanged. In contrast, “reverse connection” corresponds to a conjunctive relationship that changes the flow of discussion. These logical structures can be classified as follows according to Noya (1997). The “forward connection” structure comes in the following types.

Addition: A conjunctive relationship that adds emphasis. A good example is “in addition,” while other examples include “moreover” and “rather.” Abbreviation of such words is not infrequent.

Explanation: A conjunctive relationship typified by words and phrases such as “namely,” “in short,” “in other words,” and “in summary.” It can be broken down further into “summarization” (summarizing and clarifying what was just described), “elaboration” (in contrast to “summarization,” begins with an overview followed by a detailed description), and “substitution” (saying the same thing in another way to aid in understanding or to make a greater impression).

Demonstration: A structure indicating a reason-consequence relation. Expressions indicating a reason include “because” and “the reason is,” and those indicating a consequence include “as a result,” “accordingly,” “therefore,” and “that is why.” Conjunctive particles in Japanese like “node” (since) and “kara” (because) also indicate a reason-consequence relation.

Illustration: A conjunctive relationship most typified by the phrase “for example” having a structure that either explains or demonstrates by example.

The “reverse connection” structure comes in the following types.

Transition: A conjunctive relationship indicating a change in emphasis from A to B expressed by such structures as “A ..., but B...” and “A...; however, B...).

Restriction: A conjunctive relationship indicating a continued emphasis on A. Also referred to as a “proviso” structure typically expressed by “though in fact” and “but then.”

Concession: A type of transition that takes on a conversational structure in the case of concession or compromise. Typical expressions indicating this relationship are “certainly” and “of course.”

Contrast: A conjunctive relationship typically expressed by “at the same time,” “on the other hand,” and “in contrast.”

We extracted all $(= 125)$ phrases indicating conjunctive relationships from editorials of the Mainichi Daily News, and classified them into the above four categories for forward connection and those for reverse connection for a total of eight exclusive categories. In Jess, the system attaches labels to conjunctive relationships and tallies them to judge the strength of the discourse in the essay being scored. As in the case of rhetoric, Jess learns what an appropriate number of conjunctive relationships should be from editorials of the Mainichi Daily News, and deducts from the initially allotted points in the event of an outlier value in the model distribution.

In the scoring, we also determined whether the pattern in which these conjunctive relationships appeared in the essay was singular compared to that in the model editorials. This was accomplished by considering a trigram model (Jelinek, 1991) for the appearance patterns of forward and reverse connections. In general, an N-gram model can be represented by a stochastic finite automaton, and in a trigram model, each state of an automaton is labeled by a symbol sequence of length 2. The set of symbols here is $\Sigma = \{a$: forward-connection, b : reverse-connection$\}$. Each state transition is assigned a conditional output probability as shown in Table 1. The symbol $\sqcup$ here indicates no (prior) relationship. The initial state is shown as $\sqcup\sqcup$. For example, the expression $P(a|\sqcup\sqcup)$ signifies the probability that "a : forward connection" will appear as the initial state.

Table 1: State transition probabilities on $\{a$: forward-connection, b : reverse-connection$\}$

$P(a\|\sqcup a) = 0.48$	$P(b\|\sqcup a) = 0.52$	$P(a\|\sqcup b) = 0.36$
$P(b\|\sqcup b) = 0.64$	$P(a\|aa) = 0.35$	$P(b\|aa) = 0.65$
$P(a\|ab) = 0.55$	$P(b\|ab) = 0.44$	$P(a\|ba) = 0.28$
$P(b\|ba) = 0.72$	$P(a\|bb) = 0.35$	$P(b\|bb) = 0.65$
$P(a\|\sqcup\sqcup) = 0.44$	$P(b\|\sqcup\sqcup) = 0.56$	

In this way, the probability of occurrence of certain $\{a$: forward-connection$\}$ and $\{b$: reverse-connection$\}$ patterns can be obtained by taking the product of appropriate conditional probabilities listed in Table 1. For example, the probability of occurrence p of the pattern $\{a, b, a, a\}$ turns out to be $0.44 \times 0.52 \times 0.55 \times 0.28 = 0.035$. Furthermore, given that the probability of $\{a\}$ appearing without prior information is 0.47 and that of $\{b\}$ appearing without prior information is 0.53, the probability q that a forward connection occurs three times and a reverse connection once under the condition of no prior information would be $0.47^3 \times 0.53 = 0.055$. As shown by this example, an occurrence probability that is greater for no prior information would indicate that the forward-connection and reverse-connection appearance pattern is singular, in which case the points initially allocated to conjunctive relationships in a discussion would be reduced. The trigram model may overcome the restrictions that the essay should be written in a pyramid structure or the reversal.

4 Content

A technique called latent semantic indexing can be used to check whether the content of a written essay responds appropriately to the essay prompt. The usefulness of this technique has been stressed at the Text REtrieval Conference (TREC) and elsewhere. Latent semantic indexing begins after performing singular value decomposition on $t \times d$ term-document matrix X (t : number of words; d : number of documents) indicating the frequency of words appearing in a sufficiently large number of documents. Matrix X is generally a huge sparse matrix, and SVDPACK (Berry, 1992) is known to be an effective software package for performing singular value decomposition on a matrix of this type. This package allows the use of eight different algorithms, and Ishioka and Kameda (1999) give a detailed comparison and evaluation of these algorithms in terms of their applicability to Japanese documents. Matrix X must first be converted to the Harwell-Boeing sparse matrix format (Duff et al., 1989) in order to use SVDPACK. This format can store the data of a sparse matrix in an efficient manner, thereby saving disk space and significantly decreasing data read-in time.

5 Application

5.1 An E-rater Demonstration

An e-rater demonstration can be viewed at www.ets.org, where by clicking "Products→e-rater Home→Demo." In this demonstration, seven response patterns (seven essays) are evaluated. The scoring breakdown, given a perfect score of six, was one each for scores of 6, 5, 4, and 2 and three for a score of 3.

We translated essays A-to-G on that Web site into Japanese and then scored them using Jess, as shown in Table 2.

The second and third columns show e-rater and Jess scores, respectively, and the fourth column shows the number of characters in each essay.

Table 2: Comparison of scoring results

Essay	E-rater	Jess	No. of Characters	Time (s)
A	4	6.9 (4.1)	687	1.00
B	3	5.1 (3.0)	431	1.01
C	6	8.3 (5.0)	1,884	1.35
D	2	3.1 (1.9)	297	0.94
E	3	7.9 (4.7)	726	0.99
F	5	8.4 (5.0)	1,478	1.14
G	3	6.0 (3.6)	504	0.95

A perfect score in Jess is 10 with 5 points allocated to rhetoric, 2 to organization, and 3 to content as standard. For purposes of comparison, the Jess score converted to e-rater's 6-point system is shown in parentheses. As can be seen here, essays given good scores by e-rater are also given good scores by Jess, and the two sets of scores show good agreement. However, e-rater (and probably human raters) tends to give more points to longer essays despite similar writing formats. Here, a difference appears between e-rater and Jess, which uses the point-deduction system for scoring. Examining the scores for essay C, for example, we see that e-rater gave a perfect score of 6, while Jess gave only a score of 5 after converting to e-rater's 6-point system. In other words, the length of the essay could not compensate for various weak points in the essay under Jess's point-deduction system. The fifth column in Table 2 shows the processing time (CPU time) for Jess. The computer used was Plat'Home Standard System 801S using an 800-MHz Intel Pentium III running RedHat 7.2. The Jess program is written in C shell script, jgawk, jsed, and C, and comes to just under 10,000 lines. In addition to the ChaSen morphological analysis system, Jess also needs the kakasi kanji/kana converter program (http://kakasi.namagu.org/) to operate. At present, it runs only on UNIX. Jess can be executed on the Web at http://coca.rd.dnc.ac.jp/jess/.

5.2 An Example of using a Web Entry Sheet

Four hundred eighty applicants who were eager to be hired by a certain company entered their essays using a Web form without a time restriction, with the size of the text restricted implicitly by the Web screen, to about 800 characters. The theme of the essay was "What does working mean in your life." Table 3 summarizes the correlation coefficients between the Jess score, average score of expert raters, and score of the linguistic understanding test (LUT), developed by Recruit Management Solutions Co., Ltd. The LUT is designed to measure the ability to grasp the correct meaning of words that are the elements of a sentence, and to understand the composition and the summary of a text. Five expert raters reted the essays, and three of these scored each essay independently.

Table 3: Correlation between Jess score, average of expert raters, and linguistic understanding test

	Jess	Ave. of Experts
Ave. of Experts	0.57	
LUT	0.08	0.13

We found that the correlation between the Jess score and the average of the expert raters' scores is not small (0.57), and is larger than the correlation coefficient between the expert raters' scores of 0.48. That means that Jess is superior to the expert raters on average, and is substitutable for them. Note that the restriction of the text size (800 characters in this case) caused the low correlation owing to the difficulty in evaluating the organization and the development of the arguments; the essay scores even in expert rater tend to be dispersed.

We also found that neither the expert raters nor Jess, had much correlation with LUT, which shows that LUT does not reflect features indicating writing ability. That is, LUT measures quite different laterals from writing ability.

Another experiment using 143 university-students' essays collected at the National Institute for Japanese Language shows a similar result: for the essays on "smoking," the correlation between Jess and the expert raters was 0.83, which is higher than the average correlation of expert raters (0.70); for the essays on "festivals in Japan," the former is 0.84, the latter, 0.73. Three of four raters graded each essay independently.

6 Conclusion

An automated Japanese essay scoring system called Jess has been created for scoring essays in college-entrance exams. This system has been shown to be valid for essays of 800 to 1,600 characters. Jess, however, uses editorials and columns taken from the Mainichi Daily News newspaper as learning models, and such models are not sufficient for learning terms used in scientific and technical fields such as computers. Consequently, we found that Jess could return a low evaluation of "content" even for an essay that responded well to the essay prompt. When analyzing content, a mechanism is needed for automatically selecting a term-document cooccurrence matrix in accordance with the essay targeted for evaluation. This enable the users to avoid reverse-engineering that poor quality essays would produce perfect scores, because thresholds for detecting the outliers on rhetoric features may be varied.

Acknowledgements

We would like to extend their deep appreciation to Professor Eiji Muraki, currently of Tohoku University, Graduate School of Educational Informatics, Research Division, who, while resident at Educational Testing Service (ETS), was kind enough to arrange a visit for us during our survey of the e-rater system.

References

Bennet, R.E. and Bejar, I.I. 1998. Validity and automated scoring: It's not only the scoring, *Educational Measurement: Issues and Practice*. 17(4):9–17.

Bereiter, C. 2003. Foreword. In Shermis, M. and Burstein, J. eds. *Automated essay scoring: cross-disciplinary perspective*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Berry, M.W. 1992. Large scale singular value computations, *International Journal of Supercomputer Applications*. 6(1):13–49.

Burstein, J., Kukich, K., Wolff, S., Lu, C., Chodorow, M., Braden-Harder, L., and Harris, M.D. 1998. Automated Scoring Using A Hybrid Feature Identification Technique. *the Annual Meeting of the Association of Computational Linguistics*, Available online: www.ets.org/research/erater.html

Chase, C.I. 1986. Essay test scoring : interaction of relevant variables, *Journal of Educational Measurement*, 23(1):33–41.

Chase, C.I. 1979. The impact of achievement expectations and handwriting quality on scoring essay tests, *Journal of Educational Measurement*, 16(1):293–297.

Cooper, P.L. 1984. The assessment of writing ability: a review of research, *GRE Board Research Report*, GREB No.82-15R. Available online: www.gre.org/reswrit.html#TheAssessmentofWriting

Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(7):391–407.

Duff, I.S., Grimes, R.G. and Lewis, J.G. 1989. Sparse matrix test problem. *ACM Trans. Math. Software*, 15:1–14.

Elliot, S. 2003. IntelliMetric: From Here to Validity, 71–86. In Shermis, M. and Burstein, J. eds. *Automated essay scoring: A cross-disciplinary perspective*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Foltz, P.W., Laham, D. and Landauer, T.K. 1999. Automated Essay Scoring: Applications to Educational Technology. *EdMedia '99*.

Hughes, D.C., Keeling B. and Tuck, B.F. 1983. The effects of instructions to scorers intended to reduce context effects in essay scoring, *Educational and Psychological Measurement*, 43:1047–1050.

Ishioka, T. and Kameda, M. 1999. Document retrieval based on Words' cooccurrences — the algorithm and its applications (in Japanese), *Japanese Journal of Applied Statistics*, 28(2):107–121.

Jelinek, F. 1991. Up from trigrams! The struggle for improved Language models, *the European Conference on Speech Communication and Technology (EUROSPEECH-91)*, 1037–1040.

Knuth, D.E., Larrabee, T. and Roberts, P.M. 1988. *Mathematical Writing*, Stanford University Computer Science Department, Report Number: STAN-CS-88-1193.

Maekawa, M. 1995. *Scientific Analysis of Writing* (in Japanese), ISBN4-00-007953-0, Iwanami Shotton.

Marshall, J.C. and Powers, J.M. 1969. Writing neatness, composition errors and essay grades, *Journal of Educational Measurement*, 6(2):97–101.

Meyer, G. 1939. The choice of questions on essay examinations, *Journal of Educational Psychology*, 30(3):161–171.

Nagao, M.(ed.) 1996. *Natural Language Processing* (in Japanese), The Iwanami Software Science Series 15, ISBN 4-00-10355-5,

Noya, S.: 1997. *Logical Training* (in Japanese), Sangyo Tosho, ISBN 4-7828-0205-6.

Page, E.B., Poggio, J.P. and Keith, T.Z. 1997. Computer analysis of student essays: Finding trait differences in the student profile. *AERA/NCME Symposium on Grading Essays by Computer*.

Powers, D.E., Burstein, J.C., Chodorow, M., Fowles, M.E., and Kukich, K. 2000. Comparing the validity of automated and human essay scoring, GRE No. 98-08a. Princeton, NJ: Educational Testing Service.

Quirk, R., Greenbaum, S., Leech, G. and Svartvik, J. 1985. *A Comprehensive Grammar of the English Language*, Longman.

Rudner, L.M. and Liang, L. 2002. *National Council on Measurement in Education*, New Orleans, LA. Available online: http://ericae.net/betsy/papers/n2002e.pdf

Tweedie, F.J. and Baayen, R.H. 1998 How Variable May a Constant Be? Measures of Lexical Richness in Perspective, *Computers and the Humanities*, 32:323–352.

Watanabe, H., Taira, Y. and Inoue, T. 1988 An Analysis of Essay Examination Data (in Japanese), Research bulletin, Fuculty of Education, University of Tokyo, 28:143–164.

Yule, G.U. 1944. *The Statistical Study of Literary Vocabulary*, Cambridge University Press, Cambridge.

A Feedback-Augmented Method for Detecting Errors in the Writing of Learners of English

Ryo Nagata
Hyogo University of Teacher Education
6731494, Japan
rnagata@hyogo-u.ac.jp

Atsuo Kawai
Mie University
5148507, Japan
kawai@ai.info.mie-u.ac.jp

Koichiro Morihiro
Hyogo University of Teacher Education
6731494, Japan
mori@hyogo-u.ac.jp

Naoki Isu
Mie University
5148507, Japan
isu@ai.info.mie-u.ac.jp

Abstract

This paper proposes a method for detecting errors in article usage and singular plural usage based on the mass count distinction. First, it learns decision lists from training data generated automatically to distinguish mass and count nouns. Then, in order to improve its performance, it is augmented by feedback that is obtained from the writing of learners. Finally, it detects errors by applying rules to the mass count distinction. Experiments show that it achieves a recall of 0.71 and a precision of 0.72 and outperforms other methods used for comparison when augmented by feedback.

1 Introduction

Although several researchers (Kawai et al., 1984; McCoy et al., 1996; Schneider and McCoy, 1998; Tschichold et al., 1997) have shown that rule-based methods are effective to detecting grammatical errors in the writing of learners of English, it has been pointed out that it is hard to write rules for detecting errors concerning the articles and singular plural usage. To be precise, it is hard to write rules for distinguishing mass and count nouns which are particularly important in detecting these errors (Kawai et al., 1984). The major reason for this is that whether a noun is a mass noun or a count noun greatly depends on its meaning or its surrounding context (refer to Allan (1980) and Bond (2005) for details of the mass count distinction).

The above errors are very common among Japanese learners of English (Kawai et al., 1984; Izumi et al., 2003). This is perhaps because the Japanese language does not have a mass count distinction system similar to that of English. Thus, it is favorable for error detection systems aiming at Japanese learners to be capable of detecting these errors. In other words, such systems need to somehow distinguish mass and count nouns.

This paper proposes a method for distinguishing mass and count nouns in context to complement the conventional rules for detecting grammatical errors. In this method, first, training data, which consist of instances of mass and count nouns, are automatically generated from a corpus. Then, decision lists for distinguishing mass and count nouns are learned from the training data. Finally, the decision lists are used with the conventional rules to detect the target errors.

The proposed method requires a corpus to learn decision lists for distinguishing mass and count nouns. General corpora such as newspaper articles can be used for the purpose. However, a drawback to it is that there are differences in character between general corpora and the writing of non-native learners of English (Granger, 1998; Chodorow and Leacock, 2000). For instance, Chodorow and Leacock (2000) point out that the word *concentrate* is usually used as a noun in a general corpus whereas it is a verb 91% of the time in essays written by non-native learners of English. Consequently, the differences affect the performance of the proposed method.

In order to reduce the drawback, the proposed method is augmented by feedback; it takes as feedback learners' essays whose errors are corrected by a teacher of English (hereafter, referred to as the feedback corpus). In essence, the feedback corpus could be added to a general corpus to generate training data. Or, ideally training data could be generated only from the feedback corpus just as

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 241–248,
Sydney, July 2006. ©2006 Association for Computational Linguistics

from a general corpus. However, this causes a serious problem in practice since the size of the feedback corpus is normally far smaller than that of a general corpus. To make it practical, this paper discusses the problem and explores its solution.

The rest of this paper is structured as follows. Section 2 describes the method for detecting the target errors based on the mass count distinction. Section 3 explains how the method is augmented by feedback. Section 4 discusses experiments conducted to evaluate the proposed method.

2 Method for detecting the target errors

2.1 Generating training data

First, instances of the target noun that head their noun phrase (NP) are collected from a corpus with their surrounding words. This can be simply done by an existing chunker or parser.

Then, the collected instances are tagged with mass or count by the following tagging rules. For example, the underlined *chicken*:

... are a lot of *chicken*s in the roost ...

is tagged as

... are a lot of *chicken*s/count in the roost ...

because it is in plural form.

We have made tagging rules based on linguistic knowledge (Huddleston and Pullum, 2002). Figure 1 and Table 1 represent the tagging rules. Figure 1 shows the framework of the tagging rules. Each node in Figure 1 represents a question applied to the instance in question. For example, the root node reads “Is the instance in question plural?”. Each leaf represents a result of the classification. For example, if the answer is *yes* at the root node, the instance in question is tagged with count. Otherwise, the question at the lower node is applied and so on. The tagging rules do not classify instances as mass or count in some cases. These unclassified instances are tagged with the symbol “?”. Unfortunately, they cannot readily be included in training data. For simplicity of implementation, they are excluded from training data[1].

Note that the tagging rules can be used only for generating training data. They cannot be used to distinguish mass and count nouns in the writing of learners of English for the purpose of detecting the target errors since they are based on the articles and the distinction between singular and plural.

Finally, the tagged instances are stored in a file with their surrounding words. Each line of it consists of one of the tagged instances and its surrounding words as in the above *chicken* example.

2.2 Learning Decision Lists

In the proposed method, decision lists are used for distinguishing mass and count nouns. One of the reasons for the use of decision lists is that they have been shown to be effective to the word sense disambiguation task and the mass count distinction is highly related to word sense as we will see in this section. Another reason is that rules for distinguishing mass and count nouns are observable in decision lists, which helps understand and improve the proposed method.

A decision list consists of a set of rules. Each rule matches the template as follows:

$$\text{If } \textit{a condition is true}, \text{ then } \textit{a decision}. \qquad (1)$$

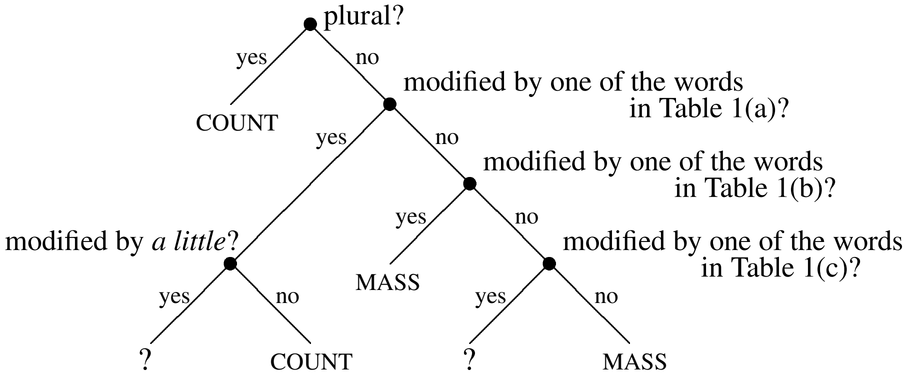
To define the template in the proposed method, let us have a look at the following two examples:

1. I read the *paper*.
2. The *paper* is made of hemp pulp.

The underlined *paper*s in both sentences cannot simply be classified as mass or count by the tagging rules presented in Section 2.1 because both are singular and modified by the definite article. Nevertheless, we can tell that the former is a count noun and the latter is a mass noun from the contexts. This suggests that the mass count distinction is often determined by words surrounding the target noun. In example 1, we can tell that the *paper* refers to something that can be read such as a newspaper or a scientific paper from *read*, and therefore it is a count noun. Likewise, in example 2, we can tell that the *paper* refers to a certain substance from *made* and *pulp*, and therefore it is a mass noun.

Taking this observation into account, we define the template based on words surrounding the target noun. To formalize the template, we will use a random variable MC that takes either $mass$ or *count* to denote that the target noun is a mass noun or a count noun, respectively. We will also use w and C to denote a word and a certain context around the target noun, respectively. We define

[1] According to experiments we have conducted, approximately 30% of instances are tagged with “?” on average. It is highly possible that performance of the proposed method will improve if these instances are included in the training data.



plural?
yes
no
COUNT
modified by one of the words in Table 1(a)?
yes
no
modified by one of the words in Table 1(b)?
modified by *a little*?
yes
no
MASS
modified by one of the words in Table 1(c)?
yes
no
yes
no
?
COUNT
?
MASS

Figure 1: Framework of the tagging rules

Table 1: Words used in the tagging rules

(a)	(b)	(c)
the indefinite article	much	*the definite article*
another	less	*demonstrative adjectives*
one	enough	*possessive adjectives*
each	sufficient	*interrogative adjectives*
–	–	*quantifiers*
–	–	*'s genitives*

three types of C: np, $-k$, and $+k$ that denote the contexts consisting of the noun phrase that the target noun heads, k words to the left of the noun phrase, and k words to its right, respectively. Then the template is formalized by:

If word w appears in context C of the target noun, then it is distinguished as MC.

Hereafter, to keep the notation simple, it will be abbreviated to

$$w_C \rightarrow MC. \tag{2}$$

Now rules that match the template can be obtained from the training data. All we need to do is to collect words in C from the training data. Here, the words in Table 1 are excluded. Also, function words (except prepositions), cardinal and quasi-cardinal numerals, and the target noun are excluded. All words are reduced to their morphological stem and converted entirely to lower case when collected. For example, the following tagged instance:

She ate fried <u>*chicken*/mass</u> for dinner.

would give a set of rules that match the template:

$$eat_{-3} \rightarrow mass$$
$$fry_{np} \rightarrow mass$$
$$for_{+3} \rightarrow mass$$
$$dinner_{+3} \rightarrow mass$$

for the target noun *chicken* when $k = 3$.

In addition, a default rule is defined. It is based on the target noun itself and used when no other applicable rules are found in the decision list for the target noun. It is defined by

$$t \rightarrow MC_{\text{major}} \tag{3}$$

where t and MC_{major} denote the target noun and the majority of MC in the training data, respectively. Equation (3) reads "If the target noun appears, then it is distinguished by the majority".

The log-likelihood ratio (Yarowsky, 1995) decides in which order rules are applied to the target noun in novel context. It is defined by[2]

$$log \frac{p(MC|w_C)}{p(\overline{MC}|w_C)} \tag{4}$$

where $\overline{MC}$ is the exclusive event of MC and $p(MC|w_C)$ is the probability that the target noun is used as MC when w appears in the context C.

It is important to exercise some care in estimating $p(MC|w_C)$. In principle, we could simply

[2] For the default rule, the log-likelihood ratio is defined by replacing w_C and MC with t and MC_{major}, respectively.

count the number of times that w appears in the context C of the target noun used as MC in the training data. However, this estimate can be unreliable, when w does not appear often in the context. To solve this problem, using a smoothing parameter α (Yarowsky, 1996), $p(MC|w_C)$ is estimated by[3]

$$p(MC|w_C) = \frac{f(w_C, MC) + \alpha}{f(w_C) + m\alpha} \tag{5}$$

where $f(w_C)$ and $f(w_C, MC)$ are occurrences of w appearing in C and those in C of the target noun used as MC, respectively. The constant m is the number of possible classes, that is, $m = 2$ ($mass$ or $count$) in our case, and introduced to satisfy $p(MC|w_C) + p(\overline{MC}|w_C) = 1$. In this paper, α is set to 1.

Rules in a decision list are sorted in descending order by the log-likelihood ratio. They are tested on the target noun in novel context in this order. Rules sorted below the default rule are discarded[4] because they are never used as we will see in Section 2.3.

Table 2 shows part of a decision list for the target noun *chicken* that was learned from a subset of the BNC (British National Corpus) (Burnard, 1995). Note that the rules are divided into two columns for the purpose of illustration in Table 2; in practice, they are merged into one.

Table 2: Rules in a decision list

Mass		Count	
w_C	LLR	w_C	LLR
$piece_{-3}$	1.49	$count_{-3}$	1.49
$fish_{-3}$	1.28	$peck_{+3}$	1.32
$dish_{-3}$	1.23	pig_{np}	1.23
$skin_{+3}$	1.23	run_{-3}	1.23
$serve_{+3}$	1.18	egg_{np}	1.18

target noun: *chicken*, $k = 3$
LLR (Log-Likelihood Ratio)

On one hand, we associate the words in the left half with food or cooking. On the other hand, we associate those in the right half with animals or birds. From this observation, we can say that *chicken* in the sense of an animal or a bird is a count noun but a mass noun when referring to food or cooking, which agrees with the knowledge presented in previous work (Ostler and Atkins, 1991).

2.3 Distinguishing mass and count nouns

To distinguish the target noun in novel context, each rule in the decision list is tested on it in the sorted order until the first applicable one is found. It is distinguished according to the first applicable one. Ties are broken by the rules below.

It should be noted that rules sorted below the default rule are never used because the default rule is always applicable to the target noun. This is the reason why rules sorted below the default rule are discarded as mentioned in Section 2.2.

2.4 Detecting the target errors

The target errors are detected by the following three steps. Rules in each step are examined on each target noun in the target text.

In the first step, any mass noun in plural form is detected as an error[5]. If an error is detected in this step, the rest of the steps are not applied.

In the second step, errors are detected by the rules described in Table 3. The symbol "×" in Table 3 denotes that the combination of the corresponding row and column is erroneous. For example, the fifth row denotes that singular and plural count nouns modified by *much* are erroneous. The symbol "–" denotes that no error can be detected by the table. If one of the rules in Table 3 is applied to the target noun, the third step is not applied.

In the third step, errors are detected by the rules described in Table 4. The symbols "×" and "–" are the same as in Table 3.

In addition, the indefinite article that modifies other than the head noun is judged to be erroneous

Table 3: Detection rules (i)

	Count		Mass
Pattern	Sing.	Pl.	Sing.
{another, each, one}	–	×	×
{all, enough, sufficient}	×	–	–
{much}	×	×	–
{that, this}	–	×	–
{few, many, several}	×	–	×
{these, those}	×	–	×
{various, numerous}	×	–	×
cardinal numbers exc. one	×	–	×

[3] The probability for the default rule is estimated just as the log-likelihood ratio for the default rule above.

[4] It depends on the target noun how many rules are discarded.

[5] Mass nouns can be used in plural in some cases. However, they are rare especially in the writing of learners of English.

Table 4: Detection rules (ii)

	Singular			Plural		
	a/an	the	ϕ	a/an	the	ϕ
Mass	×	–	–	×	×	×
Count	–	–	×	×	–	–

(e.g., *an expensive). Likewise, the definite article that modifies other than the head noun or adjective is judged to be erroneous (e.g., *the them). Also, we have made exceptions to the rules. The following combinations are excluded from the detection in the second and third steps: head nouns modified by interrogative adjectives (e.g., what), possessive adjectives (e.g., my), 's genitives, "some", "any", or "no".

3 Feedback-augmented method

As mentioned in Section 1, the proposed method takes the feedback corpus[6] as feedback to improve its performance. In essence, decision lists could be learned from a corpus consisting of a general corpus and the feedback corpus. However, since the size of the feedback corpus is normally far smaller than that of general corpora, so is the effect of the feedback corpus on $p(MC|w_C)$. This means that the feedback corpus hardly has effect on the performance.

Instead, $p(MC|w_C)$ can be estimated by interpolating the probabilities estimated from the feedback corpus and the general corpus according to confidences of their estimates. It is favorable that the interpolated probability approaches to the probability estimated from the feedback corpus as its confidence increases; the more confident its estimate is, the more effect it has on the interpolated probability. Here, confidence c of ratio p is measured by the reciprocal of variance of the ratio (Tanaka, 1977). Variance is calculated by

$$\frac{p(1-p)}{n} \tag{6}$$

where n denotes the number of samples used for calculating the ratio. Therefore, confidence of the estimate of the conditional probability used in the proposed method is measured by

$$c = \frac{f(w_C)}{p(MC|w_C)(1-p(MC|w_C))}. \tag{7}$$

To formalize the interpolated probability, we will use the symbols p_{fb}, p_g, c_{fb}, and c_g to denote the conditional probabilities estimated from the feedback corpus and the general corpus, and their confidences, respectively. Then, the interpolated probability p_i is estimated by[7]

$$p_i = \begin{cases} p_g + \frac{c_{fb}}{c_g}(p_{fb} - p_g), & c_{fb} < c_g \\ p_{fb}, & c_{fb} \geq c_g \end{cases}. \tag{8}$$

In Equation (8), the effect of p_{fb} on p_i becomes large as its confidence increases. It should also be noted that when its confidence exceeds that of p_g, the general corpus is no longer used in the interpolated probability.

A problem that arises in Equation (8) is that p_{fb} hardly has effect on p_i when a much larger general corpus is used than the feedback corpus even if p_{fb} is estimated with a sufficient confidence. For example, p_{fb} estimated from 100 samples, which are a relatively large number for estimating a probability, hardly has effect on p_i when p_g is estimated from 10000 samples; roughly, p_{fb} has a $1/100$ effect of p_g on p_i.

One way to prevent this is to limit the effect of c_g to some extent. It can be realized by taking the log of c_g in Equation (8). That is, the interpolated probability is estimated by

$$p_i = \begin{cases} p_g + \frac{c_{fb}}{\log c_g}(p_{fb} - p_g), & c_{fb} < \log c_g \\ p_{fb}, & c_{fb} \geq \log c_g \end{cases}. \tag{9}$$

It is arguable what base of the log should be used. In this paper, it is set to 2 so that the effect of p_g on the interpolated probability becomes large when the confidence of the estimate of the conditional probability estimated from the feedback corpus is small (that is, when there is little data in the feedback corpus for the estimate)[8].

In summary, Equation (9) interpolates between the conditional probabilities estimated from the feedback corpus and the general corpus in the feedback-augmented method. The interpolated probability is then used to calculate the log-likelihood ratio. Doing so, the proposed method takes the feedback corpus as feedback to improve its performance.

[6] The feedback corpus refers to learners' essays whose errors are corrected as mentioned in Section 1.

[7] In general, the interpolated probability needs to be normalized to satisfy $\sum p_i = 1$. In our case, however, it is always satisfied without normalization since $p_{fb}(MC|w_C) + p_{fb}(\overline{MC}|w_C) = 1$ and $p_g(MC|w_C) + p_g(\overline{MC}|w_C) = 1$ are satisfied.

[8] We tested several bases in the experiments and found there were little difference in performance between them.

4 Experiments

4.1 Experimental Conditions

A set of essays[9] written by Japanese learners of English was used as the target essays in the experiments. It consisted of 47 essays (3180 words) on the topic *traveling*. A native speaker of English who was a professional rewriter of English recognized 105 target errors in it.

The written part of the British National Corpus (BNC) (Burnard, 1995) was used to learn decision lists. Sentences the OAK system[10], which was used to extract NPs from the corpus, failed to analyze were excluded. After these operations, the size of the corpus approximately amounted to 80 million words. Hereafter, the corpus will be referred to as the BNC.

As another corpus, the English concept explication in the EDR English-Japanese Bilingual dictionary and the EDR corpus (1993) were used; it will be referred to as the EDR corpus, hereafter. Its size amounted to about 3 million words.

Performance of the proposed method was evaluated by recall and precision. Recall is defined by

$$\frac{\text{No. of target errors detected correctly}}{\text{No. of target errors in the target essays}}. \quad (10)$$

Precision is defined by

$$\frac{\text{No. of target errors detected correctly}}{\text{No. of detected errors}}. \quad (11)$$

4.2 Experimental Procedures

First, decision lists for each target noun in the target essays were learned from the BNC[11]. To extract noun phrases and their head nouns, the OAK system was used. An optimal value for k (window size of context) was estimated as follows. For 25 nouns shown in (Huddleston and Pullum, 2002) as examples of nouns used as both mass and count nouns, accuracy on the BNC was calculated using ten-fold cross validation. As a result of setting small ($k = 3$), medium ($k = 10$), and large ($k = 50$) window sizes, it turned out that $k = 3$ maximized the average accuracy. Following this result, $k = 3$ was selected in the experiments.

Second, the target nouns were distinguished whether they were mass or count by the learned decision lists, and then the target errors were detected by applying the detection rules to the mass count distinction. As a preprocessing, spelling errors were corrected using a spell checker. The results of the detection were compared to those done by the native-speaker of English. From the comparison, recall and precision were calculated.

Then, the feedback-augmented method was evaluated on the same target essays. Each target essay in turn was left out, and all the remaining target essays were used as a feedback corpus. The target errors in the left-out essay were detected using the feedback-augmented method. The results of all 47 detections were integrated into one to calculate overall performance. This way of feedback can be regarded as that one uses revised essays previously written in a class to detect errors in essays on the same topic written in other classes.

Finally, the above two methods were compared with their seven variants shown in Table 5. "DL" in Table 5 refers to the nine decision list based methods (the above two methods and their seven variants). The words in brackets denote the corpora used to learn decision lists; the symbol "+FB" means that the feedback corpus was simply added to the general corpus. The subscripts fb_1 and fb_2 indicate that the feedback was done by using Equation (8) and Equation (9), respectively.

In addition to the seven variants, two kinds of earlier method were used for comparison. One was one (Kawai et al., 1984) of the rule-based methods. It judges singular head nouns with no determiner to be erroneous since missing articles are most common in the writing of Japanese learners of English. In the experiments, this was implemented by treating all nouns as count nouns and applying the same detection rules as in the proposed method to the countability.

The other was a web-based method (Lapata and Keller, 2005)[12] for generating articles. It retrieves web counts for queries consisting of two words preceding the NP that the target noun head, one of the articles ($\{$a/an, the, $\phi\}$), and the core NP to generate articles. All queries are performed as exact matches using quotation marks and submitted to the Google search engine in lower case. For example, in the case of "*She is good student.", it retrieves web counts for "she is a good student",

[9] http://www.eng.ritsumei.ac.jp/lcorpus/.

[10] OAK System Homepage: http://nlp.cs.nyu.edu/oak/.

[11] If no instance of the target noun is found in the general corpora (and also in the feedback corpus in case of the feedback-augmented method), the target noun is ignored in the error detection procedure.

[12] There are other statistical methods that can be used for comparison including Lee (2004) and Minnen (2000). Lapata and Keller (2005) report that the web-based method is the best performing article generation method.

"she is the good student", and "she is good student". Then, it generates the article that maximizes the web counts. We extended it to make it capable of detecting our target errors. First, the singular/plural distinction was taken into account in the queries (e.g., "she is a good students", "she is the good students", and "she is good students" in addition to the above three queries). The one(s) that maximized the web counts was judged to be correct; the rest were judged to be erroneous. Second, if determiners other than the articles modify head nouns, only the distinction between singular and plural was taken into account (e.g., "he has some book" vs "he has some books"). In the case of "much/many", the target noun in singular form modified by "much" and that in plural form modified by "many" were compared (e.g., "he has much furniture" vs "he has many furnitures). Finally, some rules were used to detect literal errors. For example, plural head nouns modified by "this" were judged to be erroneous.

4.3 Experimental Results and Discussion

Table 5 shows the experimental results. "Rule-based" and "Web-based" in Table 5 refer to the rule-based method and the web-based method, respectively. The other symbols are as already explained in Section 4.2.

As we can see from Table 5, all the decision list based methods outperform the earlier methods. The rule-based method treated all nouns as count nouns, and thus it did not work well at all on mass nouns. This caused a lot of false-positives and false-negatives. The web-based method suffered a lot from other errors than the target errors since it implicitly assumed that there were no errors except the target errors. Contrary to this assumption, not only did the target essays contain the target errors but also other errors since they were written by Japanese learners of English. This indicate that the queries often contained the other errors when web counts were retrieved. These errors made the web counts useless, and thus it did not perform well. By contrast, the decision list based methods did because they distinguished mass and count nouns by one of the words around the target noun that was most likely to be effective according to the log-likelihood ratio[13]; the best performing decision list based method (DL$_{fb2}$ (EDR)) is significantly superior to the best performing[14] non-decision list based method (Web-based) in both recall and precision at the 99% confidence level.

Table 5: Experimental results

Method	Recall	Precision
DL (BNC)	0.66	0.65
DL (BNC+FB)	0.66	0.65
DL$_{fb1}$ (BNC)	0.66	0.65
DL$_{fb2}$ (BNC)	0.69	0.70
DL (EDR)	0.70	0.68
DL (EDR+FB)	0.71	0.69
DL$_{fb1}$ (EDR)	0.71	0.70
DL$_{fb2}$ (EDR)	0.71	0.72
DL (FB)	0.43	0.76
Rule-based	0.59	0.39
Web-based	0.49	0.53

Table 5 also shows that the feedback-augmented methods benefit from feedback. Only an exception is "DL$_{fb1}$ (BNC)". The reason is that the size of BNC is far larger than that of the feedback corpus and thus it did not affect the performance. This also explains that simply adding the feedback corpus to the general corpus achieved little or no improvement as "DL (EDR+FB)" and "DL (BNC+FB)" show. Unlike these, both "DL$_{fb2}$ (BNC)" and "DL$_{fb2}$ (EDR)" benefit from feedback since the effect of the general corpus is limited to some extent by the log function in Equation (9). Because of this, both benefit from feedback despite the differences in size between the feedback corpus and the general corpus.

Although the experimental results have shown that the feedback-augmented method is effective to detecting the target errors in the writing of Japanese learners of English, even the best performing method (DL$_{fb2}$ (EDR)) made 30 false-negatives and 29 false-positives. About 70% of the false-negatives were errors that required other sources of information than the mass count distinction to be detected. For example, extra definite articles (e.g., *the traveling) cannot be detected even if the correct mass count distinction is given. Thus, only a little improvement is expected in recall however much feedback corpus data become available. On the other hand, most of the

[13] Indeed, words around the target noun were effective. The default rules were used about 60% and 30% of the time in "DL (EDR)" and "DL (BNC)", respectively; when only the default rules were used, "DL (EDR)" ("DL (BNC)") achieved 0.66 (0.56) in recall and 0.58 (0.53) in precision.

[14] "Best performing" here means best performing in terms of F-measure.

false-positives were due to the decision lists themselves. Considering this, it is highly possible that precision will improve as the size of the feedback corpus increases.

5 Conclusions

This paper has proposed a feedback-augmented method for distinguishing mass and count nouns to complement the conventional rules for detecting grammatical errors. The experiments have shown that the proposed method detected 71% of the target errors in the writing of Japanese learners of English with a precision of 72% when it was augmented by feedback. From the results, we conclude that the feedback-augmented method is effective to detecting errors concerning the articles and singular plural usage in the writing of Japanese learners of English.

Although it is not taken into account in this paper, the feedback corpus contains further useful information. For example, we can obtain training data consisting of instances of errors by comparing the feedback corpus with its original corpus. Also, comparing it with the results of detection, we can know performance of each rule used in the detection, which make it possible to increase or decrease their log-likelihood ratios according to their performance. We will investigate how to exploit these sources of information in future work.

Acknowledgments

The authors would like to thank Sekine Satoshi who has developed the OAK System. The authors also would like to thank three anonymous reviewers for their useful comments on this paper.

References

K. Allan. 1980. Nouns and countability. *J. Linguistic Society of America*, 56(3):541–567.

F. Bond. 2005. *Translating the Untranslatable*. CSLI publications, Stanford.

L. Burnard. 1995. *Users Reference Guide for the British National Corpus. version 1.0*. Oxford University Computing Services, Oxford.

M. Chodorow and C. Leacock. 2000. An unsupervised method for detecting grammatical errors. In *Proc. of 1st Meeting of the North America Chapter of ACL*, pages 140–147.

Japan electronic dictionary research institute ltd. 1993. *EDR electronic dictionary specifications guide*. Japan electronic dictionary research institute ltd, Tokyo.

S. Granger. 1998. Prefabricated patterns in advanced EFL writing: collocations and formulae. In A. P. Cowie, editor, *Phraseology: theory, analysis, and applications*, pages 145–160. Clarendon Press.

R. Huddleston and G.K. Pullum. 2002. *The Cambridge Grammar of the English Language*. Cambridge University Press, Cambridge.

E. Izumi, K. Uchimoto, T. Saiga, T. Supnithi, and H. Isahara. 2003. Automatic error detection in the Japanese learners' English spoken data. In *Proc. of 41st Annual Meeting of ACL*, pages 145–148.

A. Kawai, K. Sugihara, and N. Sugie. 1984. ASPEC-I: An error detection system for English composition. *IPSJ Journal (in Japanese)*, 25(6):1072–1079.

M. Lapata and F. Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1):1–31.

J. Lee. 2004. Automatic article restoration. In *Proc. of the Human Language Technology Conference of the North American Chapter of ACL*, pages 31–36.

K.F. McCoy, C.A. Pennington, and L.Z. Suri. 1996. English error correction: A syntactic user model based on principled "mal-rule" scoring. In *Proc. of 5th International Conference on User Modeling*, pages 69–66.

G. Minnen, F. Bond, and A. Copestake. 2000. Memory-based learning for article generation. In *Proc. of CoNLL-2000 and LLL-2000 workshop*, pages 43–48.

N. Ostler and B.T.S Atkins. 1991. Predictable meaning shift: Some linguistic properties of lexical implication rules. In *Proc. of 1st SIGLEX Workshop on Lexical Semantics and Knowledge Representation*, pages 87–100.

D. Schneider and K.F. McCoy. 1998. Recognizing syntactic errors in the writing of second language learners. In *Proc. of 17th International Conference on Computational Linguistics*, pages 1198–1205.

Y. Tanaka. 1977. *Psychological methods (in Japanese)*. University of Tokyo Press.

C. Tschichold, F. Bodmer, E. Cornu, F. Grosjean, L. Grosjean, N. Kübler, N. Léwy, and C. Tschumi. 1997. Developing a new grammar checker for English as a second language. In *Proc. of the From Research to Commercial Applications Workshop*, pages 7–12.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of 33rd Annual Meeting of ACL*, pages 189–196.

D. Yarowsky. 1996. *Homograph Disambiguation in Speech Synthesis*. Springer-Verlag.

Correcting ESL Errors Using Phrasal SMT Techniques

Chris Brockett, William B. Dolan, and Michael Gamon

Natural Language Processing Group

Microsoft Research

One Microsoft Way, Redmond, WA 98005, USA

{chrisbkt,billdol,mgamon}@microsoft.com

Abstract

This paper presents a pilot study of the use of phrasal Statistical Machine Translation (SMT) techniques to identify and correct writing errors made by learners of English as a Second Language (ESL). Using examples of mass noun errors found in the *Chinese Learner Error Corpus (CLEC)* to guide creation of an engineered training set, we show that application of the SMT paradigm can capture errors not well addressed by widely-used proofing tools designed for native speakers. Our system was able to correct 61.81% of mistakes in a set of naturally-occurring examples of mass noun errors found on the World Wide Web, suggesting that efforts to collect alignable corpora of pre- and post-editing ESL writing samples offer can enable the development of SMT-based writing assistance tools capable of repairing many of the complex syntactic and lexical problems found in the writing of ESL learners.

1 Introduction

Every day, in schools, universities and businesses around the world, in email and on blogs and websites, people create texts in languages that are not their own, most notably English. Yet, for writers of English as a Second Language (ESL), useful editorial assistance geared to their needs is surprisingly hard to come by. Grammar checkers such as that provided in Microsoft Word have been designed primarily with native speakers in mind. Moreover, despite growing demand for ESL proofing tools, there has been remarkably little progress in this area over the last decade. Research into computer feedback for ESL writers remains largely focused on small-scale pedagogical systems implemented within the framework of CALL (Computer Aided Language Learning) (Reuer 2003; Vandeventer Faltin, 2003), while commercial ESL grammar checkers remain brittle and difficult to customize to meet the needs of ESL writers of different first-language (L1) backgrounds and skill levels.

Some researchers have begun to apply statistical techniques to identify learner errors in the context of essay evaluation (Chodorow & Leacock, 2000; Lonsdale & Strong-Krause, 2003), to detect non-native text (Tomokiyo & Jones, 2001), and to support lexical selection by ESL learners through first-language translation (Liu et al., 2000). However, none of this work appears to directly address the more general problem of how to robustly provide feedback to ESL writers—and for that matter non-native writers in any second language—in a way that is easily tailored to different L1 backgrounds and second-language (L2) skill levels.

In this paper, we show that a noisy channel model instantiated within the paradigm of Statistical Machine Translation (SMT) (Brown et al., 1993) can successfully provide editorial assistance for non-native writers. In particular, the SMT approach provides a natural mechanism for suggesting a correction, rather than simply stranding the user with a flag indicating that the text contains an error. Section 2 further motivates the approach and briefly describes our SMT system. Section 3 discusses the data used in our experiment, which is aimed at repairing a common type of ESL error that is not well-handled by current grammar checking technology: mass/count noun confusions. Section 4 presents experimental results, along with an analysis of errors produced by the system. Finally we present discussion and some future directions for investigation.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 249–256,
Sydney, July 2006. ©2006 Association for Computational Linguistics

2 Error Correction as SMT

2.1 Beyond Grammar Checking

A major difficulty for ESL proofing is that errors of grammar, lexical choice, idiomaticity, and style rarely occur in isolation. Instead, any given sentence produced by an ESL learner may involve a complex combination of all these error types. It is difficult enough to design a proofing tool that can reliably correct individual errors; the simultaneous combination of multiple errors is beyond the capabilities of current proofing tools designed for native speakers. Consider the following example, written by a Korean speaker and found on the World Wide Web, which involves the misapplication of countability to a mass noun:

> `And I knew many informations about Christmas while I was preparing this article.`

The grammar and spelling checkers in Microsoft Word 2003 correctly suggest `many` → `much` and `informations` → `information`. Accepting these proposed changes, however, does not render the sentence entirely native-like. Substituting the word `much` for `many` leaves the sentence stilted in a way that is probably undetectable to an inexperienced non-native speaker, while the use of the word `knew` represents a lexical selection error that falls well outside the scope of conventional proofing tools. A better rewrite might be:

> `And I learned a lot of information about Christmas while I was preparing this article.`

or, even more colloquially:

> `And I learned a lot about Christmas while I was preparing this article`

Repairing the error in the original sentence, then, is not a simple matter of fixing an agreement marker or substituting one determiner for another. Instead, wholesale replacement of the phrase `knew many informations` with the phrase `learned a lot` is needed to produce idiomatic-sounding output. Seen in these terms, the process of mapping from a raw, ESL-authored string to its colloquial equivalent looks remarkably like translation. Our goal is to show that providing editorial assistance for writers should be viewed as a special case of translation. Rather than learning how strings in one language map to strings in another, however, "translation" now involves learning how systematic patterns of errors in ESL learners' English map to corresponding patterns in native English

2.2 A Noisy Channel Model of ESL Errors

If ESL error correction is seen as a translation task, the task can be treated as an SMT problem using the noisy channel model of (Brown et al., 1993): here the L2 sentence produced by the learner can be regarded as having been corrupted by noise in the form of interference from his or her L1 model and incomplete language models internalized during language learning. The task, then, is to reconstruct a corresponding valid sentence of L2 (target). Accordingly, we can seek to probabilistically identify the optimal correct target sentence(s) *T** of an ESL input sentence *S* by applying the familiar SMT formula:

$$T^* = \arg\max_T \{\mathrm{P}(T \mid S)\}$$
$$= \arg\max_T \{\mathrm{P}(S \mid T)\,\mathrm{P}(T)\}$$

In the context of this model, editorial assistance becomes a matter of identifying those segments of the optimal target sentence or sentences that differ from the writer's original input and displaying them to the user. In practice, the patterns of errors produced by ESL writers of specific L1 backgrounds can be captured in the channel model as an emergent property of training data consisting ESL sentences aligned with their corrected edited counterparts. The highest frequency errors and infelicities should emerge as targets for replacement, while lesser frequency or idiosyncratic problems will in general not surface as false flags.

2.3 Implementation

In this paper, we explore the use of a large-scale production statistical machine translation system to correct a class of ESL errors. A detailed description of the system can be found in (Menezes & Quirk 2005) and (Quirk et al., 2005). In keeping with current best practices in SMT, our system is a phrasal machine translation system that attempts to learn mappings between "phrases" (which may not correspond to linguistic units) rather than individual words. What distinguishes this system from other phrasal SMT systems is that rather than aligning simple sequences of words, it maps small phrasal "treelets" generated by a dependency parse to corresponding strings in the target. This "Tree-To-String" model holds promise in that it allows us to potentially benefit from being able to access a certain amount of structural information during translation, without necessarily being completely tied to the need for a fully-well-formed linguistic analysis of the input—an important consideration when it is sought to handle ungrammatical or otherwise ill-formed ESL input, but also simultaneously to capture relationships not involving contiguous strings, for example determiner-noun relations.

In our pilot study, this system was employed without modification to the system architecture. The sole adjustment made was to have both Source (erroneous) and Target (correct) sentences tokenized using an English language tokenizer. N-best results for phrasal alignment and ordering models in the decoder were optimized by lambda training via Maximum Bleu, along the lines described in (Och, 2003).

3 Data Development

3.1 Identifying Mass Nouns

In this paper, we focus on countability errors associated with mass nouns. This class of errors (involving nouns that cannot be counted, such as `information, pollution`, and `homework`) is characteristically encountered in ESL writing by native speakers of several East Asian languages (Dalgish, 1983; Hua & Lee, 2004).[1] We began by identifying a list of English nouns that are frequently involved in mass/count errors in by writing by Chinese ESL learners, by taking the intersection of words which:

- occurred in either the *Longman Dictionary of Contemporary English* or the *American Heritage Dictionary* with a mass sense
- were involved in n ≥ 2 mass/count errors in the *Chinese Learner English Corpus CLEC* (Gui and Yang, 2003), either tagged as a mass noun error or else with an adjacent tag indicating an article error.[2]

This procedure yielded a list of 14 words: `knowledge, food, homework, fruit, news, color, nutrition, equipment, paper, advice, haste, information, lunch, and tea.`[3] Countability errors involving these words are scattered across 46 sentences in the CLEC corpus.

For a baseline representing the level of writing assistance currently available to the average ESL writer, we submitted these sentences to the proofing tools in Microsoft Word 2003. The spelling and grammar checkers correctly identified 21 of the 46 relevant errors, proposed one incorrect substitution (`a few advice` → `a few advices`), and failed to flag the remaining 25 errors. With one exception, the proofing tools successfully detected as spelling errors incorrect plurals on lexical items that permit only mass noun interpretations (e.g., `informations`), but ignored plural forms like `fruits` and `papers` even when contextually inappropriate. The proofing tools in Word 2003 also detected singular determiner mismatches with obligatory plural forms (e.g. `a news).`

3.2 Training Data

The errors identified in these sentences provided an informal template for engineering the data in our training set, which was created by manipulating well-formed, edited English sentences. Raw data came from a corpus of ~484.6 million words of Reuters Limited newswire articles, released between 1995 and 1998, combined with a ~7,175,000-word collection of articles from multiple news sources from 2004-2005. The resulting dataset was large enough to ensure that all targeted forms occurred with some frequency.

From this dataset we culled about 346,000 sentences containing examples of the 14 targeted words. We then used hand-constructed regular expressions to convert these sentences into mostly-ungrammatical strings that exhibited characteristics of the *CLEC* data, for example:

- `much` → `many: much advice` → `many advice`
- `some` → `a/an: some advice` → `an advice`
- conversions to plurals`: much good advice` → `many good advices`

[1] These constructions are also problematic for handcrafted MT systems (Bond et al., 1994).

[2] *CLEC* tagging is not comprehensive; some common mass noun errors (e.g., `make a good progress`) are not tagged in this corpus.

[3] Terms that also had a function word sense, such as `will`, were eliminated for this experiment.

Data Size	Whole	Partial	Correctly Left	New Error	Missed	Word Order Error
45K	55.28	0.81	8.13	12.20	21.14	1.63
30K	36.59	4.07	7.32	16.26	32.52	3.25
15K	47.15	2.44	5.69	11.38	29.27	4.07
cf. Word	29.27	0.81	10.57	1.63	57.72	N/A

Table 1. Replacement percentages (per sentence basis) using different training data sets

- deletion of counters: piece(s)/ item(s)/sheet(s) of)
- insertion of determiners

These were produced in multiple combinations for broad coverage, for example:

> I'm not trying to give you **legal advice.** →

- I'm not trying to give you **a legal advice**.
- I'm not trying to give you **the** legal advice.
- I'm not trying to give you **the legal advices**.

A total of 24128 sentences from the news data were "lesioned" in this manner to create a set of 65826 sentence pairs. To create a balanced training set that would not introduce too many artifacts of the substitution (e.g., *many* should not always be recast as *much* just because that is the only mapping observed in the training data), we randomly created an equivalent number of identity-mapped pairs from the 346,000 examples, with each sentence mapping to itself.

Training sets of various sizes up to 45,000 pairs were then randomly extracted from the lesioned and non-lesioned pairs so that data from both sets occurred in roughly equal proportions. Thus the 45K data set contains approximately 22,500 lesioned examples. An additional 1,000 randomly selected lesioned sentences were set aside for lambda training the SMT system's ordering and replacement models.

4 Evaluation

4.1 Test Data

The amount of tagged data in *CLEC* is too small to yield both development and test sets from the same data. In order to create a test set, we had a third party collect 150 examples of the 14 words from English websites in China. After minor cleanup to eliminate sentences irrelevant to the task,[4] we ended up with 123 example sentences to use as test set. The test examples vary widely in style, from the highly casual to more formal public announcements. Thirteen examples were determined to contain no errors relevant to our experiment, but were retained in the data.[5]

4.2 Results

Table 1 shows per-sentence results of translating the test set on systems built with training data sets of various sizes (given in thousands of sentence pairs). Numbers for the proofing tools in Word 2003 are presented by way of comparison, with the caveat that these tools have been intentionally implemented conservatively so as not to potentially irritate native users with false flags. For our purposes, a replacement string is viewed as correct if, in the view of a native speaker who might be helping an ESL writer, the replacement would appear more natural and hence potentially useful as a suggestion in the context of that sentence taken in isolation. Number disagreement on subject and verb were ignored for the purposes of this evaluation, since these errors were not modeled when we introduced lesions into the data. A correction counted as Whole if the system produced a contextually plausible substitution meeting two criteria: 1) number and 2) determiner/quantifier selection (e.g., *many informations* → *much information*). Transformations involving bare singular targets (e.g., *the fruits* → *fruit*) also counted as Whole. Partial corrections are those where only one of the two criteria was met and part of the desired correction was missing (e.g., *an*

[4] In addition to eliminating cases that only involved subject-verb number agreement, we excluded a small amount of spam-like word salad, several instances of the word *homework* being misused to mean "work done out of the home", and one misidentified quotation from Scott's *Ivanhoe*.

[5] This test set may be downloaded at http://research.microsoft.com/research/downloads

Input	Shanghai residents can buy **the fruits** for a cheaper price than before.
Replacement	Shanghai residents can buy **fruit** for a cheaper price than before .
Input	Thank u for giving me **so many advice**.
Replacement	thank u for giving me **so much advice** .
Input	Acquiring **the knowledge** of information warfare is key to winning wars
Replacement	acquiring **knowledge** of information warfare is key to winning wars
Input	**Many knowledge** about Li Bai can be gain through it.
Replacement	**much knowledge** about Li Bai can be gain through it .
Input	I especially like drinking **the tea**.
Replacement	i especially like drinking **tea** .
Input	Icons printed on **a paper** have been brought from Europe, and were pasted on boards on Taiwan.
Replacement	icons printed on **paper** have been brought from Europe , and were pasted on boards on Taiwan .

Table 2. Sample corrections, using 45K engineered training data

equipments → an equipment versus the targeted bare noun equipment). Incorrect substitutions and newly injected erroneous material anywhere in the sentence counted as New Errors, even if the proposed replacement were otherwise correct. However, changes in upper and lower case and punctuation were ignored.

The 55.28% per-sentence score for Whole matches in the system trained on the 45K data set means that it correctly proposed full corrections in 61.8% of locations where corrections needed to be made. The percentage of Missed errors, i.e., targeted errors that were ignored by the system, is correspondingly low. On the 45K training data set, the system performs nearly on a par with Word in terms of not inducing corrections on forms that did not require replacement, as shown in the Correctly Left column. The dip in accuracy in the 30K sentence pair training set is an artifact of our extraction methodology: the relatively small lexical set that we are addressing here appears to be oversensitive to random variation in the engineered training data. This makes it difficult to set a meaningful lower bound on the amount of training data that might be needed for adequate coverage. Nonetheless, it is evident from the table, that given sufficient data, SMT techniques can successfully offer corrections for a significant percentage of cases of the phenomena in question.

Table 2 shows some sample inputs together with successful corrections made by the system. Table 3 illustrates a case where two valid corrections are found in the 5-best ranked translations; intervening candidates were identical with the top-ranked candidate.

4.3 Error Analysis

Table 1 also indicates that errors associated with the SMT system itself are encouragingly few. A small number of errors in word order were found, one of which resulted in a severely garbled sentence in the 45K data set. In general, the percentage of this type of error declines consistently with growth of the training data size. Linearity of the training data may play a role, since the sentence pairs differ by only a few words. On the whole, however, we expect the system's order model to benefit from more training data.

The most frequent single class of newly introduced error relates to sporadic substitution of the word their for determiners a/the. This is associated with three words, lunch, tea, and haste, and is the principal contributor to the lower percentages in the Correctly Left bin, as compared with Word. This overgeneralization error reflects our attempt to engineer the discontinuous mapping the X of them → their X, motivated by examples like the following, encountered in the *CLEC* dataset:

Input:	And we can learn **many knowledge** or new information from TV
Candidate 1:	And we can learn **much knowledge** or new information from TV
Candidate 5:	And we can learn **a lot of knowledge** or new information from TV

Table 3. Multiple replacement candidates generated by 45K training set

> In this equal world, lots of people are still concerned on **the colors of them …**

The inability of our translation system to handle such discontinuities in a unitary manner reflects the limited ability of current SMT modeling techniques to capture long-distance effects. Similar alternations are rife in bilingual data, e.g., *ne…pas* in French (Fox, 2002) and separable prefixes in German (Collins et al. 2005). As SMT models become more adept at modeling long-distance effects in a principled manner, monolingual proofing will benefit as well.

The Missed category is heterogeneous. The SMT system has an inherent bias against deletion, with the result that unwanted determiners tended not to be deleted, especially in the smaller training sets.

Other errors related to coverage in the development data set. Several occurrences of greengrocer's apostrophes (*tea's*, *equipment's*) caused correction failures: these were not anticipated when engineering the training data. Likewise, the test data presented several malformed quantifiers and quantifier-like phrases (*plenty tea* → *plenty of tea*, *a lot information* → *a lot of information*, *few information* → *too little information*) that had been unattested in the development set. Examples such as these highlight the difficulty in obtaining complete coverage when using handcrafted techniques, whether to engineer errors, as in our case, or to handcraft targeted correction solutions.

The system performed poorly on words that commonly present both mass and count noun senses in ways that are apt to confuse L2 writers. One problematic case was *paper*. The following sentences, for example, remained uncorrected:

> He published many paper in provincial and national publication.

> He has published thirty-two pieces of papers.

Large amounts of additional training data would doubtless be helpful in providing contextual resolutions to the problems. Improved alignment models may also play a role here in capturing complex structures of the kind represented by constructions involving counters.

5 Discussion

The artificially-engineered training data that we relied on for our experiments proved surprisingly useful in modeling real errors made by non-native speakers. However, this is obviously a less than ideal data source, since the errors introduced by regular expressions are homogenously distributed in a way that naturally-occurring errors are not, creating artifacts that undoubtedly impair our SMT models.

Artificial data of this sort may be useful as proof of concept, but hand engineering such data plainly does not present a viable path to developing real world applications. In order to be able to handle the rich panoply of errors and error interactions encountered in the text of second language learners large quantities of naturally-occurring "before" and "after" texts will need to be collected. By way of illustration, Table 4 shows the output of results of "translating" our test data into more natural English by hand and dumping the pre- and post-editing pairs to the 45K training set.[6] Although we were unable to exactly recover the target sentences, inspection showed that 25 sentences had improved, some significantly, as Table 4 shows. Under the right conditions, the SMT system can capture contextual morphological alternations (*nutrition/nutritious*), together with complex mappings represented by the dependencies *learn* ← *knowledge* ← *many* (ESL) and

[6] Since a single example of each pair was insufficient to override the system's inherent bias towards unigram mappings, 5 copies of each pair were appended to the training data.

Input sentence	And we can **learn many knowledge** or new information from TV.
45K system output	and we can **learn much knowledge** or new information from TV .
45K + translation system output	we can **gain a lot of knowledge** or new information from TV .
Input sentence	The following is **one of the homework** for last week.
45K system output	the following is **one of their homework** for last week .
45K + translation system output	the following is **one of the homework assignments** for last week .
Input sentence	i like mushroom,its **very nutrition**
45K system output	i like mushroom , its **very nutrition**
45K + translation system output	i like mushroom , its **very nutritious**

Table 4. Contextual corrections before and after adding “translations” to 45K training data

gain ← *knowledge* ← *a lot of* (English). In a rule-based correction system, an immense amount of hand-coding would be required to handle even a small subset of the potential range of such mismatches between learner and native-like English. This knowledge, we believe, is best acquired from data.

5.1 The Need for Data Collection

Given a sufficiently large corpus of aligned sentences containing error patterns produced by ESL writers of the same L1 background and their corrected counterparts we expect eventually to be able to capture the rich complexity of non-native error within a noisy-channel based SMT model.

As a practical matter, however, parallel data of the kind needed is far from easy to come by. This does not mean, however, that such data does not exist. The void left by commercial grammar checkers is filled, largely unobserved, by a number of services that provide editorial assistance, ranging from foreign language teachers, to language helpdesks in multinational corporations, to mentoring services for conferences. Translation bureaus frequently offer editing services for non-native speakers. Yet, unlike translation, the “before” and “after” texts are rarely recycled in a form that can be used to build translation models. Although collecting this data will involve a large investment in time, effort, and infrastructure, a serious effort along these lines is likely to prove fruitful in terms of making it possible to apply the SMT paradigm to ESL error correction.

5.2 Feedback to SMT

One challenge faced by the SMT model is the extremely high quality that will need to be attained before a system might be usable. Since it is highly undesirable that learners should be presented with inaccurate feedback that they may not have the experience or knowledge to assess, the quality bar imposed on error correction is far higher than is that tolerated in machine translation. Exploration of error correction and writing assistance using SMT models may thus prove an important venue for testing new SMT models.

5.3 Advantages of the SMT Approach

Statistical Machine Translation has provided a hugely successful research paradigm within the field of natural language processing over the last decade. One of the major advantages of using SMT in ESL writing assistance is that it can be expected to benefit automatically from any progress made in SMT itself. In fact, the approach presented here benefits from all the advantages of statistical machine translation. Since the architecture is not dependent on hard-to-maintain rules or regular expressions, little or no linguistic expertise will be required in developing and maintain applications. As with SMT, this expertise is pushed into the data component, to be handled by instructors and editors, who do not need programming or scripting skills.

We expect it to be possible, moreover, once parallel data becomes available, to quickly ramp up new systems to accommodate the needs of learners with different first-language backgrounds and different skill levels and to writing assistance for learners of L2s other than English. It is also likely that this architecture may have applications in pedagogical environments and as a tool to assist editors and instructors who deal regularly with ESL texts, much in the manner of either Human Assisted Machine Translation or Machine Assisted Human Translation. We also believe that this same architecture could be extended naturally to provide grammar and style tools for native writers.

6 Conclusion and Future Directions

In this pilot study we have shown that SMT techniques have potential to provide error correction and stylistic writing assistance to L2 learners. The next step will be to obtain a large dataset of pre- and post-editing ESL text with which to train a model that does not rely on engineered data. A major purpose of the present study has been to determine whether our hypothesis is robust enough to warrant the cost and effort of a collection or data creation effort.

Although we anticipate that it will take a significant lead time to assemble the necessary aligned data, once a sufficiently large corpus is in hand, we expect to begin exploring ways to improve our SMT system by tailoring it more specifically to the demands of editorial assistance. In particular, we expect to be looking into alternative word alignment models and possibly enhancing our system's decoder using some of the richer, more structured language models that are beginning to emerge.

Acknowledgements

The authors have benefited extensively from discussions with Casey Whitelaw when he interned at Microsoft Research during the summer of 2005. We also thank the Butler Hill Group for collecting the examples in our test set.

References

Bond, Francis, Kentaro Ogura and Satoru Ikehara. 1994. Countability and Number in Japanese-to-English Machine Translation. *COLING-94*.

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

Martin Chodorow and Claudia Leacock. 2000. An Unsupervised Method for Detecting Grammatical Errors. *NAACL 2000.*

Michael Collins, Philipp Koehn and Ivona Kučerová. 2005. Clause Restructuring for Statistical machine Translation. *ACL 2005,* 531-540.

Gerard M. Dalgish. 1984. Computer-Assisted ESL Research. *CALICO Journal.* 2(2): 32-33

Heidi J. Fox. 2002. Phrasal Cohesion and Statistical Machine Translation. *EMNLP 2002.*

Shicun Gui and Huizhong Yang (eds). 2003 *Zhongguo Xuexizhe Yingyu Yuliaohu. (Chinese Learner English Corpus).* Shanghai: Shanghai Waiyu Jiaoyu Chubanshe. (In Chinese).

Hua Dongfan and Thomas Hun-Tak Lee. 2004. Chinese ESL Learners' Understanding of the English Count-Mass Distinction. In *Proceedings of the 7th Generative Approaches to Second Language Acquisition Conference (GASLA 2004).*

Ting Liu, Ming Zhou, Jianfeng Gao, Endong Xun, and Changning Huang. 2000. PENS: A Machine-aided English Writing System for Chinese Users. *ACL 2000.*

Deryle Lonsdale and Diane Strong-Krause. 2003. Automated Rating of ESL Essays. In *Proceedings of the HLT/NAACL Workshop: Building Educational Applications Using Natural Language Processing*.

Arul Menezes, and Chris Quirk. 2005. *Microsoft Research Treelet Translation System: IWSLT Evaluation.* Proceedings of the International Workshop on Spoken Language Translation.

Franz Josef Och, 2003. Minimum error rate training in statistical machine translation. *ACL 2003*.

Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. *ACL 2000.*

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. *Dependency Tree Translation: Syntactically Informed Phrasal SMT.* ACL 2005.

Veit Reuer. 2003. Error Recognition and Feedback with Lexical Functional Grammar. *CALICO Journal*, 20(3): 497-512.

Laura Mayfield Tomokiyo and Rosie Jones. 2001. You're not from round here, are you? Naive Bayes Detection of Non-Native Utterance Text. *NAACL 2001.*

Anne Vandeventer Faltin. 2003. Natural language processing tools for computer assisted language learning. *Linguistik online* 17, 5/03 (http:// www.linguistik-online.de/17_03/vandeventer.html)

Graph Transformations in Data-Driven Dependency Parsing

Jens Nilsson
Växjö University
jni@msi.vxu.se

Joakim Nivre
Växjö University and
Uppsala University
nivre@msi.vxu.se

Johan Hall
Växjö University
jha@msi.vxu.se

Abstract

Transforming syntactic representations in order to improve parsing accuracy has been exploited successfully in statistical parsing systems using constituency-based representations. In this paper, we show that similar transformations can give substantial improvements also in data-driven dependency parsing. Experiments on the Prague Dependency Treebank show that systematic transformations of coordinate structures and verb groups result in a 10% error reduction for a deterministic data-driven dependency parser. Combining these transformations with previously proposed techniques for recovering non-projective dependencies leads to state-of-the-art accuracy for the given data set.

1 Introduction

It has become increasingly clear that the choice of suitable internal representations can be a very important factor in data-driven approaches to syntactic parsing, and that accuracy can often be improved by internal transformations of a given kind of representation. This is well illustrated by the Collins parser (Collins, 1997; Collins, 1999), scrutinized by Bikel (2004), where several transformations are applied in order to improve the analysis of noun phrases, coordination and punctuation. Other examples can be found in the work of Johnson (1998) and Klein and Manning (2003), which show that well-chosen transformations of syntactic representations can greatly improve the parsing accuracy obtained with probabilistic context-free grammars.

In this paper, we apply essentially the same techniques to data-driven dependency parsing, specifically targeting the analysis of *coordination* and *verb groups*, two very common constructions that pose special problems for dependency-based approaches. The basic idea is that we can facilitate learning by transforming the training data for the parser and that we can subsequently recover the original representations by applying an inverse transformation to the parser's output.

The data used in the experiments come from the Prague Dependency Treebank (PDT) (Hajič, 1998; Hajič et al., 2001), the largest available dependency treebank, annotated according to the theory of Functional Generative Description (FGD) (Sgall et al., 1986). The parser used is MaltParser (Nivre and Hall, 2005; Nivre et al., 2006), a freely available system that combines a deterministic parsing strategy with discriminative classifiers for predicting the next parser action.

The paper is structured as follows. Section 2 provides the necessary background, including a definition of dependency graphs, a discussion of different approaches to the analysis of coordination and verb groups in dependency grammar, as well as brief descriptions of PDT, MaltParser and some related work. Section 3 introduces a set of dependency graph transformations, specifically defined to deal with the dependency annotation found in PDT, which are experimentally evaluated in section 4. While the experiments reported in section 4.1 deal with pure treebank transformations, in order to establish an upper bound on what can be achieved in parsing, the experiments presented in section 4.2 examine the effects of different transformations on parsing accuracy. Finally, in section 4.3, we combine these transformations with previously proposed techniques in order to optimize overall parsing accuracy. We conclude in section 5.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 257–264,
Sydney, July 2006. ©2006 Association for Computational Linguistics

2 Background

2.1 Dependency Graphs

The basic idea in dependency parsing is that the syntactic analysis consists in establishing typed, binary relations, called *dependencies*, between the words of a sentence. This kind of analysis can be represented by a labeled directed graph, defined as follows:

- Let $R = \{r_1, \ldots, r_m\}$ be a set of dependency types (arc labels).
- A dependency graph for a string of words $W = w_1 \ldots w_n$ is a labeled directed graph $G = (W, A)$, where:
  - W is the set of nodes, i.e. word tokens in the input string, ordered by a linear precedence relation $<$.
  - A is a set of labeled arcs (w_i, r, w_j), w_i, $w_j \in W$, $r \in R$.
- A dependency graph $G = (W, A)$ is well-formed iff it is acyclic and no node has an in-degree greater than 1.

We will use the notation $w_i \xrightarrow{r} w_j$ to symbolize that $(w_i, r, w_j) \in A$, where w_i is referred to as the *head* and w_j as the *dependent*. We say that an arc is *projective* iff, for every word w_j occurring between w_i and w_k (i.e., $w_i < w_j < w_k$ or $w_i > w_j > w_k$), there is a path from w_i to w_j. A graph is projective iff all its arcs are projective. Figure 1 shows a well-formed (projective) dependency graph for a sentence from the Prague Dependency Treebank.

2.2 Coordination and Verb Groups

Dependency grammar assumes that syntactic structure consists of lexical nodes linked by binary dependencies. Dependency theories are thus best suited for binary syntactic constructions, where one element can clearly be distinguished as the syntactic head. The analysis of coordination is problematic in this respect, since it normally involves at least one conjunction and two conjuncts. The verb group, potentially consisting of a whole chain of verb forms, is another type of construction where the syntactic relation between elements is not clear-cut in dependency terms.

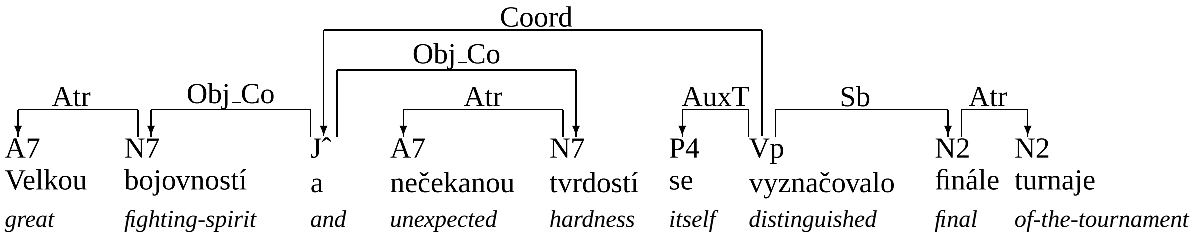
Several solutions have been proposed to the problem of coordination. One alternative is to avoid creating dependency relations between the conjuncts, and instead let the conjuncts have a direct dependency relation to the same head (Tesnière, 1959; Hudson, 1990). Another approach is to make the conjunction the head and let the conjuncts depend on the conjunction. This analysis, which appears well motivated on semantic grounds, is adopted in the FGD framework and will therefore be called Prague style (PS). It is exemplified in figure 1, where the conjunction *a* (and) is the head of the conjuncts *bojovností* and *tvrdostí*. A different solution is to adopt a more hierarchical analysis, where the conjunction depends on the first conjunct, while the second conjunct depends on the conjunction. In cases of multiple coordination, this can be generalized to a chain, where each element except the first depends on the preceding one. This more syntactically oriented approach has been advocated notably by Mel'čuk (1988) and will be called Mel'čuk style (MS). It is illustrated in figure 2, which shows a transformed version of the dependency graph in figure 1, where the elements of the coordination form a chain with the first conjunct (*bojovností*) as the topmost head. Lombardo and Lesmo (1998) conjecture that MS is more suitable than PS for incremental dependency parsing.

The difference between the more semantically oriented PS and the more syntactically oriented MS is seen also in the analysis of verb groups, where the former treats the main verb as the head, since it is the bearer of valency, while the latter treats the auxiliary verb as the head, since it is the finite element of the clause. Without questioning the theoretical validity of either approach, we can again ask which analysis is best suited to achieve high accuracy in parsing.

2.3 PDT

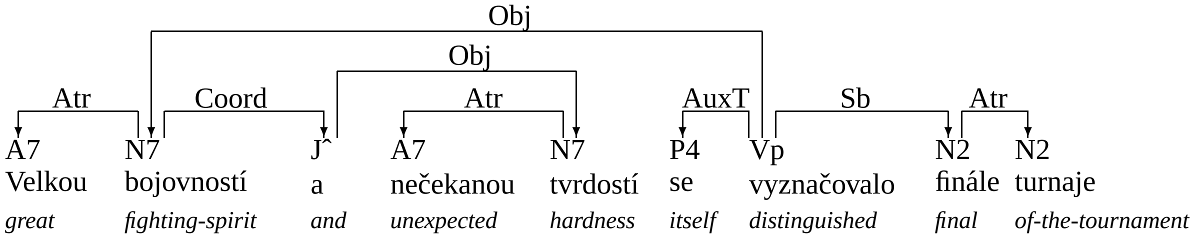
PDT (Hajič, 1998; Hajič et al., 2001) consists of 1.5M words of newspaper text, annotated in three layers: morphological, analytical and tectogrammatical. In this paper, we are only concerned with the analytical layer, which contains a surface-syntactic dependency analysis, involving a set of 28 dependency types, and not restricted to projective dependency graphs.[1] The annotation follows FGD, which means that it involves a PS analysis of both coordination and verb groups. Whether better parsing accuracy can be obtained by transforming

[1] About 2% of all dependencies are non-projective and about 25% of all sentences have a non-projective dependency graph (Nivre and Nilsson, 2005).



(“The final of the tournament was distinguished by great fighting spirit and unexpected hardness”)

Figure 1: Dependency graph for a Czech sentence from the Prague Dependency Treebank



(“The final of the tournament was distinguished by great fighting spirit and unexpected hardness”)

Figure 2: Transformed dependency graph for a Czech sentence from the Prague Dependency Treebank

this to MS is one of the hypotheses explored in the experimental study below.

2.4 MaltParser

MaltParser (Nivre and Hall, 2005; Nivre et al., 2006) is a data-driven parser-generator, which can induce a dependency parser from a treebank, and which supports several parsing algorithms and learning algorithms. In the experiments below we use the algorithm of Nivre (2003), which constructs a labeled dependency graph in one left-to-right pass over the input. Classifiers that predict the next parser action are constructed through memory-based learning (MBL), using the TiMBL software package (Daelemans and Van den Bosch, 2005), and support vector machines (SVM), using LIBSVM (Chang and Lin, 2005).

2.5 Related Work

Other ways of improving parsing accuracy with respect to coordination include learning patterns of morphological and semantical information for the conjuncts (Park and Cho, 2000). More specifically for PDT, Collins et al. (1999) relabel coordinated phrases after converting dependency structures to phrase structures, and Zeman (2004) uses a kind of pattern matching, based on frequencies of the parts-of-speech of conjuncts and conjunctions. Zeman also mentions experiments to transform the dependency structure for coordination but does not present any results.

Graph transformations in dependency parsing have also been used in order to recover non-projective dependencies together with parsers that are restricted to projective dependency graphs. Thus, Nivre and Nilsson (2005) improve parsing accuracy for MaltParser by projectivizing training data and applying an inverse transformation to the output of the parser, while Hall and Novák (2005) apply post-processing to the output of Charniak’s parser (Charniak, 2000). In the final experiments below, we combine these techniques with the transformations investigated in this paper.

3 Dependency Graph Transformations

In this section, we describe algorithms for transforming dependency graphs in PDT from PS to MS and back, starting with coordination and continuing with verb groups.

3.1 Coordination

The PS-to-MS transformation for coordination will be designated $\tau_c(\Delta)$, where Δ is a data set. The transformation begins with the identification of a *base conjunction*, based on its dependency type (*Coord*) and/or its part-of-speech (*J^*). For example, the word *a* (and) in figure 1 is identified as a base conjunction.

Before the actual transformation, the base conjunction and all its dependents need to be classified into three different categories. First, the base conjunction is categorized as a *separator* (S). If the coordination consists of more than two conjuncts, it normally has one or more commas separating conjuncts, in addition to the base conjunction. These are identified by looking at their dependency type (mostly *AuxX*) and are also categorized as S. The coordination in figure 1 contains no commas, so only the word *a* will belong to S.

The remaining dependents of the base conjunction need to be divided into conjuncts (C) and other dependents (D). To make this distinction, the algorithm again looks at the dependency type. In principle, the dependency type of a conjunct has the suffix *_Co*, although special care has to be taken for coordinated prepositional cases and embedded clauses (Böhmová et al., 2003). The words *bojovností* and *tvrdostí* in figure 1, both having the dependency type *Obj_Co*, belong to the category C. Since there are no other dependents of *a*, the coordination contains no instances of the category D.

Given this classification of the words involved in a coordination, the transformation $\tau_c(\Delta)$ is straightforward and basically connects all the arcs in a chain. Let $C_1, \ldots, C_n$ be the elements of C, ordered by linear precedence, and let $S_{1_i}, \ldots, S_{m_i}$ be the separators occurring between C_i and C_{i+1}. Then every C_i becomes the head of $S_{1_i}, \ldots, S_{m_i}$, S_{m_i} becomes the head of C_{i+1}, and C_1 becomes the only dependent of the original head of the base conjunction. The dependency types of the conjuncts are truncated by removing the suffix *_Co*.[2] Also, each word in $w_d \in D$ becomes a dependent of the conjunct closest to its left, and if such a word does not exist, w_d will depend on the leftmost conjunct. After the transformation $\tau_c(\Delta)$, every coordination forms a left-headed chain, as illustrated in figure 2.

This new representation creates a problem, however. It is no longer possible to distinguish the dependents in D from other dependents of the conjuncts. For example, the word *Velkou* in figure 2 is not distinguishable from a possible dependent in D, which is an obvious drawback when transforming back to PS. One way of distinguishing D elements is to extend the set of dependency types. The dependency type r of each $w_d \in D$ can be replaced by a completely new dependency type $r+$ (e.g., $Atr+$), theoretically increasing the number of dependency types to $2 \cdot |R|$.

The inverse transformation, $\tau_c^{-1}(\Delta)$, again starts by identifying base conjunctions, using the same conditions as before. For each identified base conjunction, it calls a procedure that performs the inverse transformation by traversing the chain of conjuncts and separators "upwards" (right-to-left), collecting conjuncts (C), separators (S) and *potential* conjunction dependents (D_{pot}). When this is done, the former head of the leftmost conjunct (C_1) becomes the head of the rightmost (base) conjunction ($S_{m_{n-1}}$). In figure 2, the leftmost conjunct is *bojovností*, with the head *vyznačovalo*, and the rightmost (and only) conjunction is *a*, which will then have *vyznačovalo* as its new head. All conjuncts in the chain become dependents of the rightmost conjunction, which means that the structure is converted back to the one depicted in figure 1.

As mentioned above, the original structure in figure 1 did not have any coordination dependents, but *Velkou* $\in D_{pot}$. The last step of the inverse transformation is therefore to sort out conjunction dependents from conjunct dependents, where the former will attach to the base conjunction. Four versions have been implemented, two of which take into account the fact that the dependency types *AuxG*, *AuxX*, *AuxY*, and *Pred* are the only dependency types that are more frequent as conjunction dependents (D) than as conjunct dependents in the training data set:

- τ_c: Do not extend arc labels in τ_c. Leave all words in D_{pot} in place in τ_c^{-1}.
- τ_{c^*}: Do not extend arc labels in τ_c. Attach all words with label *AuxG*, *AuxX*, *AuxY* or *Pred* to the base conjunction in τ_c^{-1}.
- τ_{c+}: Extend arc labels from r to $r+$ for D elements in τ_c. Attach all words with label $r+$ to the base conjunction (and change the label to r) in τ_c^{-1}.
- τ_{c+^*}: Extend arc labels from r to $r+$ for D elements in τ_c, except for the labels *AuxG*, *AuxX*, *AuxY* and *Pred*. Attach all words with label $r+$, *AuxG*, *AuxX*, *AuxY*, or *Pred* to the base conjunction (and change the label to r if necessary) in τ_c^{-1}.

[2] Preliminary results indicated that this increases parsing accuracy.

3.2 Verb Groups

To transform verb groups from PS to MS, the transformation algorithm, $\tau_v(\Delta)$, starts by identifying all auxiliary verbs in a sentence. These will belong to the set A and are processed from left to right. A word $w_{aux} \in A$ iff $w_{main} \xrightarrow{AuxV} w_{aux}$, where w_{main} is the main verb. The transformation into MS reverses the relation between the verbs, i.e., $w_{aux} \xrightarrow{AuxV} w_{main}$, and the former head of w_{main} becomes the new head of w_{aux}. The main verb can be located on either side of the auxiliary verb and can have other dependents (whereas auxiliary verbs never have dependents), which means that dependency relations to other dependents of w_{main} may become non-projective through the transformation. To avoid this, all dependents to the left of the rightmost verb will depend on the leftmost verb, whereas the others will depend on the rightmost verb.

Performing the inverse transformation for verb groups, $\tau_v^{-1}(\Delta)$, is quite simple and essentially the same procedure inverted. Each sentence is traversed from right to left looking for arcs of the type $w_{aux} \xrightarrow{AuxV} w_{main}$. For every such arc, the head of w_{aux} will be the new head of w_{main}, and w_{main} the new head of w_{aux}. Furthermore, since w_{aux} does not have dependents in PS, all dependents of w_{aux} in MS will become dependents of w_{main} in PS.

4 Experiments

All experiments are based on PDT 1.0, which is divided into three data sets, a training set (Δ_t), a development test set (Δ_d), and an evaluation test set (Δ_e). Table 1 shows the size of each data set, as well as the relative frequency of the specific constructions that are in focus here. Only 1.3% of all words in the training data are identified as auxiliary verbs (A), whereas coordination (S and C) is more common in PDT. This implies that coordination transformations are more likely to have a greater impact on overall accuracy compared to the verb group transformations.

In the parsing experiments reported in sections 4.1–4.2, we use Δ_t for training, Δ_d for tuning, and Δ_e for the final evaluation. The part-of-speech tagging used (both in training and testing) is the HMM tagging distributed with the treebank, with a tagging accuracy of 94.1%, and with the tagset compressed to 61 tags as in Collins et al. (1999).

Data	#S	#W	%S	%C	%A
Δ_t	73088	1256k	3.9	7.7	1.3
Δ_d	7319	126k	4.0	7.8	1.4
Δ_e	7507	126k	3.8	7.3	1.4

Table 1: PDT data sets; S = sentence, W = word; S = separator, C = conjunct, A = auxiliary verb

T	AS
τ_c	97.8
τ_{c^*}	98.6
τ_{c+}	99.6
τ_{c+^*}	99.4
τ_v	100.0

Table 2: Transformations; T = transformation; AS = attachment score (unlabeled) of $\tau^{-1}(\tau(\Delta_t))$ compared to Δ_t

MaltParser is used with the parsing algorithm of Nivre (2003) together with the feature model used for parsing Czech by Nivre and Nilsson (2005). In section 4.2 we use MBL, again with the same settings as Nivre and Nilsson (2005),[3] and in section 4.2 we use SVM with a polynomial kernel of degree 2.[4] The metrics for evaluation are the attachment score (AS) (labeled and unlabeled), i.e., the proportion of words that are assigned the correct head, and the exact match (EM) score (labeled and unlabeled), i.e., the proportion of sentences that are assigned a completely correct analysis. All tokens, including punctuation, are included in the evaluation scores. Statistical significance is assessed using McNemar's test.

4.1 Experiment 1: Transformations

The algorithms are fairly simple. In addition, there will always be a small proportion of syntactic constructions that do not follow the expected pattern. Hence, the transformation and inverse transformation will inevitably result in some distortion. In order to estimate the expected reduction in parsing accuracy due to this distortion, we first consider a pure treebank transformation experiment, where we compare $\tau^{-1}(\tau(\Delta_t))$ to Δ_t, for all the different transformations τ defined in the previous section. The results are shown in table 2.

We see that, even though coordination is more frequent, verb groups are easier to handle.[5] The

[3] TiMBL parameters: -k5 -mM -L3 -w0 -dID.

[4] LIBSVM parameters: -s0 -t1 -d2 -g0.12 -r0 -c1 -e0.1.

[5] The result is rounded to 100.0% but the transformed tree-

coordination version with the least loss of information (τ_{c+}) fails to recover the correct head for 0.4% of all words in Δ_t.

The difference between τ_{c+} and τ_c is expected. However, in the next section this will be contrasted with the increased burden on the parser for τ_{c+}, since it is also responsible for selecting the correct dependency type for each arc among as many as $2 \cdot |R|$ types instead of $|R|$.

4.2 Experiment 2: Parsing

Parsing experiments are carried out in four steps (for a given transformation τ):

1. Transform the training data set into $\tau(\Delta_t)$.
2. Train a parser p on $\tau(\Delta_t)$.
3. Parse a test set Δ using p with output $p(\Delta)$.
4. Transform the parser output into $\tau^{-1}(p(\Delta))$.

Table 3 presents the results for a selection of transformations using MaltParser with MBL, tested on the evaluation test set Δ_e with the untransformed data as baseline. Rows 2–5 show that transforming coordinate structures to MS improves parsing accuracy compared to the baseline, regardless of which transformation and inverse transformation are used. Moreover, the parser benefits from the verb group transformation, as seen in row 6.

The final row shows the best combination of a coordination transformation with the verb group transformation, which amounts to an improvement of roughly two percentage points, or a ten percent overall error reduction, for unlabeled accuracy.

All improvements over the baseline are statistically significant (McNemar's test) with respect to attachment score (labeled and unlabeled) and unlabeled exact match, with $p < 0.01$ except for the unlabeled exact match score of the verb group transformation, where $0.01 < p < 0.05$. For the labeled exact match, no differences are significant.

The experimental results indicate that MS is more suitable than PS as the target representation for deterministic data-driven dependency parsing. A relevant question is of course why this is the case. A partial explanation may be found in the "short-dependency preference" exhibited by most parsers (Eisner and Smith, 2005), with MaltParser being no exception. The first row of table 4 shows the accuracy of the parser for different arc lengths under the baseline condition (i.e., with no transformations). We see that it performs very well on short arcs, but that accuracy drops quite rapidly as the arcs get longer. This can be related to the mean arc length in Δ_t, which is 2.59 in the untransformed version, 2.40 in $\tau_c(\Delta_t)$ and 2.54 in $\tau_v(\Delta_t)$. Rows 3-5 in table 4 show the distribution of arcs for different arc lengths in different versions of the data set. Both τ_c and τ_v make arcs shorter on average, which may facilitate the task for the parser.

	AS		**EM**	
T	**U**	**L**	**U**	**L**
None	79.08	72.83	28.99	21.15
τ_c	80.55	74.06	30.08	21.27
τ_{c^*}	80.90	74.41	30.56	21.42
τ_{c+}	80.58	74.07	30.42	21.17
τ_{c+^*}	80.87	74.36	30.89	21.38
τ_v	79.28	72.97	29.53	21.38
$\tau_v \circ \tau_{c+^*}$	81.01	74.51	31.02	21.57

Table 3: Parsing accuracy (MBL, Δ_e); T = transformation; AS = attachment score, EM = exact match; U = unlabeled, L = labeled

AS Δ_e	90.1	83.6	70.5	59.5	45.9
Length:	**1**	**2-3**	**4-6**	**7-10**	**11-**
Δ_t	51.9	**29.4**	**11.2**	**4.4**	**3.0**
$\tau_c(\Delta_t)$	**54.1**	29.1	10.7	3.8	2.4
$\tau_v(\Delta_t)$	**52.9**	29.2	10.7	4.2	2.9

Table 4: Baseline labeled AS per arc length on Δ_e (row 1); proportion of arcs per arc length in Δ_t (rows 3–5)

Another possible explanation is that learning is facilitated if similar constructions are represented similarly. For instance, it is probable that learning is made more difficult when a unit has different heads depending on whether it is part of a coordination or not.

4.3 Experiment 3: Optimization

In this section we combine the best results from the previous section with the graph transformations proposed by Nivre and Nilsson (2005) to recover non-projective dependencies. We write τ_p for the projectivization of training data and τ_p^{-1} for the inverse transformation applied to the parser's output.[6] In addition, we replace MBL with SVM, a learning algorithm that tends to give higher accuracy in classifier-based parsing although it is more

bank contains 19 erroneous heads.

[6] More precisely, we use the variant called PATH in Nivre and Nilsson (2005).

T	LA	AS U	AS L	EM U	EM L
None	MBL	79.08	72.83	28.99	21.15
τ_p	MBL	80.79	74.39	31.54	22.53
$\tau_p \circ \tau_v \circ \tau_{c+^*}$	MBL	82.93	76.31	34.17	23.01
None	SVM	81.09	75.68	32.24	25.02
τ_p	SVM	82.93	77.28	35.99	27.05
$\tau_p \circ \tau_v \circ \tau_{c+^*}$	SVM	84.55	78.82	37.63	27.69

Table 5: Optimized parsing results (SVM, Δ_e); T = transformation; LA = learning algorithm; AS = attachment score, EM = exact match; U = unlabeled, L = labeled

T	P:S	R:S	P:C	R:C	P:A	R:A	P:M	R:M
None	52.63	72.35	55.15	67.03	82.17	82.21	69.95	69.07
$\tau_p \circ \tau_v \circ \tau_{c+^*}$	63.73	82.10	63.20	75.14	90.89	92.79	80.02	81.40

Table 6: Detailed results for SVM; T = transformation; P = unlabeled precision, R = unlabeled recall

costly to train (Sagae and Lavie, 2005).

Table 5 shows the results, for both MBL and SVM, of the baseline, the pure pseudo-projective parsing, and the combination of pseudo-projective parsing with PS-to-MS transformations. We see that pseudo-projective parsing brings a very consistent increase in accuracy of at least 1.5 percentage points, which is more than that reported by Nivre and Nilsson (2005), and that the addition of the PS-to-MS transformations increases accuracy with about the same margin. We also see that SVM outperforms MBL by about two percentage points across the board, and that the positive effect of the graph transformations is most pronounced for the unlabeled exact match score, where the improvement is more than five percentage points overall for both MBL and SVM.

Table 6 gives a more detailed analysis of the parsing results for SVM, comparing the optimal parser to the baseline, and considering specifically the (unlabeled) precision and recall of the categories involved in coordination (separators S and conjuncts C) and verb groups (auxiliary verbs A and main verbs M). All figures indicate, without exception, that the transformations result in higher precision and recall for all directly involved words. (All differences are significant beyond the 0.01 level.) It is worth noting that the error reduction is actually higher for A and M than for S and C, although the former are less frequent.

With respect to unlabeled attachment score, the results of the optimized parser are slightly below the best published results for a single parser. Hall and Novák (2005) report a score of 85.1%, applying a corrective model to the output of Charniak's parser; McDonald and Pereira (2006) achieve a score of 85.2% using a second-order spanning tree algorithm. Using ensemble methods and a pool of different parsers, Zeman and Žabokrtský (2005) attain a top score of 87.0%. For unlabeled exact match, our results are better than any previously reported results, including those of McDonald and Pereira (2006). (For the labeled scores, we are not aware of any comparable results in the literature.)

5 Conclusion

The results presented in this paper confirm that choosing the right representation is important in parsing. By systematically transforming the representation of coordinate structures and verb groups in PDT, we achieve a 10% error reduction for a data-driven dependency parser. Adding graph transformations for non-projective dependency parsing gives a total error reduction of about 20% (even more for unlabeled exact match). In this way, we achieve state-of-the-art accuracy with a deterministic, classifier-based dependency parser.

Acknowledgements

The research presented in this paper was partially supported by the Swedish Research Council. We are grateful to Jan Hajič and Daniel Zeman for help with the Czech data and to three anonymous reviewers for helpful comments and suggestions.

References

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30:479–511.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The Prague Dependency Treebank: A three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*. Kluwer Academic Publishers.

Chih-Chung Chang and Chih-Jen Lin. 2005. LIBSVM: A library for support vector machines.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 132–139.

Michael Collins, Jan Hajič, Eric Brill, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 505–512.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annatual Meeting of the Association for Computational Linguistics (ACL)*, pages 16–23.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Walter Daelemans and Antal Van den Bosch. 2005. *Memory-Based Language Processing*. Cambridge University Press.

Jason Eisner and Noah A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*.

Jan Hajič, Barbora Vidova Hladka, Jarmila Panevová, Eva Hajičová, Petr Sgall, and Petr Pajas. 2001. Prague Dependency Treebank 1.0. LDC, 2001T10.

Jan Hajič. 1998. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning*, pages 12–19. Prague Karolinum, Charles University Press.

Keith Hall and Vaclav Novák. 2005. Corrective modeling for non-projective dependency parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*.

Richard Hudson. 1990. *English Word Grammar*. Basil Blackwell.

Mark Johnson. 1998. Pcfg models of linguistic tree representations. *Computational Linguistics*, 24:613–632.

Dan Klein and Christopher Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 423–430.

Vincenzo Lombardo and Leonardo Lesmo. 1998. Unit coordination and gapping in dependency theory. In *Proceedings of the Workshop on Processing of Dependency-Based Grammars*, pages 11–20.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Igor Mel'cuk. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Joakim Nivre and Johan Hall. 2005. MaltParser: A language-independent system for data-driven dependency parsing. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT)*, pages 137–148.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Jong C. Park and Hyung Joon Cho. 2000. Informed parsing for coordination with combinatory categorial grammar. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*, pages 593–599.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*, pages 125–132.

Petr Sgall, Eva Hajičová, and Jarmila Panevová. 1986. *The Meaning of the Sentence in Its Pragmatic Aspects*. Reidel.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Editions Klincksieck.

Daniel Zeman and Zdeněk Žabokrtský. 2005. Improving parsing accuracy by combining diverse dependency parsers. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT)*.

Daniel Zeman. 2004. *Parsing with a Statistical Dependency Model*. Ph.D. thesis, Charles University.

Learning to Generate Naturalistic Utterances Using Reviews in Spoken Dialogue Systems

Ryuichiro Higashinaka
NTT Corporation
rh@cslab.kecl.ntt.co.jp

Rashmi Prasad
University of Pennsylvania
rjprasad@linc.cis.upenn.edu

Marilyn A. Walker
University of Sheffield
walker@dcs.shef.ac.uk

Abstract

Spoken language generation for dialogue systems requires a dictionary of mappings between semantic representations of concepts the system wants to express and realizations of those concepts. Dictionary creation is a costly process; it is currently done by hand for each dialogue domain. We propose a novel unsupervised method for learning such mappings from user reviews in the target domain, and test it on restaurant reviews. We test the hypothesis that user reviews that provide individual ratings for distinguished attributes of the domain entity make it possible to map review sentences to their semantic representation with high precision. Experimental analyses show that the mappings learned cover most of the domain ontology, and provide good linguistic variation. A subjective user evaluation shows that the consistency between the semantic representations and the learned realizations is high and that the naturalness of the realizations is higher than a hand-crafted baseline.

1 Introduction

One obstacle to the widespread deployment of spoken dialogue systems is the cost involved with hand-crafting the spoken language generation module. Spoken language generation requires a dictionary of mappings between semantic representations of concepts the system wants to express and realizations of those concepts. Dictionary creation is a costly process: an automatic method for creating them would make dialogue technology more scalable. A secondary benefit is that a learned dictionary may produce more natural and colloquial utterances.

We propose a novel method for mining user reviews to automatically acquire a domain specific generation dictionary for information presentation in a dialogue system. Our hypothesis is that reviews that provide individual ratings for various distinguished attributes of review entities can be used to map review sentences to a semantic representation. Figure 1 shows a user review in the restaurant domain, where we hypothesize that the user rating *food=5* indicates that the semantic representation for the sentence "The best Spanish food in New York" includes the relation 'RESTAURANT *has foodquality=5.'*

An example user review (we8there.com)

Ratings	Food=5, Service=5, Atmosphere=5, Value=5, Overall=5
Review comment	The best Spanish food in New York. I am from Spain and I had my 28th birthday there and we all had a great time. Salud!

↓

Review comment after named entity recognition

The best {NE=foodtype, string=Spanish} {NE=food, string=food, rating=5} in {NE=location, string=New York}. . . .

↓

Mapping between a semantic representation (a set of relations) and a syntactic structure (DSyntS)

- **Relations:**
  RESTAURANT has FOODTYPE
  RESTAURANT has foodquality=5
  RESTAURANT has LOCATION
  ([**foodtype, food=5, location**] for shorthand.)
- **DSyntS:**

$$\left[\begin{array}{l} \text{lexeme : food} \\ \text{class : common\_noun} \\ \text{number : sg} \\ \text{article : def} \\ \text{ATTR} \left[\begin{array}{l} \text{lexeme : best} \\ \text{class : adjective} \end{array}\right] \\ \text{ATTR} \left[\begin{array}{l} \text{lexeme : } \texttt{FOODTYPE} \\ \text{class : common\_noun} \\ \text{number : sg} \\ \text{article : no-art} \end{array}\right] \\ \text{ATTR} \left[\begin{array}{l} \text{lexeme : in} \\ \text{class : preposition} \\ \text{II} \left[\begin{array}{l} \text{lexeme : } \texttt{LOCATION} \\ \text{class : proper\_noun} \\ \text{number : sg} \\ \text{article : no-art} \end{array}\right] \end{array}\right] \end{array}\right]$$

Figure 1: Example of procedure for acquiring a generation dictionary mapping.

We apply the method to extract 451 mappings from restaurant reviews. Experimental analyses show that the mappings learned cover most of the domain ontology, and provide good linguistic variation. A subjective user evaluation indicates that the consistency between the semantic representations and the learned realizations is high and that the naturalness of the realizations is significantly higher than a hand-crafted baseline.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 265–272,
Sydney, July 2006.

Section 2 provides a step-by-step description of the method. Sections 3 and 4 present the evaluation results. Section 5 covers related work. Section 6 summarizes and discusses future work.

2 Learning a Generation Dictionary

Our automatically created generation dictionary consists of triples $(\mathcal{U}, \mathcal{R}, \mathcal{S})$ representing a mapping between the original utterance $\mathcal{U}$ in the user review, its semantic representation $\mathcal{R}(\mathcal{U})$, and its syntactic structure $\mathcal{S}(\mathcal{U})$. Although templates are widely used in many practical systems (Seneff and Polifroni, 2000; Theune, 2003), we derive syntactic structures to represent the potential realizations, in order to allow aggregation, and other syntactic transformations of utterances, as well as context specific prosody assignment (Walker et al., 2003; Moore et al., 2004).

The method is outlined briefly in Fig. 1 and described below. It comprises the following steps:

1. Collect user reviews on the web to create a population of utterances $\mathcal{U}$.
2. To derive semantic representations $\mathcal{R}(\mathcal{U})$:
   - Identify distinguished attributes and construct a domain ontology;
   - Specify lexicalizations of attributes;
   - Scrape webpages' structured data for named-entities;
   - Tag named-entities.
3. Derive syntactic representations $\mathcal{S}(\mathcal{U})$.
4. Filter inappropriate mappings.
5. Add mappings $(\mathcal{U}, \mathcal{R}, \mathcal{S})$ to dictionary.

2.1 Creating the corpus

We created a corpus of restaurant reviews by scraping 3,004 user reviews of 1,810 restaurants posted at we8there.com (http://www.we8there.com/), where each individual review includes a 1-to-5 Likert-scale rating of different restaurant attributes. The corpus consists of 18,466 sentences.

2.2 Deriving semantic representations

The distinguished attributes are extracted from the webpages for each restaurant entity. They include attributes that the users are asked to rate, i.e. *food, service, atmosphere, value*, and *overall*, which have scalar values. In addition, other attributes are extracted from the webpage, such as the *name*, *foodtype* and *location* of the restaurant, which have categorical values. The *name* attribute is assumed to correspond to the restaurant entity. Given the distinguished attributes, a simple domain ontology can be automatically derived by assuming that a meronymy relation, represented by the predicate *'has'*, holds between the entity type (RESTAURANT) and the distinguished attributes. Thus, the domain ontology consists of the relations:

Dist. Attr.	Lexicalization
food	food, meal
service	service, staff, waitstaff, wait staff, server, waiter, waitress
atmosphere	atmosphere, decor, ambience, decoration
value	value, price, overprice, pricey, expensive, inexpensive, cheap, affordable, afford
overall	recommend, place, experience, establishment

Table 1: Lexicalizations for distinguished attributes.

$$\left\{ \begin{array}{l} \text{RESTAURANT has foodquality} \\ \text{RESTAURANT has servicequality} \\ \text{RESTAURANT has valuequality} \\ \text{RESTAURANT has atmospherequality} \\ \text{RESTAURANT has overallquality} \\ \text{RESTAURANT has foodtype} \\ \text{RESTAURANT has location} \end{array} \right.$$

We assume that, although users may discuss other attributes of the entity, at least some of the utterances in the reviews realize the relations specified in the ontology. Our problem then is to identify these utterances. We test the hypothesis that, if an utterance $\mathcal{U}$ contains named-entities corresponding to the distinguished attributes, that $\mathcal{R}$ for that utterance includes the relation concerning that attribute in the domain ontology.

We define named-entities for lexicalizations of the distinguished attributes, starting with the seed word for that attribute on the webpage (Table 1).[1] For named-entity recognition, we use GATE (Cunningham et al., 2002), augmented with named-entity lists for locations, food types, restaurant names, and food subtypes (e.g. *pizza*), scraped from the we8there webpages.

We also hypothesize that the rating given for the distinguished attribute specifies the scalar value of the relation. For example, a sentence containing *food* or *meal* is assumed to realize the relation 'RESTAURANT *has foodquality*.', and the value of the *foodquality* attribute is assumed to be the value specified in the user rating for that attribute, e.g. 'RESTAURANT *has foodquality = 5'* in Fig. 1. Similarly, the other relations in Fig. 1 are assumed to be realized by the utterance "The best Spanish food in New York" because it contains

[1] In future, we will investigate other techniques for bootstrapping these lexicalizations from the seed word on the webpage.

filter	filtered	retained
No Relations Filter	7,947	10,519
Other Relations Filter	5,351	5,168
Contextual Filter	2,973	2,195
Unknown Words Filter	1,467	728
Parsing Filter	216	512

Table 2: Filtering statistics: the number of sentences filtered and retained by each filter.

one FOODTYPE named-entity and one LOCATION named-entity. Values of categorical attributes are replaced by variables representing their type before the learned mappings are added to the dictionary, as shown in Fig. 1.

2.3 Parsing and DSyntS conversion

We adopt Deep Syntactic Structures (DSyntSs) as a format for syntactic structures because they can be realized by the fast portable realizer RealPro (Lavoie and Rambow, 1997). Since DSyntSs are a type of dependency structure, we first process the sentences with Minipar (Lin, 1998), and then convert Minipar's representation into DSyntS. Since user reviews are different from the newspaper articles on which Minipar was trained, the output of Minipar can be inaccurate, leading to failure in conversion. We check whether conversion is successful in the filtering stage.

2.4 Filtering

The goal of filtering is to identify $\mathcal{U}$ that realize the distinguished attributes and to guarantee high precision for the learned mappings. Recall is less important since systems need to convey requested information as accurately as possible. Our procedure for deriving semantic representations is based on the hypothesis that if $\mathcal{U}$ contains named-entities that realize the distinguished attributes, that $\mathcal{R}$ will include the relevant relation in the domain ontology. We also assume that if $\mathcal{U}$ contains named-entities that are not covered by the domain ontology, or words indicating that the meaning of $\mathcal{U}$ depends on the surrounding context, that $\mathcal{R}$ will not completely characterizes the meaning of $\mathcal{U}$, and so $\mathcal{U}$ should be eliminated. We also require an accurate $\mathcal{S}$ for $\mathcal{U}$. Therefore, the filters described below eliminate $\mathcal{U}$ that (1) realize semantic relations not in the ontology; (2) contain words indicating that its meaning depends on the context; (3) contain unknown words; or (4) cannot be parsed accurately.

No Relations Filter: The sentence does not contain any named-entities for the distinguished attributes.

Other Relations Filter: The sentence contains named-entities for food subtypes, person names, country names, dates (e.g., today, tomorrow, Aug. 26th) or prices (e.g., 12 dollars), or POS tag CD for numerals. These indicate relations not in the ontology.

Rating Dist.Attr.	1	2	3	4	5	Total
food	5	8	6	18	57	94
service	15	3	6	17	56	97
atmosphere	0	3	3	8	31	45
value	0	0	1	8	12	21
overall	3	2	5	15	45	70
Total	23	15	21	64	201	327

Table 3: Domain coverage of single scalar-valued relation mappings.

Contextual Filter: The sentence contains *indexicals* such as *I, you, that* or *cohesive* markers of rhetorical relations that connect it to some part of the preceding text, which means that the sentence cannot be interpreted out of context. These include discourse markers, such as list item markers with LS as the POS tag, that signal the organization structure of the text (Hirschberg and Litman, 1987), as well as discourse connectives that signal semantic and pragmatic relations of the sentence with other parts of the text (Knott, 1996), such as coordinating conjunctions at the beginning of the utterance like *and* and *but* etc., and conjunct adverbs such as *however*, *also*, *then*.

Unknown Words Filter: The sentence contains words not in WordNet (Fellbaum, 1998) (which includes typographical errors), or POS tags contain NN (Noun), which may indicate an unknown named-entity, or the sentence has more than a fixed length of words,[2] indicating that its meaning may not be estimated solely by named entities.

Parsing Filter: The sentence fails the parsing to DSyntS conversion. Failures are automatically detected by comparing the original sentence with the one realized by RealPro taking the converted DSyntS as an input.

We apply the filters, in a cascading manner, to the 18,466 sentences with semantic representations. As a result, we obtain 512 (2.8%) mappings of $(\mathcal{U}, \mathcal{R}, \mathcal{S})$. After removing 61 duplicates, 451 distinct (2.4%) mappings remain. Table 2 shows the number of sentences eliminated by each filter.

3 Objective Evaluation

We evaluate the learned expressions with respect to domain coverage, linguistic variation and generativity.

[2] We used 20 as a threshold.

#	Combination of Dist. Attrs	Count
1	food-service	39
2	food-value	21
3	atmosphere-food	14
4	atmosphere-service	10
5	atmosphere-food-service	7
6	food-foodtype	4
7	atmosphere-food-value	4
8	location-overall	3
9	food-foodtype-value	3
10	food-service-value	2
11	food-foodtype-location	2
12	food-overall	2
13	atmosphere-foodtype	2
14	atmosphere-overall	2
15	service-value	1
16	overall-service	1
17	overall-value	1
18	foodtype-overall	1
19	food-foodtype-location-overall	1
20	atmosphere-food-service-value	1
21	atmosphere-food-overall-service-value	1
	Total	122

Table 4: Counts for multi-relation mappings.

3.1 Domain Coverage

To be usable for a dialogue system, the mappings must have good domain coverage. Table 3 shows the distribution of the 327 mappings realizing a single scalar-valued relation, categorized by the associated rating score.[3] For example, there are 57 mappings with $\mathcal{R}$ of ‘RESTAURANT *has foodquality=5,*’ and a large number of mappings for both the foodquality and servicequality relations. Although we could not obtain mappings for some relations such as price={1,2}, coverage for expressing a single relation is fairly complete.

There are also mappings that express several relations. Table 4 shows the counts of mappings for multi-relation mappings, with those containing a food or service relation occurring more frequently as in the single scalar-valued relation mappings. We found only 21 combinations of relations, which is surprising given the large potential number of combinations (There are 50 combinations if we treat relations with different scalar values differently). We also find that most of the mappings have two or three relations, perhaps suggesting that system utterances should not express too many relations in a single sentence.

3.2 Linguistic Variation

We also wish to assess whether the linguistic variation of the learned mappings was greater than what we could easily have generated with a hand-crafted dictionary, or a hand-crafted dictionary augmented with aggregation operators, as in (Walker et al., 2003). Thus, we first categorized the mappings by the patterns of the DSyntSs. Table 5 shows the most common syntactic patterns (more than 10 occurrences), indicating that 30% of the learned patterns consist of the simple form “X is ADJ” where ADJ is an adjective, or “X is RB ADJ,” where RB is a degree modifier. Furthermore, up to 55% of the learned mappings could be generated from these basic patterns by the application of a combination operator that coordinates multiple adjectives, or coordinates predications over distinct attributes. However, there are 137 syntactic patterns in all, 97 with unique syntactic structures and 21 with two occurrences, accounting for 45% of the learned mappings. Table 6 shows examples of learned mappings with distinct syntactic structures. It would be surprising to see this type of variety in a hand-crafted generation dictionary. In addition, the learned mappings contain 275 distinct lexemes, with a minimum of 2, maximum of 15, and mean of 4.63 lexemes per DSyntS, indicating that the method extracts a wide variety of expressions of varying lengths.

Another interesting aspect of the learned mappings is the wide variety of adjectival phrases (APs) in the common patterns. Tables 7 and 8 show the APs in single scalar-valued relation mappings for *food* and *service* categorized by the associated ratings. Tables for *atmosphere, value* and *overall* can be found in the Appendix. Moreover, the meanings for some of the learned APs are very specific to the particular attribute, e.g. *cold* and *burnt* associated with foodquality of 1, *attentive and prompt* for servicequality of 5, *silly and inattentive* for servicequality of 1. and *mellow* for atmosphere of 5. In addition, our method places the adjectival phrases (APs) in the common patterns on a more fine-grained scale of 1 to 5, similar to the strength classifications in (Wilson et al., 2004), in contrast to other automatic methods that classify expressions into a binary *positive* or *negative* polarity (e.g. (Turney, 2002)).

3.3 Generativity

Our motivation for deriving syntactic representations for the learned expressions was the possibility of using an off-the-shelf sentence planner to derive new combinations of relations, and apply aggregation and other syntactic transformations. We examined how many of the learned DSyntSs can be combined with each other, by taking every pair of DSyntSs in the mappings and applying the built-in merge operation in the SPaRKy generator (Walker et al., 2003). We found that only 306 combinations out of a potential 81,318

[3] There are two other single-relation but not scalar-valued mappings that concern LOCATION in our mappings.

#	syntactic pattern	example utterance	count	ratio	accum.
1	NN VB JJ	The atmosphere is wonderful.	92	20.4%	20.4%
2	NN VB RB JJ	The atmosphere was very nice.	52	11.5%	31.9%
3	JJ NN	Bad service.	36	8.0%	39.9%
4	NN VB JJ CC JJ	The food was flavorful but cold.	25	5.5%	45.5%
5	RB JJ NN	Very trendy ambience.	22	4.9%	50.3%
6	NN VB JJ CC NN VB JJ	The food is excellent and the atmosphere is great.	13	2.9%	53.2%
7	NN CC NN VB JJ	The food and service were fantastic.	10	2.2%	55.4%

Table 5: Common syntactic patterns of DSyntSs, flattened to a POS sequence for readability. NN, VB, JJ, RB, CC stand for noun, verb, adjective, adverb, and conjunction, respectively.

[overall=1, value=2] Very disappointing experience for the money charged.
[food=5, value=5] The food is excellent and plentiful at a reasonable price.
[food=5, service=5] The food is exquisite as well as the service and setting.
[food=5, service=5] The food was spectacular and so was the service.
[food=5, foodtype, value=5] Best FOODTYPE food with a great value for money.
[food=5, foodtype, value=5] An absolutely outstanding value with fantastic FOODTYPE food.
[food=5, foodtype, location, overall=5] This is the best place to eat FOODTYPE food in LOCATION.
[food=5, foodtype] Simply amazing FOODTYPE food.
[food=5, foodtype] RESTAURANTNAME is the best of the best for FOODTYPE food.
[food=5] The food is to die for.
[food=5] What incredible food.
[food=4] Very pleasantly surprised by the food.
[food=1] The food has gone downhill.
[atmosphere=5, overall=5] This is a quiet little place with great atmosphere.
[atmosphere=5, food=5, overall=5, service=5, value=5] The food, service and ambience of the place are all fabulous and the prices are downright cheap.

Table 6: Acquired generation patterns (with shorthand for relations in square brackets) whose syntactic patterns occurred only once.

food=1	awful, bad, burnt, cold, very ordinary
food=2	acceptable, bad, flavored, not enough, very bland, very good
food=3	adequate, bland and mediocre, flavorful but cold, pretty good, rather bland, very good
food=4	absolutely wonderful, awesome, decent, excellent, good, good and generous, great, outstanding, rather good, really good, traditional, very fresh and tasty, very good, very very good
food=5	absolutely delicious, absolutely fantastic, absolutely great, absolutely terrific, ample, well seasoned and hot, awesome, best, delectable and plentiful, delicious, delicious but simple, excellent, exquisite, fabulous, fancy but tasty, fantastic, fresh, good, great, hot, incredible, just fantastic, large and satisfying, outstanding, plentiful and outstanding, plentiful and tasty, quick and hot, simply great, so delicious, so very tasty, superb, terrific, tremendous, very good, wonderful

Table 7: Adjectival phrases (APs) in single scalar-valued relation mappings for *foodquality*.

combinations (0.37%) were successful. This is because the merge operation in SPaRKy requires that the subjects and the verbs of the two DSyntSs are identical, e.g. the subject is RESTAURANT and verb is *has*, whereas the learned DSyntSs often place the attribute in subject position as a definite noun phrase. However, the learned DSyntS can be incorporated into SPaRKy using the semantic representations to substitute learned DSyntSs into nodes in the sentence plan tree. Figure 2 shows some example utterances generated by SPaRKy with its original dictionary and example utterances when the learned mappings are incorporated. The resulting utterances seem more natural and colloquial; we examine whether this is true in the next section.

4 Subjective Evaluation

We evaluate the obtained mappings in two respects: the consistency between the automatically derived semantic representation and the realization, and the naturalness of the realization.

For comparison, we used a baseline of handcrafted mappings from (Walker et al., 2003) except that we changed the word *decor* to *atmosphere* and added five mappings for *overall*. For scalar relations, this consists of the realization “RESTAURANT *has* ADJ LEX” where ADJ is *mediocre, decent, good, very good*, or *excellent* for rating values 1-5, and LEX is *food quality, service, atmosphere, value*, or *overall* depending on the relation. RESTAURANT is filled with the name of a restaurant at runtime. For example, ‘RESTAURANT *has foodquality=1’* is realized as “RESTAURANT *has mediocre food quality.”* The location and food type relations are mapped to “RESTAURANT *is located in* LOCATION” and “RESTAURANT *is a* FOODTYPE *restaurant.”*

The learned mappings include 23 distinct semantic representations for a single-relation (22 for scalar-valued relations and one for location) and 50 for multi-relations. Therefore, using the handcrafted mappings, we first created 23 utterances for the single-relations. We then created three utterances for each of 50 multi-relations using different clause-combining operations from (Walker et al., 2003). This gave a total of 173 baseline utterances, which together with 451 learned mappings,

service=1	awful, bad, great, horrendous, horrible, inattentive, forgetful and slow, marginal, really slow, silly and inattentive, still marginal, terrible, young
service=2	overly slow, very slow and inattentive
service=3	bad, bland and mediocre, friendly and knowledgeable, good, pleasant, prompt, very friendly
service=4	all very warm and welcoming, attentive, extremely friendly and good, extremely pleasant, fantastic, friendly, friendly and helpful, good, great, great and courteous, prompt and friendly, really friendly, so nice, swift and friendly, very friendly, very friendly and accommodating
service=5	all courteous, excellent, excellent and friendly, extremely friendly, fabulous, fantastic, friendly, friendly and helpful, friendly and very attentive, good, great, great, prompt and courteous, happy and friendly, impeccable, intrusive, legendary, outstanding, pleasant, polite, attentive and prompt, prompt and courteous, prompt and pleasant, quick and cheerful, stupendous, superb, the most attentive, unbelievable, very attentive, very congenial, very courteous, very friendly, very friendly and helpful, very friendly and pleasant, very friendly and totally personal, very friendly and welcoming, very good, very helpful, very timely, warm and friendly, wonderful

Table 8: Adjectival phrases (APs) in single scalar-valued relation mappings for *servicequality*.

yielded 624 utterances for evaluation.

Ten subjects, all native English speakers, evaluated the mappings by reading them from a webpage. For each system utterance, the subjects were asked to express their degree of agreement, on a scale of 1 (lowest) to 5 (highest), with the statement (a) *The meaning of the utterance is consistent with the ratings expressing their semantics*, and with the statement (b) *The style of the utterance is very natural and colloquial.* They were asked not to correct their decisions and also to rate each utterance on its own merit.

4.1 Results

Table 9 shows the means and standard deviations of the scores for baseline vs. learned utterances for consistency and naturalness. A t-test shows that the consistency of the learned expression is significantly lower than the baseline (df=4712, $p < .001$) but that their naturalness is significantly higher than the baseline (df=3107, $p < .001$). However, consistency is still high. Only 14 of the learned utterances (shown in Tab. 10) have a mean consistency score lower than 3, which indicates that, by and large, the human judges felt that the inferred semantic representations were consistent with the meaning of the learned expressions. The correlation coefficient between consistency and naturalness scores is 0.42, which indicates that consistency does not greatly relate to naturalness.

Original SPaRKy utterances

- Babbo has the best overall quality among the selected restaurants with excellent decor, excellent service and superb food quality.
- Babbo has excellent decor and superb food quality with excellent service. It has the best overall quality among the selected restaurants.

↓

Combination of SPaRKy and learned DSyntS

- Because **the food is excellent, the wait staff is professional and the decor is beautiful and very comfortable**, Babbo has the best overall quality among the selected restaurants.
- Babbo has the best overall quality among the selected restaurants because **atmosphere is exceptionally nice, food is excellent and the service is superb**.
- Babbo has superb food quality, **the service is exceptional and the atmosphere is very creative**. It has the best overall quality among the selected restaurants.

Figure 2: Utterances incorporating learned DSyntSs (Bold font) in SPaRKy.

	baseline		learned		stat.
	mean	sd.	mean	sd.	sig.
Consistency	**4.714**	0.588	**4.459**	0.890	+
Naturalness	**4.227**	0.852	**4.613**	0.844	+

Table 9: Consistency and naturalness scores averaged over 10 subjects.

We also performed an ANOVA (ANalysis Of VAriance) of the effect of each relation in $\mathcal{R}$ on naturalness and consistency. There were no significant effects except that mappings combining food, service, and atmosphere were significantly worse (df=1, F=7.79, p=0.005). However, there is a trend for mappings to be rated higher for the *food* attribute (df=1, F=3.14, p=0.08) and the *value* attribute (df=1, F=3.55, p=0.06) for consistency, suggesting that perhaps it is easier to learn some mappings than others.

5 Related Work

Automatically finding sentences with the same meaning has been extensively studied in the field of automatic paraphrasing using parallel corpora and corpora with multiple descriptions of the same events (Barzilay and McKeown, 2001; Barzilay and Lee, 2003). Other work finds predicates of similar meanings by using the similarity of contexts around the predicates (Lin and Pantel, 2001). However, these studies find a set of sentences with the same meaning, but do not associate a specific meaning with the sentences. One exception is (Barzilay and Lee, 2002), which derives mappings between semantic representations and realizations using a parallel (but unaligned) corpus consisting of both complex semantic input and corresponding natural language verbalizations for mathemat-

shorthand for relations and utterance	score
[food=4] The food is delicious and beautifully prepared.	2.9
[overall=4] A wonderful experience.	2.9
[service=3] The service is bland and mediocre.	2.8
[atmosphere=2] The atmosphere here is eclectic.	2.6
[overall=3] Really fancy place.	2.6
[food=3, service=4] Wonderful service and great food.	2.5
[service=4] The service is fantastic.	2.5
[overall=2] The RESTAURANTNAME is once a great place to go and socialize.	2.2
[atmosphere=2] The atmosphere is unique and pleasant.	2.0
[food=5, foodtype] FOODTYPE and FOODTYPE food.	1.8
[service=3] Waitstaff is friendly and knowledgeable.	1.7
[atmosphere=5, food=5, service=5] The atmosphere, food and service.	1.6
[overall=3] Overall, a great experience.	1.4
[service=1] The waiter is great.	1.4

Table 10: The 14 utterances with consistency scores below 3.

ical proofs. However, our technique does not require parallel corpora or previously existing semantic transcripts or labeling, and user reviews are widely available in many different domains (See *http://www.epinions.com/*).

There is also significant previous work on mining user reviews. For example, Hu and Liu (2005) use reviews to find adjectives to describe products, and Popescu and Etzioni (2005) automatically find features of a product together with the polarity of adjectives used to describe them. They both aim at summarizing reviews so that users can make decisions easily. Our method is also capable of finding polarities of modifying expressions including adjectives, but on a more fine-grained scale of 1 to 5. However, it might be possible to use their approach to create rating information for raw review texts as in (Pang and Lee, 2005), so that we can create mappings from reviews without ratings.

6 Summary and Future Work

We proposed automatically obtaining mappings between semantic representations and realizations from reviews with individual ratings. The results show that: (1) the learned mappings provide good coverage of the domain ontology and exhibit good linguistic variation; (2) the consistency between the semantic representations and realizations is high; and (3) the naturalness of the realizations are significantly higher than the baseline.

There are also limitations in our method. Even though consistency is rated highly by human subjects, this may actually be a judgement of whether the polarity of the learned mapping is correctly placed on the 1 to 5 rating scale. Thus, alternate ways of expressing, for example *foodquality*=5, shown in Table 7, cannot be guaranteed to be synonymous, which may be required for use in spoken language generation. Rather, an examination of the adjectival phrases in Table 7 shows that different aspects of the food are discussed. For example *ample* and *plentiful* refer to the portion size, *fancy* may refer to the presentation, and *delicious* describes the flavors. This suggests that perhaps the ontology would benefit from representing these sub-attributes of the food attribute, and sub-attributes in general. Another problem with consistency is that the same AP, e.g. *very good* in Table 7 may appear with multiple ratings. For example, *very good* is used for every foodquality rating from 2 to 5. Thus some further automatic or by-hand analysis is required to refine what is learned before actual use in spoken language generation. Still, our method could reduce the amount of time a system designer spends developing the spoken language generator, and increase the naturalness of spoken language generation.

Another issue is that the recall appears to be quite low given that all of the sentences concern the same domain: only 2.4% of the sentences could be used to create the mappings. One way to increase recall might be to automatically augment the list of distinguished attribute lexicalizations, using WordNet or work on automatic identification of synonyms, such as (Lin and Pantel, 2001). However, the method here has high precision, and automatic techniques may introduce noise. A related issue is that the filters are in some cases too strict. For example the contextual filter is based on POS-tags, so that sentences that do not require the prior context for their interpretation are eliminated, such as sentences containing subordinating conjunctions like *because*, *when*, *if*, whose arguments are both given in the same sentence (Prasad et al., 2005). In addition, recall is affected by the domain ontology, and the automatically constructed domain ontology from the review webpages may not cover all of the domain. In some review domains, the attributes that get individual ratings are a limited subset of the domain ontology. Techniques for automatic feature identification (Hu and Liu, 2005; Popescu and Etzioni, 2005) could possibly help here, although these techniques currently have the limitation that they do not automatically identify different lexicalizations of the same feature.

A different type of limitation is that dialogue systems need to generate utterances for information gathering whereas the mappings we obtained can only be used for information presentation. Thus these would have to be constructed by hand, as in current practice, or perhaps other types of corpora or resources could be utilized. In addition, the utility of syntactic structures in the mappings should be further examined, especially given the failures in DSyntS conversion. An alternative would be to leave some sentences unparsed and use them as templates with hybrid generation techniques (White and Caldwell, 1998). Finally, while we believe that this technique will apply across domains, it would be useful to test it on domains such as movie reviews or product reviews, which have more complex domain ontologies.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported by a Royal Society Wolfson award to Marilyn Walker and a research collaboration grant from NTT to the Cognitive Systems Group at the University of Sheffield.

References

Regina Barzilay and Lillian Lee. 2002. Bootstrapping lexical choice via multiple-sequence alignment. In *Proc. EMNLP*, pages 164–171.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proc. HLT/NAACL*, pages 16–23.

Regina Barzilay and Kathleen McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proc. 39th ACL*, pages 50–57.

Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proc. 40th ACL*.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. The MIT Press.

Julia Hirschberg and Diane. J. Litman. 1987. Now let's talk about NOW: Identifying cue phrases intonationally. In *Proc. 25th ACL*, pages 163–171.

Minqing Hu and Bing Liu. 2005. Mining and summarizing customer reviews. In *Proc. KDD*, pages 168–177.

Alistair Knott. 1996. *A Data-Driven Methodology for Motivating a Set of Coherence Relations*. Ph.D. thesis, University of Edinburgh, Edinburgh.

Benoit Lavoie and Owen Rambow. 1997. A fast and portable realizer for text generation systems. In *Proc. 5th Applied NLP*, pages 265–268.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7(4):343–360.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*.

Johanna D. Moore, Mary Ellen Foster, Oliver Lemon, and Michael White. 2004. Generating tailored, comparative descriptions in spoken dialogue. In *Proc. 7th FLAIR*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proc. 43st ACL*, pages 115–124.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proc. HLT/EMNLP*, pages 339–346.

Rashmi Prasad, Aravind Joshi, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, and Bonnie Webber. 2005. The Penn Discourse TreeBank as a resource for natural language generation. In *Proc. Corpus Linguistics Workshop on Using Corpora for NLG*.

Stephanie Seneff and Joseph Polifroni. 2000. Formal and natural language generation in the mercury conversational system. In *Proc. ICSLP*, volume 2, pages 767–770.

Mariët Theune. 2003. From monologue to dialogue: natural language generation in OVIS. In *AAAI 2003 Spring Symposium on Natural Language Generation in Written and Spoken Dialogue*, pages 141–150.

Peter D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proc. 40th ACL*, pages 417–424.

Marilyn Walker, Rashmi Prasad, and Amanda Stent. 2003. A trainable generator for recommendations in multimodal dialog. In *Proc. Eurospeech*, pages 1697–1700.

Michael White and Ted Caldwell. 1998. EXEMPLARS: A practical, extensible framework for dynamic text generation. In *Proc. INLG*, pages 266–275.

Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. 2004. Just how mad are you? finding strong and weak opinion clauses. In *Proc. AAAI*, pages 761–769.

Appendix

Adjectival phrases (APs) in single scalar-valued relation mappings for *atmosphere, value*, and *overall*.

atmosphere=2	eclectic, unique and pleasant
atmosphere=3	busy, pleasant but extremely hot
atmosphere=4	fantastic, great, quite nice and simple, typical, very casual, very trendy, wonderful
atmosphere=5	beautiful, comfortable, excellent, great, interior, lovely, mellow, nice, nice and comfortable, phenomenal, pleasant, quite pleasant, unbelievably beautiful, very comfortable, very cozy, very friendly, very intimate, very nice, very nice and relaxing, very pleasant, very relaxing, warm and contemporary, warm and very comfortable, wonderful

value=3	very reasonable
value=4	great, pretty good, reasonable, very good
value=5	best, extremely reasonable, good, great, reasonable, totally reasonable, very good, very reasonable

overall=1	just bad, nice, thoroughly humiliating
overall=2	great, really bad
overall=3	bad, decent, great, interesting, really fancy
overall=4	excellent, good, great, just great, never busy, not very busy, outstanding, recommended, wonderful
overall=5	amazing, awesome, capacious, delightful, extremely pleasant, fantastic, good, great, local, marvelous, neat, new, overall, overwhelmingly pleasant, pampering, peaceful but idyllic, really cool, really great, really neat, really nice, special, tasty, truly great, ultimate, unique and enjoyable, very enjoyable, very excellent, very good, very nice, very wonderful, warm and friendly, wonderful

Measuring Language Divergence by Intra-Lexical Comparison

T. Mark Ellison
Informatics
University of Edinburgh
mark@markellison.net

Simon Kirby
Language Evolution and Computation Research Unit
Philosophy, Psychology and Language Sciences,
University of Edinburgh
simon@ling.ed.ac.uk

Abstract

This paper presents a method for building genetic language taxonomies based on a new approach to comparing lexical forms. Instead of comparing forms cross-linguistically, a matrix of language-internal similarities between forms is calculated. These matrices are then compared to give distances between languages. We argue that this coheres better with current thinking in linguistics and psycholinguistics. An implementation of this approach, called PHILOLOGICON, is described, along with its application to Dyen et al.'s (1992) ninety-five wordlists from Indo-European languages.

1 Introduction

Recently, there has been burgeoning interest in the computational construction of genetic language taxonomies (Dyen et al., 1992; Nerbonne and Heeringa, 1997; Kondrak, 2002; Ringe et al., 2002; Benedetto et al., 2002; McMahon and McMahon, 2003; Gray and Atkinson, 2003; Nakleh et al., 2005).

One common approach to building language taxonomies is to ascribe language-language distances, and then use a generic algorithm to construct a tree which explains these distances as much as possible. Two questions arise with this approach. The first asks what aspects of languages are important in measuring inter-language distance. The second asks how to measure distance given these aspects.

A more traditional approach to building language taxonomies (Dyen et al., 1992) answers these questions in terms of **cognates**. A word in language A is said to be cognate with word in language B if the forms shared a common ancestor in the parent language of A and B. In the cognate-counting method, inter-language distance depends on the lexical forms of the languages. The distance between two languages is a function of the number or fraction of these forms which are cognate between the two languages[1]. This approach to building language taxonomies is hard to implement in toto because constructing ancestor forms is not easily automatable.

More recent approaches, such as Kondrak's (2002) and Heggarty et al's (2005) work on dialect comparison, take the synchronic word forms themselves as the language aspect to be compared. Variations on edit distance (see Kessler (2005) for a survey) are then used to evaluate differences between languages for each word, and these differences are aggregated to give a distance between languages or dialects as a whole. This approach is largely automatable, although some methods do require human intervention.

In this paper, we present novel answers to the two questions. The features of language we will compare are not sets of words or phonological forms. Instead we compare the similarities between forms, expressed as confusion probabilities. The distribution of confusion probabilities in one language is called a **lexical metric**. Section 2 presents the definition of lexical metrics and some arguments for their being good language representatives for the purposes of comparison.

The distance between two languages is the divergence their lexical metrics. In section 3, we detail two methods for measuring this divergence:

[1] McMahon and McMahon (2003) for an account of tree-inference from the cognate percentages in the Dyen et al. (1992) data.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 273–280,
Sydney, July 2006. ©2006 Association for Computational Linguistics

Kullback-Liebler (herafter KL) divergence and Rao distance. The subsequent section (4) describes the application of our approach to automatically constructing a taxonomy of Indo-European languages from Dyen et al. (1992) data.

Section 5 suggests how lexical metrics can help identify cognates. The final section (6) presents our conclusions, and discusses possible future directions for this work.

Versions of the software and data files described in the paper will be made available to coincide with its publication.

2 Lexical Metric

The first question posed by the distance-based approach to genetic language taxonomy is: what should we compare?

In some approaches (Kondrak, 2002; McMahon et al., 2005; Heggarty et al., 2005; Nerbonne and Heeringa, 1997), the answer to this question is that we should compare the phonetic or phonological realisations of a particular set of meanings across the range of languages being studied. There are a number of problems with using lexical forms in this way.

Firstly, in order to compare forms from different languages, we need to embed them in common phonetic space. This phonetic space provides granularity, marking two phones as identical or distinct, and where there is a graded measure of phonetic distinction it measures this.

There is growing doubt in the field of phonology and phonetics about the meaningfulness of assuming of a common phonetic space. Port and Leary (2005) argue convincingly that this assumption, while having played a fundamental role in much recent linguistic theorising, is nevertheless unfounded. The degree of difference between sounds, and consequently, the degree of phonetic difference between words can only be ascertained within the context of a single language.

It may be argued that a common phonetic space can be found in either acoustics or degrees of freedom in the speech articulators. Language-specific categorisation of sound, however, often restructures this space, sometimes with distinct sounds being treated as homophones. One example of this is the realisation of orthographic **rr** in European Portuguese: it is indifferently realised with an apical or a uvular trill, different sounds made at distinct points of articulation.

If there is no language-independent, common phonetic space with an equally common similarity measure, there can be no principled approach to comparing forms in one language with those of another.

In contrast, language-specific word-similarity is well-founded. A number of psycholinguistic models of spoken word recognition (Luce et al., 1990) are based on the idea of lexical neighbourhoods. When a word is accessed during processing, the other words that are phonemically or orthographically similar are also activated. This effect can be detected using experimental paradigms such as priming.

Our approach, therefore, is to abandon the cross-linguistic comparison of phonetic realisations, in favour of language-internal comparison of forms. (See also work by Shillcock et al. (2001) and Tamariz (2005)).

2.1 Confusion probabilities

One psychologically well-grounded way of describing the similarity of words is in terms of their **confusion probabilities**. Two words have high confusion probability if it is likely that one word could be produced or understood when the other was intended. This type of confusion can be measured experimentally by giving subjects words in noisy environments and measuring what they apprehend.

A less pathological way in which confusion probability is realised is in coactivation. If a person hears a word, then they more easily and more quickly recognise similar words. This coactivation occurs because the phonological realisation of words is not completely separate in the mind. Instead, realisations are interdependent with realisations of similar words.

We propose that confusion probabilities are ideal information to constitute the lexical metric. They are language-specific, psychologically grounded, can be determined by experiment, and integrate with existing psycholinguistic models of word recognition.

2.2 NAM and beyond

Unfortunately, experimentally determined confusion probabilities for a large number of languages are not available. Fortunately, models of spoken word recognition allow us to predict these probabilities from easily-computable measures of word similarity.

For example, the **neighbourhood activation model** (**NAM**) (Luce et al., 1990; Luce and Pisoni, 1998) predicts confusion probabilities from the relative frequency of words in the neighbourhood of the target. Words are *in the neighbourhood* of the target if their Levenstein (1965) edit distance from the target is one. The more frequent the word is, the greater its likelihood of replacing the target.

Bailey and Hahn (2001) argue, however, that the all-or-nothing nature of the lexical neighbourhood is insufficient. Instead word similarity is the complex function of frequency and phonetic similarity shown in equation (1). Here A, B, C and D are constants of the model, u and v are words, and d is a phonetic similarity model.

$$s = (AF(u)^2 + BF(u) + C)e^{-D.d(u,v)} \qquad (1)$$

We have adapted this model slightly, in line with NAM, taking the similarity s to be the probability of confusing stimulus v with form u. Also, as our data usually offers no frequency information, we have adopted the maximum entropy assumption, namely, that all relative frequencies are equal. Consequently, the probability of confusion of two words depends solely on their similarity distance. While this assumption degrades the psychological reality of the model, it does not render it useless, as the similarity measure continues to provide important distinctions in neighbourhood confusability.

We also assume for simplicity, that the constant D has the value 1.

With these simplifications, equation (2) shows the probability of apprehending word w, out of a set W of possible alternatives, given a stimulus word w_s.

$$P(w|w_s) = e^{-d(w,w_s)}/N(w_s) \qquad (2)$$

The normalising constant $N(s)$ is the sum of the non-normalised values for $e^{-d(w,w_s)}$ for all words w.

$$N(w_s) = \sum_{w \in W} e^{-d(u,v)}$$

2.3 Scaled edit distances

Kidd and Watson (1992) have shown that discriminability of frequency and of duration of tones in a tone sequence depends on its length as a proportion of the length of the sequence. Kapatsinski (2006) uses this, with other evidence, to argue that word recognition edit distances must be scaled by word-length.

There are other reasons for coming to the same conclusion. The simple Levenstein distance exaggerates the disparity between long words in comparison with short words. A word of consisting of 10 symbols, purely by virtue of its length, will on average be marked as more different from other words than a word of length two. For example, Levenstein distance between **interested** and **rest** is six, the same as the distance between **rest** and **by**, even though the latter two have nothing in common. As a consequence, close phonetic transcriptions, which by their very nature are likely to involve more symbols per word, will result in larger edit distances than broad phonemic transcriptions of the same data.

To alleviate this problem, we define a new edit distance function d_2 which scales Levenstein distances by the average length of the words being compared (see equation 3). Now the distance between **interested** and **rest** is 0.86, while that between **rest** and **by** is 2.0, reflecting the greater relative difference in the second pair.

$$d_2(w_2, w_1) = \frac{2d(w_2, w_1)}{|w_1| + |w_2|} \qquad (3)$$

Note that by scaling the raw edit distance with the average lengths of the words, we are preserving the symmetric property of the distance measure.

There are other methods of comparing strings, for example string kernels (Shawe-Taylor and Cristianini, 2004), but using Levenstein distance keeps us coherent with the psycholinguistic accounts of word similarity.

2.4 Lexical Metric

Bringing this all together, we can define the lexical metric.

A lexicon L is a mapping from a set of meanings M, such as “DOG”, “TO RUN”, “GREEN”, etc., onto a set F of forms such as /pies/, /biec/, /zielony/.

The confusion probability P of m_1 for m_2 in lexical L is the normalised negative exponential of the scaled edit-distance of the corresponding forms. It is worth noting that when frequencies are assumed to follow the maximum entropy distribution, this connection between confusion probabilities and distances (see equation 4) is the same as that proposed by Shepard (1987).

$$P(m_1|m_2;L) = \frac{e^{-d_2(L(m_1),L(m_2))}}{N(m_2;L)} \quad (4)$$

A lexical metric of L is the mapping $LM(L) : M^2 \to [0,1]$ which assigns to each pair of meanings m_1, m_2 the probability of confusing m_1 for m_2, scaled by the frequency of m_2.

$$\begin{aligned} & LM(L)(m_1,m_2) \\ = \; & P(L(m_1)|L(m_2))P(m_2) \\ = \; & \frac{e^{-d_2(L(m_1),L(m_2))}}{N(m_2;L)|M|} \end{aligned}$$

where $N(m_2;L)$ is the normalising function defined in equation (5).

$$N(m_2;L) = \sum_{m \in M} e^{-d_2(L(m),L(m_2))} \quad (5)$$

Table 1 shows a minimal lexicon consisting only of the numbers one to five, and a corresponding lexical metric. The values in the lexical metric are inferred word confusion probabilities. The matrix is normalised so that the sum of each row is 0.2, ie. one-fifth for each of the five words, so the total of the matrix is one. Note that the diagonal values vary because the off-diagonal values in each row vary, and consequently, so does the normalisation for the row.

	one	two	three	four	five
one	0.102	0.027	0.023	0.024	0.024
two	0.028	0.107	0.024	0.026	0.015
three	0.024	0.024	0.107	0.023	0.023
four	0.025	0.025	0.022	0.104	0.023
five	0.026	0.015	0.023	0.025	0.111

Table 1: A lexical metric on a mini-lexicon consisting of the numbers one to five.

3 Language-Language Distance

In the previous section, we introduced the lexical metric as the key measurable for comparing languages. Since lexical metrics are probability distributions, comparison of metrics means measuring the difference between probability distributions. To do this, we use two measures: the symmetric Kullback-Liebler divergence (Jeffreys, 1946) and the Rao distance (Rao, 1949; Atkinson and Mitchell, 1981; Micchelli and Noakes, 2005) based on Fisher Information (Fisher, 1959). These can be defined in terms the **geometric path** from one distribution to another.

3.1 Geometric paths

The geometric path between two distributions P and Q is a conditional distribution R with a continuous parameter α such that at $\alpha = 0$, the distribution is P, and at $\alpha = 1$ it is Q. This conditional distribution is called the **geometric** because it consists of normalised weighted geometric means of the two defining distributions (equation 6).

$$R(\bar{w}|\alpha) = P(\bar{w})^{\alpha} Q(\bar{w})^{1-\alpha} / k(\alpha; P, Q) \quad (6)$$

The function $k(\alpha; P, Q)$ is a normaliser for the conditional distribution, being the sum of the weighted geometric means of values from P and Q (equation 7). This value is known as the Chernoff coefficient or Helliger path (Basseville, 1989). For brevity, the P, Q arguments to k will be treated as implicit and not expressed in equations.

$$k(\alpha) = \sum_{\bar{w} \in W^2} P(\bar{w})^{1-\alpha} Q(\bar{w})^{\alpha} \quad (7)$$

3.2 Kullback-Liebler distance

The first-order (equation 8) differential of the normaliser with regard to α is of particular interest.

$$k'(\alpha) = \sum_{\bar{w} \in W^2} \log \frac{Q(\bar{w})}{P(\bar{w})} P(\bar{w})^{1-\alpha} Q(\bar{w})^{\alpha} \quad (8)$$

At $\alpha = 0$, this value is the negative of the Kullback-Liebler distance $KL(P|Q)$ of Q with regard to P (Basseville, 1989). At $\alpha = 1$, it is the Kullback-Liebler distance $KL(Q|P)$ of P with regard to Q. Jeffreys' (1946) measure is a symmetrisation of KL distance, by averaging the commutations (equations 9,10).

$$\begin{aligned} KL(P,Q) &= \frac{KL(Q|P) + KL(P|Q)}{2} \quad (9) \\ &= \frac{k'(1) - k'(0)}{2} \quad (10) \end{aligned}$$

3.3 Rao distance

Rao distance depends on the second-order (equation 11) differential of the normaliser with regard to α.

$$k''(\alpha) = \sum_{\bar{w} \in W^2} \log^2 \frac{Q(\bar{w})}{P(\bar{w})} P(\bar{w})^{1-\alpha} Q(\bar{w})^{\alpha} \quad (11)$$

Fisher information is defined as in equation (12).

$$FI(P,x) = -\int \frac{\partial^2 \log P(y|x)}{\partial x^2} P(y|x) dy \quad (12)$$

Equation (13) expresses Fisher information along the path R from P to Q at point α using k and its first two derivatives.

$$\mathrm{FI}(R,\alpha) = \frac{k(\alpha)k''(\alpha) - k'(\alpha)^2}{k(\alpha)^2} \qquad (13)$$

The Rao distance $r(P,Q)$ along R can be approximated by the square root of the Fisher information at the path's midpoint $\alpha = 0.5$.

$$\mathrm{r}(P,Q) = \sqrt{\frac{k(0.5)k''(0.5) - k'(0.5)^2}{k(0.5)^2}} \qquad (14)$$

3.4 The PHILOLOGICON algorithm

Bringing these pieces together, the PHILOLOGICON algorithm for measuring the divergence between two languages has the following steps:

1. determine their joint confusion probability matrices, P and Q,

2. substitute these into equation (7), equation (8) and equation (11) to calculate $k(0)$, $k(0.5)$, $k(1)$, $k'(0.5)$, and $k''(0.5)$,

3. and put these into equation (10) and equation (14) to calculate the KL and Rao distances between between the languages.

4 Indo-European

The ideal data for reconstructing Indo-European would be an accurate phonemic transcription of words used to express specifically defined meanings. Sadly, this kind of data is not readily available. However, as a stop-gap measure, we can adopt the data that Dyen et al. collected to construct a Indo-European taxonomy using the cognate method.

4.1 Dyen et al's data

Dyen et al. (1992) collected 95 data sets, each pairing a meaning from a Swadesh (1952)-like 200-word list with its expression in the corresponding language. The compilers annotated with data with cognacy relations, as part of their own taxonomic analysis of Indo-European.

There are problems with using Dyen's data for the purposes of the current paper. Firstly, the word forms collected are not phonetic, phonological or even full orthographic representations. As the authors state, the forms are expressed in sufficient detail to allow an interested reader acquainted with the language in question to identify which word is being expressed.

Secondly, many meanings offer alternative forms, presumably corresponding to synonyms. For a human analyst using the cognate approach, this means that a language can participate in two (or more) word-derivation systems. In preparing this data for processing, we have consistently chosen the first of any alternatives.

A further difficulty lies in the fact that many languages are not represented by the full 200 meanings. Consequently, in comparing lexical metrics from two data sets, we frequently need to restrict the metrics to only those meanings expressed in both the sets. This means that the KL divergence or the Rao distance between two languages were measured on lexical metrics cropped and rescaled to the meanings common to both data-sets. In most cases, this was still more than 190 words.

Despite these mismatches between Dyen et al.'s data and our needs, it provides an testbed for the PHILOLOGICON algorithm. Our reasoning being, that if successful with this data, the method is reasonably reliable. Data was extracted to language-specific files, and preprocessed to clean up problems such as those described above. An additional data-set was added with random data to act as an outlier to root the tree.

4.2 Processing the data

PHILOLOGICON software was then used to calculate the lexical metrics corresponding to the individual data files and to measure KL divergences and Rao distances between them. The program NEIGHBOR from the PHYLIP[2] package was used to construct trees from the results.

4.3 The results

The tree based on Rao distances is shown in figure 1. The discussion follows this tree except in those few cases mentioning differences in the KL tree.

The standard against which we measure the success of our trees is the conservative traditional taxonomy to be found in the Ethnologue (Grimes and Grimes, 2000). The fit with this taxonomy was so good that we have labelled the major branches with their traditional names: Celtic, Germanic, etc. In fact, in most cases, the branch-internal divisions — eg. Brythonic/Goidelic in Celtic, Western/Eastern/Southern in Slavic, or

[2] See **http://evolution.genetics.washington.edu/phylip.html**.

Western/Northern in Germanic — also accord. Note that PHILOLOGICON even groups Baltic and Slavic together into a super-branch Balto-Slavic.

Where languages are clearly out of place in comparison to the traditional taxonomy, these are highlighted: visually in the tree, and verbally in the following text. In almost every case, there are obvious contact phenomena which explain the deviation from the standard taxonomy.

Armenian was grouped with the Indo-Iranian languages. Interestingly, Armenian was at first thought to be an Iranian language, as it shares much vocabulary with these languages. The common vocabulary is now thought to be the result of borrowing, rather than common genetic origin. In the KL tree, Armenian is placed outside of the Indo-Iranian languages, except for Gypsy. On the other hand, in this tree, Ossetic is placed as an outlier of the Indian group, while its traditional classification (and the Rao distance tree) puts it among the Iranian languages. Gypsy is an Indian language, related to Hindi. It has, however, been surrounded by European languages for some centuries. The effects of this influence is the likely cause for it being classified as an outlier in the Indo-Iranian family. A similar situation exists for Slavic: one of the two lists that Dyen et al. offer for Slovenian is classed as an outlier in Slavic, rather than classifying it with the Southern Slavic languages. The other Slovenian list is classified correctly with Serbocroatian. It is possible that the significant impact of Italian on Slovenian has made it an outlier. In Germanic, it is English that is the outlier. This may be due to the impact of the English creole, Takitaki, on the hierarchy. This language is closest to English, but is very distinct from the rest of the Germanic languages. Another misclassification also is the result of contact phenomena. According to the Ethnologue, Sardinian is Southern Romance, a separate branch from Italian or from Spanish. However, its constant contact with Italian has influenced the language such that it is classified here with Italian. We can offer no explanation for why Wakhi ends up an outlier to all the groups.

In conclusion, despite the noisy state of Dyen et al.'s data (for our purposes), the PHILOLOGICON generates a taxonomy close to that constructed using the traditional methods of historical linguistics. Where it deviates, the deviation usually points to identifiable contact between languages.

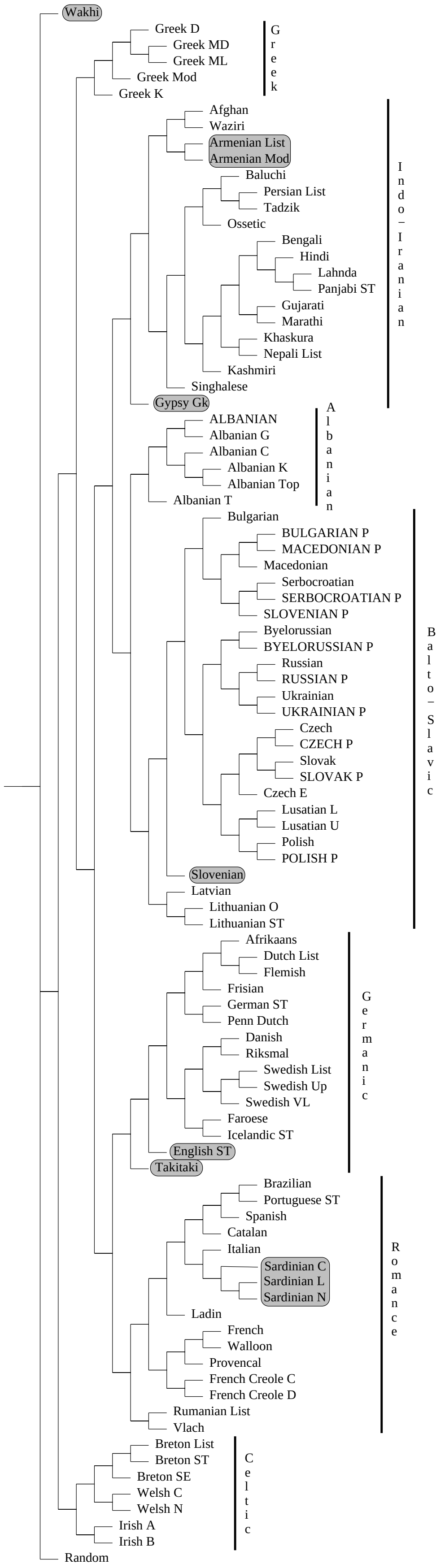


Figure 1: Taxonomy of 95 Indo-European data sets and artificial outlier using PHILOLOGICON and PHYLIP

5 Reconstruction and Cognacy

Subsection 3.1 described the construction of geometric paths from one lexical metric to another. This section describes how the synthetic lexical metric at the midpoint of the path can indicate which words are cognate between the two languages.

The synthetic lexical metric (equation 15) applies the formula for the geometric path equation (6) to the lexical metrics equation (5) of the languages being compared, at the midpoint $\alpha = 0.5$.

$$R_{\frac{1}{2}}(m_1, m_2) = \frac{\sqrt{P(m_1|m_2)Q(m_1|m_2)}}{|M|k(\frac{1}{2})} \qquad (15)$$

If the words for m_1 and m_2 in both languages have common origins in a parent language, then it is reasonable to expect that their confusion probabilities in both languages will be similar. Of course different cognate pairs m_1, m_2 will have differing values for R, but the confusion probabilities in P and Q will be similar, and consequently, the reinforce the variance.

If either m_1 or m_2, or both, is non-cognate, that is, has been replaced by another arbitrary form at some point in the history of either language, then the P and Q for this pair will take independently varying values. Consequently, the geometric mean of these values is likely to take a value more closely bound to the average, than in the purely cognate case.

Thus rows in the lexical metric with wider dynamic ranges are likely to correspond to cognate words. Rows corresponding to non-cognates are likely to have smaller dynamic ranges. The dynamic range can be measured by taking the Shannon information of the probabilities in the row.

Table 2 shows the most low- and high-information rows from English and Swedish (Dyen et al's (1992) data). At the extremes of low and high information, the words are invariably cognate and non-cognate. Between these extremes, the division is not so clear cut, due to chance effects in the data.

English	Swedish	$10^4(h - \bar{h})$
	Low Information	
we	vi	-1.30
here	her	-1.19
to sit	sitta	-1.14
to flow	flyta	-1.04
wide	vid	-0.97
	:	
scratch	klosa	0.78
dirty	smutsig	0.79
left (hand)	vanster	0.84
because	emedan	0.89
	High Information	

Table 2: Shannon information of confusion distributions in the reconstruction of English and Swedish. Information levels are shown translated so that the average is zero.

6 Conclusions and Future Directions

In this paper, we have presented a distance-based method, called PHILOLOGICON, that constructs genetic trees on the basis of lexica from each language. The method only compares words language-internally, where comparison seems both psychologically real and reliable, and never cross-linguistically, where comparison is less well-founded. It uses measures founded in information theory to compare the intra-lexical differences.

The method successfully, if not perfectly, recreated the phylogenetic tree of Indo-European languages on the basis of noisy data. In further work, we plan to improve both the quantity and the quality of the data. Since most of the mis-placements on the tree could be accounted for by contact phenomena, it is possible that a network-drawing, rather than tree-drawing, analysis would produce better results.

Likewise, we plan to develop the method for identifying cognates. The key improvement needed is a way to distinguish indeterminate distances in reconstructed lexical metrics from determinate but uniform ones. This may be achieved by retaining information about the distribution of the original values which were combined to form the reconstructed metric.

References

C. Atkinson and A.F.S. Mitchell. 1981. Rao's distance measure. *Sankhyā*, 4:345–365.

Todd M. Bailey and Ulrike Hahn. 2001. Determinants of wordlikeness: Phonotactics or lexical neighborhoods? *Journal of Memory and Language*, 44:568–591.

Michle Basseville. 1989. Distance measures for signal processing and pattern recognition. *Signal Processing*, 18(4):349–369, December.

D. Benedetto, E. Caglioti, and V. Loreto. 2002. Language trees and zipping. *Physical Review Letters*, 88.

Isidore Dyen, Joseph B. Kruskal, and Paul Black. 1992. An indo-european classification: a lexicostatistical experiment. *Transactions of the American Philosophical Society*, 82(5).

R.A. Fisher. 1959. *Statistical Methods and Scientific Inference*. Oliver and Boyd, London.

Russell D. Gray and Quentin D. Atkinson. 2003. Language-tree divergence times support the anatolian theory of indo-european origin. *Nature*, 426:435–439.

B.F. Grimes and J.E. Grimes, editors. 2000. *Ethnologue: Languages of the World*. SIL International, 14th edition.

Paul Heggarty, April McMahon, and Robert McMahon, 2005. *Perspectives on Variation*, chapter From phonetic similarity to dialect classification. Mouton de Gruyter.

H. Jeffreys. 1946. An invariant form for the prior probability in estimation problems. *Proc. Roy. Soc. A*, 186:453–461.

Vsevolod Kapatsinski. 2006. Sound similarity relations in the mental lexicon: Modeling the lexicon as a complex network. Technical Report 27, Indiana University Speech Research Lab.

Brett Kessler. 2005. Phonetic comparison algorithms. *Transactions of the Philological Society*, 103(2):243–260.

Gary R. Kidd and C.S. Watson. 1992. The ”proportion-of-the-total-duration rule for the discrimination of auditory patterns. *Journal of the Acoustic Society of America*, 92:3109–3118.

Grzegorz Kondrak. 2002. *Algorithms for Language Reconstruction*. Ph.D. thesis, University of Toronto.

V.I. Levenstein. 1965. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848.

Paul Luce and D. Pisoni. 1998. Recognizing spoken words: The neighborhood activation model. *Ear and Hearing*, 19:1–36.

Paul Luce, D. Pisoni, and S. Goldinger, 1990. *Cognitive Models of Speech Perception: Psycholinguistic and Computational Perspectives*, chapter Similarity neighborhoods of spoken words, pages 122–147. MIT Press, Cambridge, MA.

April McMahon and Robert McMahon. 2003. Finding families: quantitative methods in language classification. *Transactions of the Philological Society*, 101:7–55.

April McMahon, Paul Heggarty, Robert McMahon, and Natalia Slaska. 2005. Swadesh sublists and the benefits of borrowing: an andean case study. *Transactions of the Philological Society*, 103(2):147–170.

Charles A. Micchelli and Lyle Noakes. 2005. Rao distances. *Journal of Multivariate Analysis*, 92(1):97–115.

Luay Nakleh, Tandy Warnow, Don Ringe, and Steven N. Evans. 2005. A comparison of phylogenetic reconstruction methods on an ie dataset. *Transactions of the Philological Society*, 103(2):171–192.

J. Nerbonne and W. Heeringa. 1997. Measuring dialect distance phonetically. In *Proceedings of SIGPHON-97: 3rd Meeting of the ACL Special Interest Group in Computational Phonology*.

B. Port and A. Leary. 2005. Against formal phonology. *Language*, 81(4):927–964.

C.R. Rao. 1949. On the distance between two populations. *Sankhyā*, 9:246–248.

D. Ringe, Tandy Warnow, and A. Taylor. 2002. Indo-european and computational cladistics. *Transactions of the Philological Society*, 100(1):59–129.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

R.N. Shepard. 1987. Toward a universal law of generalization for physical science. *Science*, 237:1317–1323.

Richard C. Shillcock, Simon Kirby, Scott McDonald, and Chris Brew. 2001. Filled pauses and their status in the mental lexicon. In *Proceedings of the 2001 Conference of Disfluency in Spontaneous Speech*, pages 53–56.

M. Swadesh. 1952. Lexico-statistic dating of prehistoric ethnic contacts. *Proceedings of the American philosophical society*, 96(4).

Monica Tamariz. 2005. *Exploring the Adaptive Structure of the Mental Lexicon*. Ph.D. thesis, University of Edinburgh.

Enhancing electronic dictionaries with an index based on associations

Olivier Ferret
CEA –LIST/LIC2M
18 Route du Panorama
F-92265 Fontenay-aux-Roses
ferreto@zoe.cea.fr

Michael Zock[1]
LIF-CNRS
163 Avenue de Luminy
F-13288 Marseille Cedex 9
michael.zock@lif.univ-mrs.fr

Abstract

A good dictionary contains not only many entries and a lot of information concerning each one of them, but also adequate means to reveal the stored information. Information access depends crucially on the quality of the index. We will present here some ideas of how a dictionary could be enhanced to support a speaker/writer to find the word s/he is looking for. To this end we suggest to add to an existing electronic resource an index based on the notion of association. We will also present preliminary work of how a subset of such associations, for example, topical associations, can be acquired by filtering a network of lexical co-occurrences extracted from a corpus.

1 Introduction

A dictionary user typically pursues one of two goals (Humble, 2001): as a *decoder* (reading, listening), he may look for the definition or the translation of a specific target word, while as an *encoder* (speaker, writer) he may want to find a word that expresses well not only a given concept, but is also appropriate in a given context.

Obviously, readers and writers come to the dictionary with different mindsets, information and expectations concerning input and output. While the decoder can provide the word he wants additional information for, the encoder (language producer) provides the meaning of a word for which he lacks the corresponding form. In sum, users with different goals need access to different indexes, one that is based on form (decoding), the other being based on meaning or meaning relations (encoding).

Our concern here is more with the encoder, *i.e.* lexical access in language production, a feature largely neglected in lexicographical work. Yet, a good dictionary contains not only many entries and a lot of information concerning each one of them, but also efficient means to reveal the stored information. Because, what is a huge dictionary good for, if one cannot access the information it contains?

2 Lexical access on the basis of what: *concepts* (*i.e.* meanings) or *words*?

Broadly speaking, there are two views concerning lexicalization: the process is **conceptually-driven** (meaning, or parts of it are the starting point) or **lexically-driven**[2]: the target word is accessed via a source word. This is typically the case when we are looking for a *synonym*, *antonym*, *hypernym* (paradigmatic associations), or any of its syntagmatic associates (red-rose, coffee-black), the kind of association we will be concerned with here.

Yet, besides conceptual knowledge, people seem also to know a lot of things concerning the lexical form (Brown and Mc Neill, 1966): number of *syllables*, beginning/ending of the target word, *part of speech* (noun, verb, adjective, etc.), *origin* (Greek or Latin), *gender* (Vigliocco et al.,

[1] In alphabetical order

[2] Of course, the input can also be hybrid, that is, it can be composed of a conceptual and a linguistic component. For example, in order to express the notion of *intensity*, MAGN in Mel'čuk's theory (Mel'čuk *et al.*, 1995), a speaker or writer has to use different words (very, seriously, high) depending on the form of the argument (ill, wounded, price), as he says *very* ill, *seriously* wounded, *high* price. In each case he expresses the very same notion, but by using a different word. While he could use the adverb *very* for qualifying the state of somebody's health (he is *ill*), he cannot do so when qualifying the words *injury* or *price*. Likewise, he cannot use this specific adverb to qualify the noun *illness*.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 281–288,
Sydney, July 2006. ©2006 Association for Computational Linguistics

1997). While in principle, all this information could be used to constrain the search space, we will deal here only with one aspect, the words' relations to other concepts or words (associative knowledge).

Suppose, you were looking for a word expressing the following ideas: *domesticated animal*, *producing milk suitable for making cheese*. Suppose further that you knew that the target word was neither *cow, buffalo* nor *sheep*. While none of this information is sufficient to guarantee the access of the intended word *goat*, the information at hand (part of the definition) could certainly be used[3]. Besides this type of information, people often have other kinds of knowledge concerning the target word. In particular, they know how the latter relates to other words. For example, they know that *goats* and *sheep* are somehow connected, sharing a great number of features, that both are *animals* (hypernym), that *sheep* are appreciated for their wool and meat, that they tend to follow each other blindly, etc., while *goats* manage to survive, while hardly eating anything, etc. In sum, people have in their mind a huge lexico-conceptual network, with words[4], concepts or ideas being highly interconnected. Hence, any one of them can evoke the other. The likelihood for this to happen depends on such factors as *frequency* (associative strength), *saliency* and *distance* (direct vs. indirect access). As one can see, associations are a very general and powerful mechanism. No matter what we hear, read or say, anything is likely to remind us of something else. This being so, we should make use of it.

3 Accessing the target word by navigating in a huge associative network

If one agrees with what we have just said, one could view the *mental lexicon* as a huge semantic network composed of *nodes* (words and concepts) and *links* (associations), with either being able to activate the other[5]. Finding a word involves entering the network and following the links leading from the *source node* (the first word that comes to your mind) to the *target word* (the one you are looking for). Suppose you wanted to find the word *nurse* (*target word*), yet the only token coming to your mind is *hospital*. In this case the system would generate internally a graph with the *source word* at the center and all the associated words at the periphery. Put differently, the system would build internally a semantic network with *hospital* in the center and all its associated words as satellites (see Figure 1, next page).

Obviously, the greater the number of associations, the more complex the graph. Given the diversity of situations in which a given object may occur we are likely to build many associations. In other words, lexical graphs tend to become complex, too complex to be a good representation to support navigation. Readability is hampered by at least two factors: *high connectivity* (the great number of links or associations emanating from each word), and *distribution*: conceptually related nodes, that is, nodes activated by the same kind of association are scattered around, that is, they do not necessarily occur next to each other, which is quite confusing for the user. In order to solve this problem, we suggest to display by category (chunks) all the words linked by the same kind of association to the source word (see Figure 2). Hence, rather than displaying all the connected words as a flat list, we suggest to present them in chunks to allow for categorial search. Having chosen a category, the user will be presented a list of words or categories from which he must choose. If the target word is in the category chosen by the user (suppose he looked for a hypernym, hence he checked the ISA-bag), search stops, otherwise it continues. The user could choose either another category (e.g. AKO or TIORA), or a word in the current list, which would then become the new starting point.

[3] For some concrete proposals going in this direction, see dictionaries offering reverse lookup: http://www.ultralingua.net/ ,http://www.onelook.com/reverse-dictionary.shtml.

[4] Of course, one can question the very fact that people store words in their mind. Rather than considering the human mind as a *wordstore* one might consider it as a *wordfactory*. Indeed, by looking at some of the work done by psychologists who try to emulate the mental lexicon (Levelt et al., 1999) one gets the impression that words are synthesized rather than located and call up. In this case one might conclude that rather than having words in our mind we have a set of highly distributed, more or less abstract information. By propagating energy rather than data —(as there is no message passing, transformation or cumulation of information, there is only activation spreading, that is, changes of energy levels, call it weights, electronic impulses, or whatever),— that we propagate signals, activating ultimately certain peripherical organs (larynx, tongue, mouth, lips, hands) in such a way as to produce movements or sounds, that, not knowing better, we call words.

[5] While the links in our brain may only be weighted, they need to be labelled to become interpretable for human beings using them for navigational purposes in a lexicon.

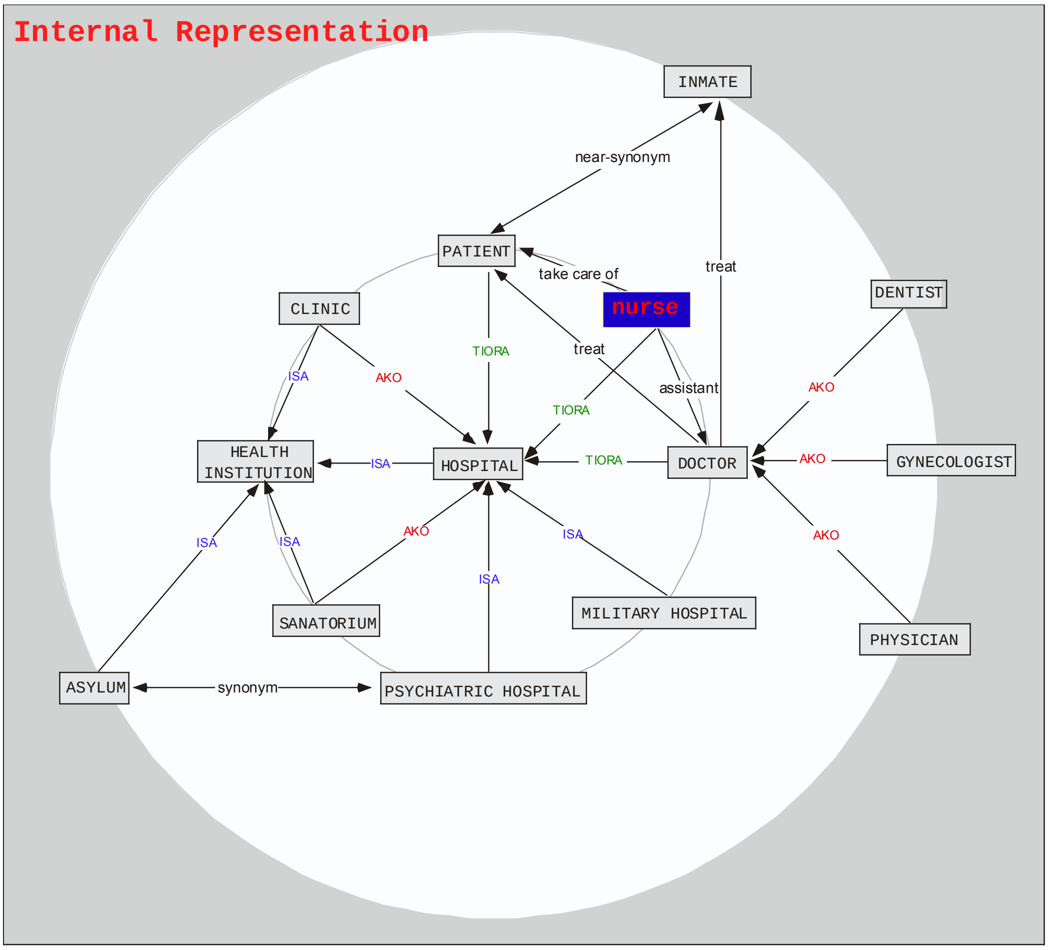


Figure 1*:* Search based on navigating in a network (internal representation)
AKO: a kind of; **ISA**: subtype; **TIORA**: **T**ypically **I**nvolved **O**bject, **R**elation or **A**ctor.

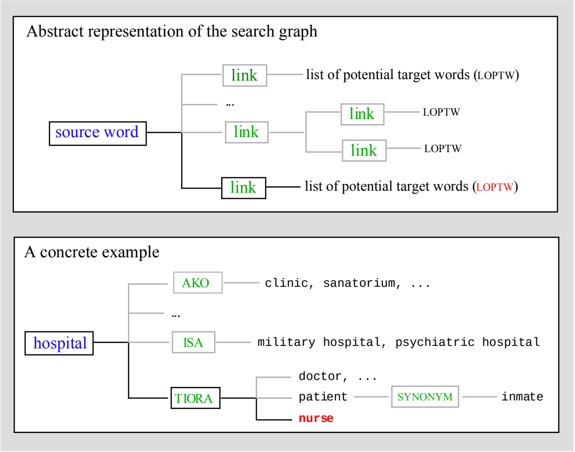

Abstract representation of the search graph
source word
link
list of potential target words (LOPTW)
...
link
link
LOPTW
link
LOPTW
link
list of potential target words (LOPTW)
A concrete example
hospital
AKO
clinic, sanatorium, ...
...
ISA
military hospital, psychiatric hospital
TIORA
doctor, ...
patient
SYNONYM
inmate
nurse

Figure 2: Proposed candidates, grouped by family, *i.e.* according to the nature of the link

As one can see, the fact that the links are labeled has some very important consequences:

(a) While maintaining the power of a highly connected graph (possible cyclic navigation), it has at the interface level the simplicity of a tree: each node points only to data of the same type, *i.e.* to the same kind of association.

(b) With words being presented in clusters, navigation can be accomplished by clicking on the appropriate category.

The assumption being that the user generally knows to which category the target word belongs (or at least, he can recognize within which of the listed categories it falls), and that categorical search is in principle faster than search in a huge list of unordered (or, alphabetically ordered) words[6].

Obviously, in order to allow for this kind of access, the resource has to be built accordingly. This requires at least two things: (a) indexing words by the associations they evoke, (b) identi-

[6] Even though very important, at this stage we shall not worry too much for the names given to the links. Indeed, one might question nearly all of them. What is important is the underlying rational: help users to navigate on the basis of symbolically qualified links. In reality a whole set of words (synonyms, of course, but not only) could amount to a link, *i.e.* be its conceptual equivalent.

fying and labelling the most frequent/useful associations. This is precisely our goal. Actually, we propose to build an associative network by enriching an existing electronic dictionary (essentially) with (syntagmatic) associations coming from a corpus, representing the average citizen's shared, basic knowledge of the world (encyclopaedia). While some associations are too complex to be extracted automatically by machine, others are clearly within reach. We will illustrate in the next section how this can be achieved.

4 Automatic extraction of *topical* relations

4.1 Definition of the problem

We have argued in the previous sections that dictionaries must contain many kinds of relations on the syntagmatic and paradigmatic axis to allow for natural and flexible access of words. Synonymy, hypernymy or meronymy fall clearly in this latter category, and well known resources like WordNet (Miller, 1995), EuroWordNet (Vossen, 1998) or MindNet (Richardson *et al.*, 1998) contain them. However, as various researchers have pointed out (Harabagiu *et al.*, 1999), these networks lack information, in particular with regard to syntagmatic associations, which are generally unsystematic. These latter, called TIORA (Zock and Bilac, 2004) or *topical relations* (Ferret, 2002) account for the fact that two words refer to the same topic, or take part in the same situation or scenario. Word-pairs like *doctor–hospital*, *burglar–policeman* or *plane–airport*, are examples in case. The lack of such topical relations in resources like WordNet has been dubbed as the *tennis problem* (Roger Chaffin, cited in Fellbaum, 1998). Some of these links have been introduced more recently in WordNet via the *domain* relation. Yet their number remains still very small. For instance, WordNet 2.1 does not contain any of the three associations mentioned here above, despite their high frequency.

The lack of systematicity of these topical relations makes their extraction and typing very difficult on a large scale. This is why some researchers have proposed to use automatic learning techniques to extend lexical networks like WordNet. In (Harabagiu & Moldovan, 1998), this was done by extracting topical relations from the glosses associated to the synsets. Other researchers used external sources: Mandala *et al.* (1999) integrated co-occurrences and a thesaurus to WordNet for query expansion; Agirre *et al.* (2001) built topic signatures from texts in relation to synsets; Magnini and Cavagliá (2000) annotated the synsets with Subject Field Codes. This last idea has been taken up and extended by (Avancini *et al.*, 2003) who expanded the domains built from this annotation.

Despite the improvements, all these approaches are limited by the fact that they rely too heavily on WordNet and some of its more sophisticated features (such as the definitions associated with the synsets). While often being exploited by acquisition methods, these features are generally lacking in similar lexico-semantic networks. Moreover, these methods attempt to learn *topical knowledge* from a lexical network rather than *topical relations*. Since our goal is different, we have chosen not to rely on any significant resource, all the more as we would like our method to be applicable to a wide array of languages. In consequence, we took an incremental approach (Ferret, 2006): starting from a network of lexical co-occurrences[7] collected from a large corpus, we used these latter to select potential topical relations by using a topical analyzer.

4.2 From a network of co-occurrences to a set of Topical Units

We start by extracting lexical co-occurrences from a corpus to build a network. To this end we follow the method introduced by (Church and Hanks, 1990), *i.e.* by sliding a window of a given size over some texts. The parameters of this extraction were set in such a way as to catch the most obvious topical relations: the window was fairly large (20-words wide), and while it took text boundaries into account, it ignored the order of the co-occurrences. Like (Church and Hanks, 1990), we used mutual information to measure the cohesion between two words. The finite size of the corpus allows us to normalize this measure in line with the maximal mutual information relative to the corpus.

This network is used by TOPICOLL (Ferret, 2002), a topic analyzer, which performs simultaneously three tasks, relevant for this goal:

- it segments texts into topically homogeneous segments;
- it selects in each segment the most representative words of its topic;

[7] Such a network is only another view of a set of co-occurrences: its nodes are the co-occurrent words and its edges are the co-occurrence relations.

- it proposes a restricted set of words from the co-occurrence network to expand the selected words of the segment.

These three tasks rely on a common mechanism: a window is moved over the text to be analyzed in order to limit the focus space of the analysis. This latter contains a lemmatized version of the text's plain words. For each position of this window, we select only words of the co-occurrence network that are linked to at least three other words of the window (see Figure 3). This leads to select both words that are in the window (first order co-occurrents) and words coming from the network (second order co-occurrents). The number of links between the selected words of the network, called *expansion words*, and those of the window is a good indicator of the topical coherence of the window's content. Hence, when their number is small, a segment boundary can be assumed. This is the basic principle underlying our topic analyzer.

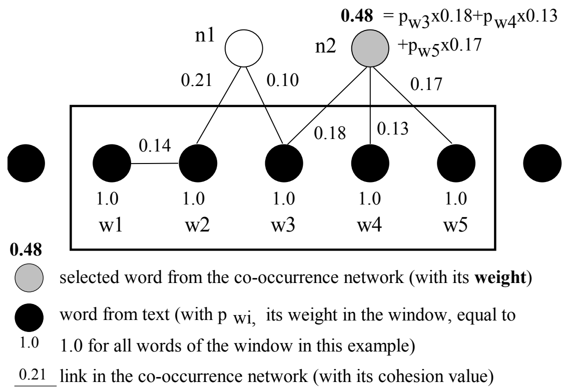

0.48 = p_w3 x0.18+p_w4 x0.13 +p_w5 x0.17
n1
n2
0.21
0.10
0.17
0.18
0.13
0.14
1.0
1.0
1.0
1.0
1.0
w1
w2
w3
w4
w5
0.48
selected word from the co-occurrence network (with its weight)
word from text (with p wi, its weight in the window, equal to
1.0
1.0 for all words of the window in this example)
0.21
link in the co-occurrence network (with its cohesion value)

Figure 3: Selection and weighting of words from the co-occurrence network

The words selected for each position of the window are summed, to keep only those occurring in 75% of the positions of the segment. This allows reducing the number of words selected from non-topical co-occurrences. Once a corpus has been processed by TOPICOLL, we obtain a set of segments and a set of expansion words for each one of them. The association of the selected words of a segment and its expansion words is called a Topical Unit. Since both sets of words are selected for reasons of topical homogeneity, their co-occurrence is more likely to be a topical relation than in our initial network.

4.3 Filtering of Topical Units

Before recording the co-occurrences in the Topical Units built in this way, the units are filtered twice. The first filter aims at discarding heterogeneous Topical Units, which can arise as a side effect of a document whose topics are so intermingled that it is impossible to get a reliable linear segmentation of the text. We consider that this occurs when for a given text segment, no word can be selected as a representative of the topic of the segment. Moreover, we only keep the Topical Units that contain at least two words from their original segment. A topic is defined here as a configuration of words. Note that the identification of such a configuration cannot be based solely on a single word.

Text words	**Expansion words**
surveillance (*watch*)	police_judiciaire (*judiciary police*)
téléphonique (*telephone*)	écrouer (*to imprison*)
juge (*judge*)	garde_à_vue (*police custody*)
policier (*policeman*)	écoute_téléphonique (*phone tapping*)
brigade (*squad*)	juge_d'instruction (*examining judge*)
enquête (*investigation*)	contrôle_judiciaire (*judicial review*)
placer (*to put*)	

Table 1: Content of a filtered Topical Unit

The second filter is applied to the expansion words of each Topical Unit to increase their topical homogeneity. The principle of the filtering of these words is the same as the principle of their selection described in Section 4.2: an expansion word is kept if it is linked in the co-occurrence network to at least three text words of the Topical Unit. Moreover, a selective threshold is applied to the frequency and the cohesion of the co-occurrences supporting these links: only co-occurrences whose frequency and cohesion are respectively higher or equal to 15 and 0.15 are used. For instance in Table 1, which shows an example of a Topical Unit after its filtering, *écrouer* (*to imprison*) is selected, because it is linked in the co-occurrence network to the following words of the text:

juge (judge): 52 (frequency) – 0.17 (cohesion)
policier (policeman): 56 – 0.17
enquête (investigation): 42 – 0.16

word	freq.	word	freq.	word	freq.	word	freq.
scène (*stage*)	884	théâtral (*dramatic*)	62	cynique (*cynical*)	26	scénique (*theatrical*)	14
théâtre (*theater*)	679	scénariste (*scriptwriter*)	51	miss (*miss*)	20	Chabol (*Chabol*)	13
réalisateur (*director*)	220	comique (*comic*)	51	parti_pris (*bias*)	16	Tchekov (*Tchekov*)	13
cinéaste (*film-marker*)	135	oscar (*oscar*)	40	monologue (*monolog*)	15	allocataire (*beneficiary*)	13
comédie (*comedy*)	104	film_américain (*american film*)	38	revisiter (*to revisit*)	14	satirique (*satirical*)	13
costumer (*to dress up*)	63	hollywoodien (*Hollywood*)	30	gros_plan (*close-up*)	14		

Table 2: Co-occurrents of the word *acteur* (*actor*) with a cohesion of 0.16
(the co-occurrents removed by our filtering method are underlined)

4.4 From Topical Units to a network of topical relations

After the filtering, a Topical Unit gathers a set of words supposed to be strongly coherent from the topical point of view. Next, we record the co-occurrences between these words for all the Topical Units remaining after filtering. Hence, we get a large set of topical co-occurrences, despite the fact that a significant number of non-topical co-occurrences remains, the filtering of Topical Units being an unsupervised process. The frequency of a co-occurrence in this case is given by the number of Topical Units containing both words simultaneously. No distinction concerning the origin of the words of the Topical Units is made.

The network of topical co-occurrences built from Topical Units is a subset of the initial network. However, it also contains co-occurrences that are not part of it, *i.e.* co-occurrences that were not extracted from the corpus used for setting the initial network or co-occurrences whose frequency in this corpus was too low. Only some of these "new" co-occurrences are topical. Since it is difficult to estimate globally which ones are interesting, we have decided to focus our attention only on the co-occurrences of the topical network already present in the initial network.

Thus, we only use the network of topical co-occurrences as a filter for the initial co-occurrence network. Before doing so, we filter the topical network in order to discard co-occurrences whose frequency is too low, that is, co-occurrences that are unstable and not representative. From the use of the final network by TOPICOLL (see Section 4.5), we set the threshold experimentally to 5. Finally, the initial network is filtered by keeping only co-occurrences present in the topical network. Their frequency and cohesion are taken from the initial network. While the frequencies given by the topical network are potentially interesting for their topical significance, we do not use them because the results of the filtering of Topical Units are too hard to evaluate.

4.5 Results and evaluation

We applied the method described here to an initial co-occurrence network extracted from a corpus of 24 months of *Le Monde*, a major French newspaper. The size of the corpus was around 39 million words. The initial network contained 18,958 words and 341,549 relations. The first run produced 382,208 Topical Units. After filtering, we kept 59% of them. The network built from these Topical Units was made of 11,674 words and 2,864,473 co-occurrences. 70% of these co-occurrences were new with regard to the initial network and were discarded. Finally, we got a filtered network of 7,160 words and 183,074 relations, which represents a cut of 46% of the initial network. A qualitative study showed that most of the discarded relations are non-topical. This is illustrated by Table 2, which gives the co-occurrents of the word *acteur (actor)* that are filtered by our method among its co-occurrents with a high cohesion (equal to 0.16). For instance, the words *cynique (cynical)* or *allocataire (beneficiary)* are cohesive co-occurrents of the word *actor*, even though they are not topically linked to it. These words are filtered out, while we keep words like *gros_plan (close-up)* or *scénique (theatrical)*, which topically cohere with *acteur (actor)* despite their lower frequency than the discarded words.

	Recall[8]	**Precision**	**F1-measure**	**Error (P_k)**[9]
initial (I)	0.85	0.79	0.82	0.20
topical filtering (T)	0.85	0.79	0.82	0.21
frequency filtering (F)	0.83	0.71	0.77	0.25

Table 3: TOPICOLL's results with different networks

In order to evaluate more objectively our work, we compared the quantitative results of TOPICOLL with the initial network and its filtered version. The evaluation showed that the performance of the segmenter remains stable, even if we use a topically filtered network (see Table 3). Moreover, it became obvious that a network filtered only by frequency and cohesion performs significantly less well, even with a comparable size. For testing the statistical significance of these results, we applied to the P_k values a one-side t-test with a null hypothesis of equal means. Levels lower or equal to 0.05 are considered as statistically significant:

p_{val} (I-T): 0.08
p_{val} (I-F): 0.02
p_{val} (T-F): 0.05

These values confirm that the difference between the initial network (I) and the topically filtered one (T) is actually not significant, whereas the filtering based on co-occurrence frequencies leads to significantly lower results, both compared to the initial network and the topically filtered one. Hence, one may conclude that our method is an effective way of selecting topical relations by preference.

5 Discussion and conclusion

We have raised and partially answered the question of how a dictionary should be indexed in order to support word access, a question initially addressed in (Zock, 2002) and (Zock and Bilac, 2004). We were particularly concerned with the language producer, as his needs (and knowledge at the onset) are quite different from the ones of the language receiver (listener/reader). It seems that, in order to achieve our goal, we need to do two things: add to an existing electronic dictionary information that people tend to associate with a word, that is, build and enrich a semantic network, and provide a tool to navigate in it. To this end we have suggested to label the links, as this would reduce the graph complexity and allow for type-based navigation. Actually our basic proposal is to extend a resource like WordNet by adding certain links, in particular on the syntagmatic axis. These links are associations, and their role consists in helping the encoder to find ideas (concepts/words) related to a given stimulus (brainstorming), or to find the word he is thinking of (word access).

One problem that we are confronted with is to identify possible associations. Ideally we would need a complete list, but unfortunately, this does not exist. Yet, there is a lot of highly relevant information out there. For example, Mel'cuk's lexical functions (Mel'cuk, 1995), Fillmore's FRAMENET[10], work on ontologies (CYC), thesaurus (Roget), WordNets (the original version from Princeton, various Euro-WordNets, BalkaNet), HowNet[11], the work done by MICRA, the FACTOTUM project[12], or the Wordsmyth dictionary/thesaurus[13].

Since words are linked via associations, it is important to reveal these links. Once this is done, words can be accessed by following these links. We have presented here some preliminary work for extracting an important subset of such links from texts, topical associations, which are generally absent from dictionaries or resources like WordNet. An evaluation of the topic segmentation has shown that the relations extracted are sound from the topical point of view, and that they can be extracted automatically. However,

[8] Precision is given by N_t / N_b and recall by N_t / D, with D being the number of document breaks, N_b the number of boundaries found by TOPICOLL and N_t the number of boundaries that are document breaks (the boundary should not be farther than 9 plain words from the document break).

[9] P_k (Beeferman et al., 1999) evaluates the probability that a randomly chosen pair of words, separated by k words, is wrongly classified, *i.e.* they are found in the same segment by TOPICOLL, while they are actually in different ones (miss of a document break), or they are found in different segments, while they are actually in the same one (false alarm).

[10] http://www.icsi.berkeley.edu/~framenet/

[11] http://www.keenage.com/html/e_index.html

[12] http://humanities.uchicago.edu/homes/MICRA/

[13] http://www.wordsmyth.com/

they still contain too much noise to be directly exploitable by an end user for accessing a word in a dictionary. One way of reducing the noise of the extracted relations would be to build from each text a representation of its topics and to record the co-occurrences in these representations rather than in the segments delimited by a topic segmenter. This is a hypothesis we are currently exploring. While we have focused here only on word access on the basis of (other) words, one should not forget that most of the time speakers or writers start from meanings. Hence, we shall consider this point more carefully in our future work, by taking a serious look at the proposals made by Bilac et al. (2004); Durgar and Oflazer (2004), or Dutoit and Nugues (2002).

References

Eneko Agirre, Olatz Ansa, David Martinez and Eduard Hovy. 2001. Enriching WordNet concepts with topic signatures. In *NAACL'01 Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations*.

Henri Avancini, Alberto Lavelli, Bernardo Magnini, Fabrizio Sebastiani and Roberto Zanoli. 2003. Expanding Domain-Specific Lexicons by Term Categorization. In *18th ACM Symposium on Applied Computing (SAC-03)*.

Doug Beeferman, Adam Berger and Lafferty. 1999. Statistical Models for Text Segmentation. *Machine Learning*, 34(1): 177-210.

Slaven Bilac, Wataru Watanabe, Taiichi Hashimoto, Takenobu Tokunaga and Hozumi Tanaka. 2004. Dictionary search based on the target word description. In *Tenth Annual Meeting of The Association for Natural Language Processing (NLP2004)*, pages 556-559.

Roger Brown and David McNeill. 1996. The tip of the tongue phenomenon. *Journal of Verbal Learning and Verbal Behaviour*, 5: 325-337.

Kenneth Church and Patrick Hanks. 1990. Word Association Norms, Mutual Information, And Lexicography. *Computational Linguistics*, 16(1): 177-210.

Ilknur Durgar El-Kahlout and Kemal Oflazer. 2004. Use of Wordnet for Retrieving Words from Their Meanings, In *2nd Global WordNet Conference*, Brno

Dominique Dutoit and Pierre Nugues. 2002. A lexical network and an algorithm to find words from definitions. In *15th European Conference on Artificial Intelligence (ECAI 2002)*, Lyon, pages 450-454, IOS Press.

Christiane Fellbaum. 1998. *WordNet - An Electronic Lexical Database*, MIT Press.

Olivier Ferret. 2006. Building a network of topical relations from a corpus. In *LREC 2006*.

Olivier Ferret. 2002. Using collocations for topic segmentation and link detection. In *COLING 2002*, pages 260-266.

Sanda M. Harabagiu, George A. Miller and Dan I. Moldovan. 1999. WordNet 2 - A Morphologically and Semantically Enhanced Resource. In *ACL-SIGLEX99: Standardizing Lexical Resources*, pages 1-8.

Sanda M. Harabagiu and Dan I. Moldovan. 1998. Knowledge Processing on an Extended WordNet. *In WordNet - An Electronic Lexical Database*, pages 379-405.

Philip Humble. 2001. *Dictionaries and Language Learners*, Haag and Herchen.

William Levelt, Ardi Roelofs and Antje Meyer. 1999. A theory of lexical access in speech production, *Behavioral and Brain Sciences*, 22: 1-75.

Bernardo Magnini and Gabriela Cavagliá. 2000. Integrating Subject Field Codes into WordNet. In *LREC 2000*.

Rila Mandala, Takenobu Tokunaga and Hozumi Tanaka. 1999. Complementing WordNet with Roget's and Corpus-based Thesauri for Information Retrieval. *In EACL 99*.

Igor Mel'čuk, Arno Clas and Alain Polguère. 1995. *Introduction à la lexicologie explicative et combinatoire*, Louvain, Duculot.

George A. Miller. 1995. WordNet: A lexical Database, *Communications of the ACM*. 38(11): 39-41.

Stephen D. Richardson, William B. Dolan and Lucy Vanderwende. 1998. MindNet: Acquiring and Structuring Semantic Information from Text. In *ACL-COLING'98*, pages 1098-1102.

Piek Vossen. 1998. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publisher.

Gabriella Vigliocco, Antonini, T., and Merryl Garrett. 1997. Grammatical gender is on the tip of Italian tongues. *Psychological Science*, 8: 314-317.

Michael Zock. 2002. Sorry, what was your name again, or how to overcome the tip-of-the tongue problem with the help of a computer? In *SemaNet workshop, COLING 2002*, Taipei. http://acl.ldc.upenn.edu /W/W02/W02-1118.pdf

Michael Zock and Slaven Bilac. 2004. Word lookup on the basis of associations: from an idea to a roadmap. In *COLING 2004 workshop: Enhancing and using dictionaries*, Geneva. http://acl.ldc.upenn.edu/ coling2004/W10/pdf/5.pdf

Guiding a Constraint Dependency Parser with Supertags

Kilian Foth, Tomas By, and Wolfgang Menzel
Department für Informatik, Universität Hamburg, Germany
foth|by|menzel@informatik.uni-hamburg.de

Abstract

We investigate the utility of supertag information for guiding an existing dependency parser of German. Using weighted constraints to integrate the additionally available information, the decision process of the parser is influenced by changing its preferences, without excluding alternative structural interpretations from being considered. The paper reports on a series of experiments using varying models of supertags that significantly increase the parsing accuracy. In addition, an upper bound on the accuracy that can be achieved with perfect supertags is estimated.

1 Introduction

Supertagging is based on the combination of two powerful and influential ideas of natural language processing: On the one hand, parsing is (at least partially) reduced to a decision on the optimal sequence of categories, a problem for which efficient and easily trainable procedures exist. On the other hand, supertagging exploits complex categories, i.e. tree fragments which much better reflect the mutual compatibility between neighbouring lexical items than say part-of-speech tags.

Bangalore and Joshi (1999) derived the notion of supertag within the framework of Lexicalized Tree-Adjoining Grammars (LTAG) (Schabes and Joshi, 1991). They considered supertagging a process of almost parsing, since all that needs to be done after having a sufficiently reliable sequence of supertags available is to decide on their combination into a spanning tree for the complete sentence. Thus the approach lends itself easily to preprocessing sentences or filtering parsing results with the goal of guiding the parser or reducing its output ambiguity.

Nasr and Rambow (2004) estimated that perfect supertag information already provides for a parsing accuracy of 98% if a correct supertag assignment were available. Unfortunately, perfectly reliable supertag information cannot be expected; usually this uncertainty is compensated by running the tagger in multi-tagging mode, expecting that the reliability can be increased by not forcing the tagger to take unreliable decisions but instead offering a set of alternatives from which a subsequent processing component can choose.

A grammar formalism which seems particularly well suited to decompose structural descriptions into lexicalized tree fragments is dependency grammar. It allows us to define supertags on different levels of granularity (White, 2000; Wang and Harper, 2002), thus facilitating a fine grained analysis of how the different aspects of supertag information influence the parsing behaviour. In the following we will use this characteristic to study in more detail the utility of different kinds of supertag information for guiding the parsing process.

Usually supertags are combined with a parser in a filtering mode, i.e. parsing hypotheses which are not compatible with the supertag predictions are simply discarded. Drawing on the ability of Weighted Constraint Dependency Grammar (WCDG) (Schröder et al., 2000) to deal with defeasible constraints, here we try another option for making available supertag information: Using a *score* to estimate the general reliability of unique supertag decisions, the information can be combined with evidence derived from other constraints of the grammar in a soft manner. It makes possible to rank parsing hypotheses according to their plausibility and allows the parser to even override potentially wrong supertag decisions.

Starting from a range of possible supertag models, Section 2 explores the reliability with which dependency-based supertags can be determined on

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 289–296,
Sydney, July 2006. ©2006 Association for Computational Linguistics

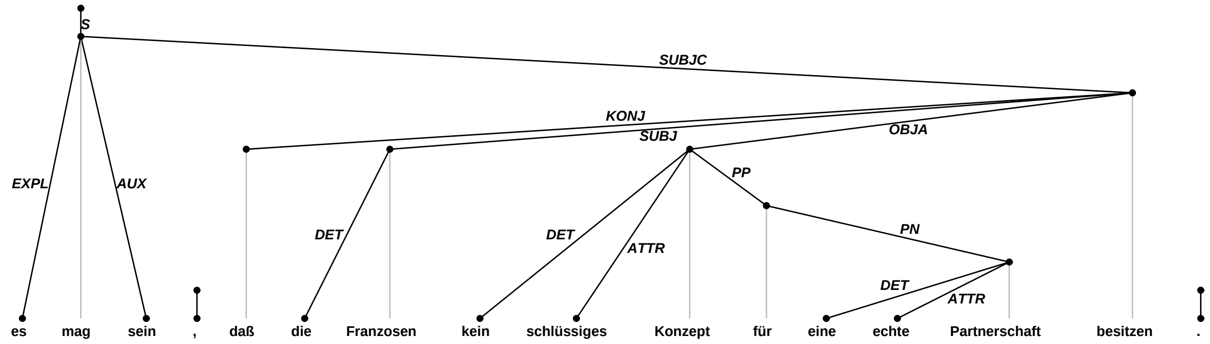


Figure 1: Dependency tree for sentence 19601 of the NEGRA corpus.

different levels of granularity. Then, Section 3 describes how supertags are integrated into the existing parser for German. The complex nature of supertags as we define them makes it possible to separate the different structural predictions made by a single supertag into components and study their contributions independently (c.f. Section 4). We can show that indeed the parser is robust enough to tolerate supertag errors and that even with a fairly low tagger performance it can profit from the additional, though unreliable information.

2 Supertagging German text

In defining the nature of supertags for dependency parsing, a trade-off has to be made between expressiveness and accuracy. A simple definition with very small number of supertags will not be able to capture the full variety of syntactic contexts that actually occur, while an overly expressive definition may lead to a tag set that is so large that it cannot be accurately learnt from the training data. The local context of a word to be encoded in a supertag could include its edge label, the attachment direction, the occurrence of obligatory[1] or of all dependents, whether each predicted dependent occurs to the right or to the left of the word, and the relative order among different dependents. The simplest useful task that could be asked of a supertagger would be to predict the dependency relation that each word enters. In terms of the WCDG formalism, this means associating each word at least with one of the syntactic labels that decorate dependency edges, such as SUBJ or DET; in other words, the supertag set would be identical to the label set. The example sentence

> "Es mag sein, daß die Franzosen kein schlüssiges Konzept für eine echte Partnerschaft besitzen."
>
> (Perhaps the French do not have a viable concept for a true partnership.)

if analyzed as in Figure 1, would then be described by a supertag sequence beginning with `EXPL S AUX ...`

Following (Wang and Harper, 2002), we further classify dependencies into Left (L), Right (R), and No attachments (N), depending on whether a word is attached to its left or right, or not at all. We combine the label with the attachment direction to obtain composite supertags. The sequence of supertags describing the example sentence would then begin with `EXPL/R S/N AUX/L ...`

Although this kind of supertag describes the role of each word in a sentence, it still does not specify the entire local context; for instance, it associates the information that a word functions as a subject only with the subject and not with the verb that takes the subject. In other words, it does not predict the relations *under* a given word. Greater expressivity is reached by also encoding the labels of these relations into the supertag. For instance, the word 'mag' in the example sentence is modified by an expletive (EXPL) on its left side and by an auxiliary (AUX) and a subject clause (SUBJC) dependency on its right side. To capture this extended local context, these labels must be encoded into the supertag. We add the local context of a word to the end of its supertag, separated with the delimiter `+`. This yields the expression `S/N+AUX,EXPL,SUBJC`. If we also want to express that the EXPL precedes the word but the AUX follows it, we can instead add two new fields to the left and to the right of the supertag, which leads to the new supertag `EXPL+S/N+AUX,SUBJC`.

Table 1 shows the annotation of the example us-

[1] The model of German used here considers the objects of verbs, prepositions and conjunctions to be obligatory and most other relations as optional. This corresponds closely to the set of needs roles of (Wang and Harper, 2002).

Word	Supertag model J
es	+EXPL/R+
mag	EXPL+S/N+AUX,SUBJC
sein	+AUX/L+
,	+/N+
daß	+KONJ/R+
die	+DET/R+
Franzosen	DET+SUBJ/R+
kein	+DET/R+
schlüssiges	+ATTR/R+
Konzept	ATTR,DET+OBJA/R+PP
für	+PP/L+PN
eine	+DET/R+
echte	+ATTR/R+
Partnerschaft	ATTR,DET+PN/L+
besitzen	KONJ,OBJA,SUBJ+SUBJC/L+
.	+/N+

Table 1: An annotation of the example sentence

ST model	Prediction of				#tags	Supertag accuracy	Component accuracy
	label	direction	dependents	order			
A	yes	no	none	no	35	84.1%	84.1%
B	yes	yes	none	no	73	78.9%	85.7%
C	yes	no	oblig.	no	914	81.1%	88.5%
D	yes	yes	oblig.	no	1336	76.9%	90.8%
E	yes	no	oblig.	yes	1465	80.6%	91.8%
F	yes	yes	oblig.	yes	2026	76.2%	90.9%
G	yes	no	all	no	6858	71.8%	81.3%
H	yes	yes	all	no	8684	67.9%	85.8%
I	yes	no	all	yes	10762	71.6%	84.3%
J	yes	yes	all	yes	12947	67.6%	84.5%

Table 2: Definition of all supertag models used.

ing the most sophisticated supertag model. Note that the notation +EXPL/R+ explicitly represents the fact that the word labelled EXPL has no dependents of its own, while the simpler EXPL/R made no assertion of this kind. The extended context specification with two + delimiters expresses the complete set of dependents of a word and whether they occur to its left or right. However, it does not distinguish the order of the left or right dependents among each other (we order the labels on either side alphabetically for consistency). Also, duplicate labels among the dependents on either side are not represented. For instance, a verb with two post-modifying prepositions would still list PP only once in its right context. This ensures that the set of possible supertags is finite. The full set of different supertag models we used is given in Table 2. Note that the more complicated models G, H, I and J predict all dependents of each word, while the others predict obligatory dependents only, which should be an easier task.

To obtain and evaluate supertag predictions, we used the NEGRA and TIGER corpora (Brants et al., 1997; Brants et al., 2002), automatically transformed into dependency format with the freely available tool DepSy (Daum et al., 2004). As our test set we used sentences 18,602–19,601 of the NEGRA corpus, for comparability to earlier work. All other sentences (59,622 sentences with 1,032,091 words) were used as the training set. For each word in the training set, the local context was extracted and expressed in our supertag notation. The word/supertag pairs were then used to train the statistical part-of-speech tagger TnT (Brants, 2000), which performs trigram tagging efficiently and allows easy retraining on different data. However, a few of TnT's limitations had to be worked around: since it cannot deal with words that have more than 510 different possible tags, we systematically replaced the rarest tags in the training set with a generic 'OTHER' tag until the limit was met. Also, in tagging mode it can fail to process sentences with many unknown words in close succession. In such cases, we simply ran it on shorter fragments of the sentence until no error occurred. Fewer than 0.5% of all sentences were affected by this problem even with the largest tag set.

A more serious problem arises when using a stochastic process to assign tags that partially predict structure: the tags emitted by the model may contradict each other. Consider, for instance, the following supertagger output for the previous example sentence:

```
es: +EXPL/R+  mag: +S/N+AUX,SUBJC
sein: PRED+AUX/L+  ...
```

The supertagger correctly predicts that the first three labels are EXPL, S, and AUX. It also predicts that the word 'sein' has a preceding PRED complement, but this is impossible if the two preceding words are labelled EXPL and S. Such contradictory information is not fatal in a robust system, but it is likely to cause unnecessary work for the parser when some rules demand the impossible. We therefore decided simply to ignore context predictions when they contradict the basic label predictions made for the same sentence; in other words, we pretend that the prediction for the third word was just +AUX/L+ rather than PRED+AUX/L+. Up to 13% of all predictions were simplified in this way for the most complex supertag model.

The last columns of Table 2 give the number of different supertags in the training set and the performance of the retrained TnT on the test set in single-tagging mode. Although the number of occurring tags rises and the prediction accuracy falls with the supertag complexity, the correlation is not absolute: It seems markedly easier to predict supertags with complements but no direction information (C) than supertags with direction information but no complements (B), although the tag set is larger by an order of magnitude. In fact, the prediction of attachment direction seems much more difficult than that of undirected supertags in every case, due to the semi-free word order of German. The greater tag set size when predicting complements of each words is at least partly offset by the contextual information available to the *n*-gram model, since it is much more likely that a word will have, e.g., a 'SUBJ' complement when an adjacent 'SUBJ' supertag is present.

For the simplest model A, all 35 possible supertags actually occur, while in the most complicated model J, only 12,947 different supertags are observed in the training data (out of a theoretically possible 10^{24} for a set of 35 edge labels). Note that this is still considerably larger than most other reported supertag sets. The prediction quality falls to rather low values with the more complicated models; however, our goal in this paper is not to optimize the supertagger, but to estimate the effect that an imperfect one has on an existing parser. Altogether most results fall into a range of 70–80% of accuracy; as we will see later, this is in fact enough to provide a benefit to automatic parsing.

Although supertag accuracy is usually determined by simply counting matching and non-matching predictions, a more accurate measure should take into account how many of the individual predictions that are combined into a supertag are correct or wrong. For instance, a word that is attached to its left as a subject, is preceded by a preposition and an attributive adjective, and followed by an apposition would bear the supertag `PP,ATTR+SUBJ/L+APP`. Since the prepositional attachment is notoriously difficult to predict, a supertagger might miss it and emit the slightly different tag `ATTR+SUBJ/L+APP`. Although this supertag is technically wrong, it is in fact much more right than wrong: of the four predictions of label, direction, preceding and following dependents, three are correct and only one is wrong. We therefore define the *component accuracy* for a given model as the ratio of correct predictions among the possible ones, which results in a value of 0.75 rather than 0 for the example prediction. The component accuracy of the supertag model J e. g. is in fact 84.5% rather than 67.6%. We would expect the component accuracy to match the effect on parsing more closely than the supertag accuracy.

3 Using supertag information in WCDG

Weighted Constraint Dependency Grammar (WCDG) is a formalism in which declarative constraints can be formulated that describe well-formed dependency trees in a particular natural language. A grammar composed of such constraints can be used for parsing by feeding it to a constraint-solving component that searches for structures that satisfy the constraints.

Each constraint carries a numeric score or *penalty* between 0 and 1 that indicates its importance. The penalties of all instances of constraint violations are multiplied to yield a score for an entire analysis; hence, an analysis that satisfies all rules of the WCDG bears the score 1, while lower values indicate small or large aberrations from the language norm. A constraint penalty of 0, then, corresponds to a hard constraint, since every analysis that violates such a constraint will always bear the worst possible score of 0. This means that of two constraints, the one with the *lower* penalty is more important to the grammar.

Since constraints can be soft as well as hard, parsing in the WCDG formalism amounts to multi-dimensional optimization. Of two possible analyses of an utterance, the one that satisfies more (or more important) constraints is always preferred. All knowledge about grammatical rules is encoded in the constraints that (together with the lexicon) constitute the grammar. Adding a constraint which is sensitive to supertag predictions will therefore change the objective function of the optimization problem, hopefully leading to a higher share of correct attachments. Details about the WDCG parser can be found in (Foth and Menzel, 2006).

A grammar of German is available (Foth et al., 2004) that achieves a good accuracy on written German input. Despite its good results, it seems probable that the information provided by a supertag prediction component could improve the accuracy further. First, because the optimization problem that WCDG defines is infeasible to solve exactly, the parser must usually use incomplete, heuristic algorithms to try to compute the optimal analysis. This means that it sometimes fails to find the correct analysis even if the language model accurately defines it, because of search errors during heuristic optimization. A component that makes specific predictions about local structure could guide the process so that the correct alternative is tried first in more cases, and help prevent such search errors. Second, the existing grammar rules deal mainly with structural compatibility, while supertagging exploits patterns in the sequence of words in its input, i. e. both models contribute complementary information. Moreover, the parser can be expected to profit from supertags providing highly lexicalized pieces of information.

	Supertag	Component	Parsing accuracy	
Model	accuracy	accuracy	unlabelled	labelled
baseline	–	–	89.6%	87.9%
A	84.1%	84.1%	90.8%	89.4%
B	78.9%	85.7%	90.6%	89.2%
C	81.1%	88.5%	91.0%	89.6%
D	76.9%	90.8%	91.1%	89.8%
E	80.6%	91.8%	90.9%	89.6%
F	76.2%	90.9%	91.4%	90.0%
G	71.8%	81.3%	90.8%	89.4%
H	67.9%	85.8%	90.8%	89.4%
I	71.6%	84.3%	91.8%	90.4%
J	67.6%	84.5%	91.8%	90.5%

Table 3: Influence of supertag integration on parsing accuracy.

	Parsing accuracy	
Constraint penalty	unlabelled	labelled
0.0	3.7%	3.7%
0.05	85.2%	83.5%
0.1	87.6%	85.7%
0.2	88.9%	87.3%
0.5	91.2%	89.5%
0.7	91.5%	90.1%
0.9	91.8%	90.5%
0.95	91.1%	89.8%
1.0	89.6%	87.9%

Table 4: Parsing accuracy depending on different strength of supertag integration.

To make the information from the supertag sequence available to the parser, we treat the complex supertags as a set of predictions and write constraints to prefer those analyses that satisfy them. The predictions of label and direction made by models A and B are mapped onto two constraints which demand that each word in the analysis should exhibit the predicted label and direction. The more complicated supertag models constrain the local context of each word further. Effectively, they predict that the specified dependents of a word occur, and that no other dependents occur. The former prediction equates to an existence condition, so constraints are added which demand the presence of the predicted relation types under that word (one for left dependents and one for right dependents). The latter prediction disallows all other dependents; it is implemented by two constraints that test the edge label of each word-to-word attachment against the set of predicted dependents of the regent (again, separately for left and right dependents). Altogether six new constraints are added to the grammar which refer to the output of the supertagger on the current sentence.

Note that in contrast to most other approaches we do not perform multi-supertagging; exactly one supertag is assumed for each word. Alternatives could be integrated by computing the logical disjunctions of the predictions made by each supertag, and then adapting the new constraints accordingly.

4 Experiments

We tested the effect of supertag predictions on a full parser by adding the new constraints to the WCDG of German described in (Foth et al., 2004) and re-parsing the same 1,000 sentences from the NEGRA corpus. The quality of a dependency parser such as this can be measured as the ratio of correctly attached words to all words (structural accuracy) or the ratio of the correctly attached and correctly labelled words to all words (labelled accuracy). Note that because the parser always finds exactly one analysis with exactly one subordination per word, there is no distinction between recall and precision. The structural accuracy without any supertags is 89.6%.

To determine the best trade-off between complexity and prediction quality, we tested all 10 supertag models against the baseline case of no supertags at all. The results are given in Table 3. Two observations can be made about the effect of the supertag model on parsing. Firstly, all types of supertag prediction, even the very basic model A which predicts only edge labels, improve the overall accuracy of parsing, although the baseline is already quite high. Second, the richer models of supertags appear to be more suitable for guiding the parser than the simpler ones, even though their own accuracy is markedly lower; almost one third of the supertag predictions according to the most complicated definition J are wrong, but nevertheless their inclusion reduces the remaining error rate of the parser by over 20%.

This result confirms the assumption that if supertags are integrated as individual constraints, their component accuracy is more important than the supertag accuracy. The decreasing accuracy of more complex supertags is more than counterbalanced by the additional information that they contribute to the analysis. Obviously, this trend cannot continue indefinitely; a supertag definition that predicted even larger parts of the dependency tree would certainly lead to much lower accuracy by even the most lenient measure, and a prediction that is mostly wrong must ultimately degrade parsing performance. Since the most complex model J shows no parsing improvement over its successor I, this point might already have been reached.

The use of supertags in WCDG is comparable to previous work which integrated POS tagging and chunk parsing. (Foth and Hagenström, 2002; Daum et al., 2003) showed that the correct balance between the new knowledge and the existing grammar is crucial for successful integration. This is achieved by means of an additional parameter, modeling how trustworthy supertag predictions are considered. Its effect is shown in Table 4. As expected, making supertag constraints hard (with a value of 0.0) over-constrains most parsing problems, so that hardly any analyses can be computed. Other values near 0 avoid this problem but still lead to much worse overall performance, as wrong or even impossible predictions too often overrule the normal syntax constraints. The previously used value of 0.9 actually yields the best results with this particular grammar.

The fact that a statistical model can improve parsing performance when superimposed on a sophisticated hand-written grammar is of particular interest because the statistical model we used is so simple, and in fact not particularly accurate; it certainly does not represent the state of the art in supertagging. This gives rise to the hope that as better supertaggers for German become available, parsing results will continue to see additional improvements, i.e., future supertagging research will directly benefit parsing. The obvious question is how great this benefit might conceivably become under optimal conditions. To obtain this upper limit of the utility of supertags we repeated the process of translating each supertag into additional WCDG constraints, but this time using the test set itself rather than TnT's predictions.

Supertag	Constraint penalty	
model	0.9	0.0
A	92.7% / 92.2%	94.0% / 94.0%
B	94.3% / 93.7%	96.0% / 96.0%
C	92.8% / 92.4%	94.1% / 94.1%
D	94.3% / 93.8%	96.0% / 96.0%
E	93.1% / 92.6%	94.3% / 94.3%
F	94.6% / 94.1%	96.1% / 96.1%
G	94.2% / 93.7%	95.8% / 95.8%
H	95.2% / 94.7%	97.4% / 97.4%
I	97.1% / 96.8%	99.5% / 99.5%
J	97.1% / 96.8%	99.6% / 99.6%

Table 5: Unlabelled and labelled parsing accuracy with a simulated perfect supertagger.

Table 5 again gives the unlabelled and labelled parsing accuracy for all 10 different supertag models with the integration strengths of 0 and 0.9. (Note that since all our models predict the edge label of each word, hard integration of perfect predictions eliminates the difference between labelled und unlabelled accuracy.) As expected, an improved accuracy of supertagging would lead to improved parsing accuracy in each case. In fact, knowing the correct supertag would solve the parsing problem almost completely with the more complex models. This confirms earlier findings for English (Nasr and Rambow, 2004).

Since perfect supertaggers are not available, we have to make do with the imperfect ones that do exist. One method of avoiding some errors introduced by supertagging would be to reject supertag predictions that tend to be wrong. To this end, we ran the supertagger on its training set and determined the average component accuracy of each occurring supertag. The supertags whose average precision fell below a variable threshold were not considered during parsing as if the supertagger had not made a prediction. This means that a threshold of 100% corresponds to the baseline of not using supertags at all, while a threshold of 0% prunes nothing, so that these two cases duplicate the first and last line from Table 2.

As Table 6 shows, pruning supertags that are wrong more often than they are right results in a further small improvement in parsing accuracy: unlabelled syntax accuracy rises up to 92.1% against the 91.8% if all supertags of model J are used. However, the effect is not very noticeable, so that it would be almost certainly more useful to

	Parsing accuracy	
Threshold	unlabelled	labelled
0%	91.8%	90.5%
20%	91.8%	90.4%
40%	91.9%	90.5%
50%	92.0%	90.7%
60%	92.1%	91.0%
80%	91.4%	90.0%
100%	89.6%	87.9%

Table 6: Parsing accuracy with empirically pruned supertag predictions.

improve the supertagger itself rather than second-guess its output.

5 Related work

Supertagging was originally suggested as a method to reduce lexical ambiguity, and thereby the amount of disambiguation work done by the parser. Sakar et al. (2000) report that this increases the speed of their LTAG parser by a factor of 26 (from 548k to 21k seconds) but at the price of only being able to parse 59% of the sentences in their test data (of 2250 sentences), because too often the correct supertag is missing from the output of the supertagger. Chen et al. (2002) investigate different supertagging methods as pre-processors to a Tree-Adjoining Grammar parser, and they claim a 1-best supertagging accuracy of 81.47%, and a 4-best accuracy of 91.41%. With the latter they reach the highest parser coverage, about three quarters of the 1700 sentences in their test data.

Clark and Curran (2004a; 2004b) describe a combination of supertagger and parser for parsing Combinatory Categorial Grammar, where the tagger is used to filter the parses produced by the grammar, before the computation of the model parameters. The parser uses an incremental method: the supertagger first assigns a small number of categories to each word, and the parser requests more alternatives only if the analysis fails. They report 91.4% precision and 91.0% recall of unlabelled dependencies and a speed of 1.6 minutes to parse 2401 sentences, and claim a parser speedup of a factor of 77 thanks to supertagging.

The supertagging approach that is closest to ours in terms of linguistic representations is probably (Wang and Harper, 2002; Wang and Harper, 2004) whose ‘Super Abstract Role Values’ are very similar to our model F supertags (Table 2). It is interesting to note that they only report between 328 and 791 SuperARVs for different corpora, whereas we have 2026 category F supertags. Part of the difference is explained by our larger label set: 35, the same as the number of model A supertags in table 2 against their 24 (White, 2000, p. 50). Also, we are not using the same corpus. In addition to determining the optimal SuperARV sequence in isolation, Wang and Harper (2002) also combine the SuperARV *n*-gram probabilities with a dependency assignment probability into a dependency parser for English. A maximum tagging accuracy of 96.3% (for sentences up to 100 words) is achieved using a 4-gram *n*-best tagger producing the 100 best SuperARV sequences for a sentence. The tightly integrated model is able to determine 96.6% of SuperARVs correctly. The parser itself reaches a labelled precision of 92.6% and a labelled recall of 92.2% (Wang and Harper, 2004).

In general, the effect of supertagging in the other systems mentioned here is to reduce the ambiguity in the input to the parser and thereby increase its speed, in some cases dramatically. For us, supertagging decreases the speed slightly, because additional constraints means more work for the parser, and because our supertagger-parser integration is not yet optimal. On the other hand it gives us better parsing accuracy. Using a constraint penalty of 0.0 for the supertagger integration (c.f. Table 5) does speed up our parser several times, but would only be practical with very high tagging accuracy. An important point is that for some other systems, like (Sarkar et al., 2000) and (Chen et al., 2002), parsing is not actually feasible without the supertagging speedup.

6 Conclusions and future work

We have shown that a statistical supertagging component can significantly improve the parsing accuracy of a general-purpose dependency parser for German. The error rate among syntactic attachments can be reduced by 24% over an already competitive baseline. After all, the integration of the supertagging results helped to reach a quality level which compares favourably with the state-of-the-art in probabilistic dependency parsing for German as defined with 87.34%/90.38% labelled/unlabelled attachment accuracy on this years shared CoNLL task by (McDonald et al., 2005) (see (Foth and Menzel, 2006) for a more detailed comparison). Although the statistical model used in our system is rather simple-minded, it clearly captures at least some distributional characteristics of German text that the hand-written rules do not.

A crucial factor for success is the defeasible integration of the supertagging predictions via soft constraints. Rather than pursuing a strict filtering approach where supertagging errors are partially compensated by an *n*-best selection, we commit to only one supertag per word, but reduce its influence. Treating supertag predictions as weak preferences yields the best results. By measuring the accuracy of the different types of predictions made by complex supertags, different weights could also be assigned to the six new constraints.

Of the investigated supertag models, the most complex ones guide the parser best, although their own accuracy is not the best one, even when measured by the more pertinent component accuracy. Since purely statistical parsing methods do not reach comparable parsing accuracy on the same data, we assume that this trend does not continue indefinitely, but would stop at some point, perhaps already reached.

References

S. Bangalore and A. K. Joshi. 1999. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

T. Brants, R. Hendriks, S. Kramp, B. Krenn, C. Preis, W. Skut, and H. Uszkoreit. 1997. Das NEGRA-Annotationsschema. Technical report, Universität des Saarlandes, Computerlinguistik.

S. Brants, St. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER treebank. In *Proc. Workshop on Treebanks and Linguistic Theories*, Sozopol.

T. Brants. 2000. TnT – A statistical part-of-speech tagger. In *Proc. the 6th Conf. on Applied Natural Language Processing, ANLP-2000*, pages 224–231, Seattle, WA.

J. Chen, S. Bangalore, M. Collins, and O. Rambow. 2002. Reranking an N-gram supertagger. In *Proc. 6th Int. Workshop on Tree Adjoining Grammar and Related Frameworks*.

S. Clark and J. R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *Proc. 20th Int. Conf. on Computational Linguistics*.

S. Clark and J. R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proc. 42nd Meeting of the ACL*.

M. Daum, K. Foth, and W. Menzel. 2003. Constraint based integration of deep and shallow parsing techniques. In *Proc. 11th Conf. of the EACL*, Budapest, Hungary.

M. Daum, K. Foth, and W. Menzel. 2004. Automatic transformation of phrase treebanks to dependency trees. In *Proc. 4th Int. Conf. on Language Resources and Evaluation, LREC-2004*, pages 99–106, Lisbon, Portugal.

K. Foth and J. Hagenström. 2002. Tagging for robust parsers. In *2nd Workshop on Robust Methods in Analysis of Natural Language Data, ROMAND-2002*, pages 21 – 32, Frascati, Italy.

K. Foth and W. Menzel. 2006. Hybrid parsing: Using probabilistic models as predictors for a symbolic parser. In *Proc. 21st Int. Conf. on Computational Linguistics, Coling-ACL-2006*, Sydney.

K. Foth, M. Daum, and W. Menzel. 2004. A broad-coverage parser for german based on defeasible constraints. In *7. Konferenz zur Verarbeitung natürlicher Sprache, KONVENS-2004*, pages 45–52, Wien.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. Human Language Technology Conference, HLT/EMNLP-2005*, Vancouver, B.C.

A. Nasr and O. Rambow. 2004. A simple string-rewriting formalism for dependency grammar. In *Coling-Workshop Recent Advances in Dependency Grammar*, pages 17–24, Geneva, Switzerland.

A. Sarkar, F. Xia, and A. Joshi. 2000. Some experiments on indicators of parsing complexity for lexicalized grammars. In *Proc. COLING Workshop on Efficiency in Large-Scale Parsing Systems*.

Y. Schabes and A. K. Joshi. 1991. Parsing with lexicalized tree adjoining grammar. In M. Tomita, editor, *Current Issues in Parsing Technologies*. Kluwer Academic Publishers.

I. Schröder, W. Menzel, K. Foth, and M. Schulz. 2000. Modeling dependency grammar with restricted constraints. *Traitement Automatique des Langues (T.A.L.)*, 41(1):97–126.

W. Wang and M. P. Harper. 2002. The SuperARV language model: Investigating the effectiveness of tightly integrating multiple knowledge sources. In *Proc. Conf. on Empirical Methods in Natural Language Processing, EMNLP-2002*, pages 238–247, Philadelphia, PA.

W. Wang and M. P. Harper. 2004. A statistical constraint dependency grammar (CDG) parser. In *Proc. ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*, pages 42–49, Barcelona, Spain.

Ch. M. White. 2000. *Rapid Grammar Development and Parsing: Constraint Dependency Grammar with Abstract Role Values*. Ph.D. thesis, Purdue University, West Lafayette, IN.

Efficient Unsupervised Discovery of Word Categories Using Symmetric Patterns and High Frequency Words

Dmitry Davidov
ICNC
The Hebrew University
Jerusalem 91904, Israel
dmitry@alice.nc.huji.ac.il

Ari Rappoport
Institute of Computer Science
The Hebrew University
Jerusalem 91904, Israel
www.cs.huji.ac.il/∼arir

Abstract

We present a novel approach for discovering word categories, sets of words sharing a significant aspect of their meaning. We utilize meta-patterns of high-frequency words and content words in order to discover pattern candidates. Symmetric patterns are then identified using graph-based measures, and word categories are created based on graph clique sets. Our method is the first pattern-based method that requires no corpus annotation or manually provided seed patterns or words. We evaluate our algorithm on very large corpora in two languages, using both human judgments and WordNet-based evaluation. Our fully unsupervised results are superior to previous work that used a POS tagged corpus, and computation time for huge corpora are orders of magnitude faster than previously reported.

1 Introduction

Lexical resources are crucial in most NLP tasks and are extensively used by people. Manual compilation of lexical resources is labor intensive, error prone, and susceptible to arbitrary human decisions. Hence there is a need for automatic authoring that would be as unsupervised and language-independent as possible.

An important type of lexical resource is that given by grouping words into categories. In general, the notion of a *category* is a fundamental one in cognitive psychology (Matlin, 2005). A *lexical category* is a set of words that share a significant aspect of their meaning, e.g., sets of words denoting vehicles, types of food, tool names, etc. A word can obviously belong to more than a single category. We will use ‘category’ instead of ‘lexical category’ for brevity[1].

Grouping of words into categories is useful in itself (e.g., for the construction of thesauri), and can serve as the starting point in many applications, such as ontology construction and enhancement, discovery of verb subcategorization frames, etc.

Our goal in this paper is a fully unsupervised discovery of categories from large unannotated text corpora. We aim for categories containing single words (multi-word lexical items will be dealt with in future papers.) Our approach is based on patterns, and utilizes the following stages:

1. Discovery of a set of pattern candidates that might be useful for induction of lexical relationships. We do this in a fully unsupervised manner, using meta-patterns comprised of *high frequency words* and *content words.*

2. Identification of pattern candidates that give rise to *symmetric* lexical relationships. This is done using simple measures in a word relationship graph.

3. Usage of a novel graph clique-set algorithm in order to generate categories from information on the co-occurrence of content words in the symmetric patterns.

We performed a thorough evaluation on two English corpora (the BNC and a 68GB web corpus) and on a 33GB Russian corpus, and a sanity-check test on smaller Danish, Irish and Portuguese corpora. Evaluations were done using both human

[1] Some people use the term ‘concept’. We adhere to the cognitive psychology terminology, in which ‘concept’ refers to the mental representation of a category (Matlin, 2005).

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 297–304, Sydney, July 2006.

judgments and WordNet in a setting quite similar to that done (for the BNC) in previous work. Our unsupervised results are superior to previous work that used a POS tagged corpus, are less language dependent, and are very efficient computationally[2].

Patterns are a common approach in lexical acquisition. Our approach is novel in several aspects: (1) we discover patterns in a fully unsupervised manner, as opposed to using a manually prepared pattern set, pattern seed or words seeds; (2) our pattern discovery requires no annotation of the input corpus, as opposed to requiring POS tagging or partial or full parsing; (3) we discover general symmetric patterns, as opposed to using a few hard-coded ones such as 'x and y'; (4) the clique-set graph algorithm in stage 3 is novel. In addition, we demonstrated the relatively language independent nature of our approach by evaluating on very large corpora in two languages[3].

Section 2 surveys previous work. Section 3 describes pattern discovery, and Section 4 describes the formation of categories. Evaluation is presented in Section 5, and a discussion in Section 6.

2 Previous Work

Much work has been done on lexical acquisition of all sorts. The three main distinguishing axes are (1) the type of corpus annotation and other human input used; (2) the type of lexical relationship targeted; and (3) the basic algorithmic approach. The two main approaches are pattern-based discovery and clustering of context feature vectors.

Many of the papers cited below aim at the construction of hyponym (is-a) hierarchies. Note that they can also be viewed as algorithms for category discovery, because a subtree in such a hierarchy defines a lexical category.

A first major algorithmic approach is to represent word contexts as vectors in some space and use similarity measures and automatic clustering in that space (Curran and Moens, 2002). Pereira (1993) and Lin (1998) use syntactic features in the vector definition. (Pantel and Lin, 2002) improves on the latter by clustering by committee. Caraballo (1999) uses conjunction and appositive annotations in the vector representation.

The only previous works addressing our problem and not requiring any syntactic annotation are those that decompose a lexically-defined matrix (by SVD, PCA etc), e.g. (Schütze, 1998; Deerwester et al, 1990). Such matrix decomposition is computationally heavy and has not been proven to scale well when the number of words assigned to categories grows.

Agglomerative clustering (e.g., (Brown et al, 1992; Li, 1996)) can produce hierarchical word categories from an unannotated corpus. However, we are not aware of work in this direction that has been evaluated with good results on lexical category acquisition. The technique is also quite demanding computationally.

The second main algorithmic approach is to use lexico-syntactic patterns. Patterns have been shown to produce more accurate results than feature vectors, at a lower computational cost on large corpora (Pantel et al, 2004). Hearst (1992) uses a manually prepared set of initial lexical patterns in order to discover hierarchical categories, and utilizes those categories in order to automatically discover additional patterns.

(Berland and Charniak, 1999) use hand crafted patterns to discover part-of (meronymy) relationships, and (Chklovski and Pantel, 2004) discover various interesting relations between verbs. Both use information obtained by parsing. (Pantel et al, 2004) reduce the depth of the linguistic data used but still requires POS tagging.

Many papers directly target specific applications, and build lexical resources as a side effect. Named Entity Recognition can be viewed as an instance of our problem where the desired categories contain words that are names of entities of a particular kind, as done in (Freitag, 2004) using co-clustering. Many Information Extraction papers discover relationships between words using syntactic patterns (Riloff and Jones, 1999).

(Widdows and Dorow, 2002; Dorow et al, 2005) discover categories using two hard-coded symmetric patterns, and are thus the closest to us. They also introduce an elegant graph representation that we adopted. They report good results. However, they require POS tagging of the corpus, use only two hard-coded patterns ('x and y', 'x or y'), deal only with nouns, and require non-trivial computations on the graph.

A third, less common, approach uses set-theoretic inference, for example (Cimiano et al,

[2] We did not compare against methods that use richer syntactic information, both because they are supervised and because they are much more computationally demanding.

[3] We are not aware of any multilingual evaluation previously reported on the task.

2005). Again, that paper uses syntactic information.

In summary, no previous work has combined the accuracy, scalability and performance advantages of patterns with the fully unsupervised, unannotated nature possible with clustering approaches. This severely limits the applicability of previous work on the huge corpora available at present.

3 Discovery of Patterns

Our first step is the discovery of patterns that are useful for lexical category acquisition. We use two main stages: discovery of pattern candidates, and identification of the symmetric patterns among the candidates.

3.1 Pattern Candidates

An examination of the patterns found useful in previous work shows that they contain one or more very frequent word, such as 'and', 'is', etc. Our approach towards unsupervised pattern induction is to find such words and utilize them.

We define a *high frequency word (HFW)* as a word appearing more than T_H times per million words, and a *content word (CW)* as a word appearing less than T_C times per a million words[4].

Now define a *meta-pattern* as any sequence of HFWs and CWs. In this paper we require that meta-patterns obey the following constraints: (1) at most 4 words; (2) exactly two content words; (3) no two consecutive CWs. The rationale is to see what can be achieved using relatively short patterns and where the discovered categories contain single words only. We will relax these constraints in future papers. Our meta-patterns here are thus of four types: CHC, CHCH, CHHC, and HCHC.

In order to focus on patterns that are more likely to provide high quality categories, we removed patterns that appear in the corpus less than T_P times per million words. Since we can ensure that the number of HFWs is bounded, the total number of pattern candidates is bounded as well. Hence, this stage can be computed in time linear in the size of the corpus (assuming the corpus has been already pre-processed to allow direct access to a word by its index.)

3.2 Symmetric Patterns

Many of the pattern candidates discovered in the previous stage are not usable. In order to find a usable subset, we focus on the symmetric patterns. Our rationale is that two content-bearing words that appear in a symmetric pattern are likely to be semantically similar in some sense. This simple observation turns out to be very powerful, as shown by our results. We will eventually combine data from several patterns and from different corpus windows (Section 4.)

For identifying symmetric patterns, we use a version of the graph representation of (Widdows and Dorow, 2002). We first define the *single-pattern graph* $G(P)$ as follows. Nodes correspond to content words, and there is a directed arc $A(x, y)$ from node x to node y iff (1) the words x and y both appear in an instance of the pattern P as its two CWs; and (2) x precedes y in P. Denote by $Nodes(G), Arcs(G)$ the nodes and arcs in a graph G, respectively.

We now compute three measures on $G(P)$ and combine them for all pattern candidates to filter asymmetric ones. The first measure (M_1) counts the proportion of words that can appear in both slots of the pattern, out of the total number of words. The reasoning here is that if a pattern allows a large percentage of words to participate in both slots, its chances of being a symmetric pattern are greater:

$$M_1 := \frac{|\{x|\exists y A(x,y) \wedge \exists z A(z,x)\}|}{|Nodes(G(P))|}$$

M_1 filters well patterns that connect words having different parts of speech. However, it may fail to filter patterns that contain multiple levels of asymmetric relationships. For example, in the pattern 'x belongs to y', we may find a word B on both sides ('A belongs to B', 'B belongs to C') while the pattern is still asymmetric.

In order to detect symmetric relationships in a finer manner, for the second and third measures we define $SymG(P)$, the symmetric subgraph of $G(P)$, containing only the bidirectional arcs and nodes of $G(P)$:

$$SymG(P) = \{\{x\}, \{(x,y)\}|A(x,y) \wedge A(y,x)\}$$

The second and third measures count the proportion of the number of symmetric nodes and edges in $G(P)$, respectively:

$$M_2 := \frac{|Nodes(SymG(P))|}{|Nodes(G(P))|}$$

[4] Considerations for the selection of thresholds are discussed in Section 5.

$$M_3 := \frac{|Arcs(SymG(P))|}{|Arcs(G(P))|}$$

All three measures yield values in $[0,1]$, and in all three a higher value indicates more symmetry. M_2 and M_3 are obviously correlated, but they capture different aspects of a pattern's nature: M_3 is informative for highly interconnected but small word categories (e.g., month names), while M_2 is useful for larger categories that are more loosely connected in the corpus.

We use the three measures as follows. For each measure, we prepare a sorted list of all candidate patterns. We remove patterns that are not in the top Z_T (we use 100, see Section 5) in any of the three lists, and patterns that are in the bottom Z_B in at least one of the lists. The remaining patterns constitute our final list of symmetric patterns.

We do not rank the final list, since the category discovery algorithm of the next section does not need such a ranking. Defining and utilizing such a ranking is a subject for future work.

A sparse matrix representation of each graph can be computed in time linear in the size of the input corpus, since (1) the number of patterns $|P|$ is bounded, (2) vocabulary size $|V|$ (the total number of graph nodes) is much smaller than corpus size, and (3) the average node degree is much smaller than $|V|$ (in practice, with the thresholds used, it is a small constant.)

4 Discovery of Categories

After the end of the previous stage we have a set of symmetric patterns. We now use them in order to discover categories. In this section we describe the graph clique-set method for generating initial categories, and category pruning techniques for increased quality.

4.1 The Clique-Set Method

Our approach to category discovery is based on connectivity structures in the all-pattern word relationship graph G, resulting from merging all of the single-pattern graphs into a single unified graph. The graph G can be built in time $O(|V| \times |P| \times AverageDegree(G(P))) = O(|V|)$ (we use V rather than $Nodes(G)$ for brevity.)

When building G, no special treatment is done when one pattern is contained within another. For example, any pattern of the form CHC is contained in a pattern of the form HCHC ('x and y', 'both x and y'.) The shared part yields exactly the same subgraph. This policy could be changed for a discovery of finer relationships.

The main observation on G is that words that are highly interconnected are good candidates to form a category. This is the same general observation exploited by (Widdows and Dorow, 2002), who try to find graph regions that are more connected internally than externally.

We use a different algorithm. We find all strong n-cliques (subgraphs containing n nodes that are all bidirectionally interconnected.) A clique Q defines a category that contains the nodes in Q plus all of the nodes that are (1) at least unidirectionally connected to all nodes in Q, and (2) bidirectionally connected to at least one node in Q.

In practice we use 2-cliques. The strongly connected cliques are the bidirectional arcs in G and their nodes. For each such arc A, a category is generated that contains the nodes of all triangles that contain A and at least one additional bidirectional arc. For example, suppose the corpus contains the text fragments 'book and newspaper', 'newspaper and book', 'book and note', 'note and book' and 'note and newspaper'. In this case the three words are assigned to a category.

Note that a pair of nodes connected by a symmetric arc can appear in more than a single category. For example, suppose a graph G containing five nodes and seven arcs that define exactly three strongly connected triangles, ABC, ABD, ACE. The arc (A, B) yields a category $\{A, B, C, D\}$, and the arc (A, C) yields a category $\{A, C, B, E\}$. Nodes A and C appear in both categories. Category merging is described below.

This stage requires an $O(1)$ computation for each bidirectional arc of each node, so its complexity is $O(|V| \times AverageDegree(G)) = O(|V|)$.

4.2 Enhancing Category Quality: Category Merging and Corpus Windowing

In order to cover as many words as possible, we use the smallest clique, a single symmetric arc. This creates redundant categories. We enhance the quality of the categories by merging them and by windowing on the corpus.

We use two simple merge heuristics. First, if two categories are identical we treat them as one. Second, given two categories Q, R, we merge them iff there's more than a 50% overlap between them: $(|Q \bigcap R| > |Q|/2) \wedge (|Q \bigcap R| > |R|/2)$.

This could be added to the clique-set stage, but the phrasing above is simpler to explain and implement.

In order to increase category quality and remove categories that are too context-specific, we use a simple corpus windowing technique. Instead of running the algorithm of this section on the whole corpus, we divide the corpus into windows of equal size (see Section 5 for size determination) and perform the category discovery algorithm of this section on each window independently. Merging is also performed in each window separately. We now have a set of categories for each window. For the final set, we select only those categories that appear in at least two of the windows. This technique reduces noise at the potential cost of lowering coverage. However, the numbers of categories discovered and words they contain is still very large (see Section 5), so windowing achieves higher precision without hurting coverage in practice.

The complexity of the merge stage is $O(|V|)$ times the average number of categories per word times the average number of words per category. The latter two are small in practice, so complexity amounts to $O(|V|)$.

5 Evaluation

Lexical acquisition algorithms are notoriously hard to evaluate. We have attempted to be as thorough as possible, using several languages and both automatic and human evaluation. In the automatic part, we followed as closely as possible the methodology and data used in previous work, so that meaningful comparisons could be made.

5.1 Languages and Corpora

We performed in-depth evaluation on two languages, English and Russian, using three corpora, two for English and one for Russian. The first English corpus is the BNC, containing about 100M words. The second English corpus, Dmoz (Gabrilovich and Markovitch, 2005), is a web corpus obtained by crawling and cleaning the URLs in the Open Directory Project (dmoz.org), resulting in 68GB containing about 8.2G words from 50M web pages.

The Russian corpus was assembled from many web sites and carefully filtered for duplicates, to yield 33GB and 4G words. It is a varied corpus comprising literature, technical texts, news, newsgroups, etc.

As a preliminary sanity-check test we also applied our method to smaller corpora in Danish, Irish and Portuguese, and noted some substantial similarities in the discovered patterns. For example, in all 5 languages the pattern corresponding to 'x and y' was among the 50 selected.

5.2 Thresholds, Statistics and Examples

The thresholds T_H, T_C, T_P, Z_T, Z_B, were determined by memory size considerations: we computed thresholds that would give us the maximal number of words, while enabling the pattern access table to reside in main memory. The resulting numbers are $100, 50, 20, 100, 100$.

Corpus window size was determined by starting from a very small window size, defining at random a single window of that size, running the algorithm, and iterating this process with increased window sizes until reaching a desired vocabulary category participation percentage (i.e., x% of the different words in the corpus assigned into categories. We used 5%.) This process has only a negligible effect on running times, because each iteration is run only on a single window, not on the whole corpus.

The table below gives some statistics. V is the total number of different words in the corpus. W is the number of words belonging to at least one of our categories. C is the number of categories (after merging and windowing.) AS is the average category size. Running times are in minutes on a 2.53Ghz Pentium 4 XP machine with $1GB$ memory. Note how small they are, when compared to (Pantel et al, 2004), which took 4 days for a smaller corpus using the same CPU.

	V	W	C	AS	Time
Dmoz	16M	330K	142K	12.8	93m
BNC	337K	25K	9.6K	10.2	6.8m
Russian	10M	235K	115K	11.6	60m

Among the patterns discovered are the ubiquitous 'x and y', 'x or y' and many patterns containing them. Additional patterns include 'from x to y', 'x and/or y' (punctuation is treated here as white space), 'x and a y', and 'neither x nor y'.

We discover categories of different parts of speech. Among the noun ones, there are many whose precision is 100%: 37 countries, 18 languages, 51 chemical elements, 62 animals, 28 types of meat, 19 fruits, 32 university names, etc. A nice verb category example is {*dive, snorkel, swim, float, surf, sail, canoe, kayak, paddle, tube, drift*}. A nice adjective example is {*amazing,* *awesome, fascinating, inspiring, inspirational, exciting, fantastic, breathtaking, gorgeous.*}

5.3 Human Judgment Evaluation

The purpose of the human evaluation was dual: to assess the quality of the discovered categories in terms of precision, and to compare with those obtained by a baseline clustering algorithm.

For the baseline, we implemented k-means as follows. We have removed stopwords from the corpus, and then used as features the words which appear before or after the target word. In the calculation of feature values and inter-vector distances, and in the removal of less informative features, we have strictly followed (Pantel and Lin, 2002). We ran the algorithm 10 times using $k = 500$ with randomly selected centroids, producing 5000 clusters. We then merged the resulting clusters using the same 50% overlap criterion as in our algorithm. The result included 3090, 2116, and 3206 clusters for Dmoz, BNC and Russian respectively.

We used 8 subjects for evaluation of the English categories and 15 subjects for evaluation of the Russian ones. In order to assess the subjects' reliability, we also included random categories (see below.)

The experiment contained two parts. In Part I, subjects were given 40 triplets of words and were asked to rank them using the following scale: (1) the words definitely share a significant part of their meaning; (2) the words have a shared meaning but only in some context; (3) the words have a shared meaning only under a very unusual context/situation; (4) the words do not share any meaning; (5) I am not familiar enough with some/all of the words.

The 40 triplets were obtained as follows. 20 of our categories were selected at random from the non-overlapping categories we have discovered, and three words were selected from each of these at random. 10 triplets were selected in the same manner from the categories produced by k-means, and 10 triplets were generated by random selection of content words from the same window in the corpus.

In Part II, subjects were given the full categories of the triplets that were graded as 1 or 2 in Part I (that is, the full 'good' categories in terms of sharing of meaning.) They were asked to grade the categories from 1 (worst) to 10 (best) according to how much the full category had met the expectations they had when seeing only the triplet.

Results are given in Table 1. The first line gives the average percentage of triplets that were given scores of 1 or 2 (that is, 'significant shared meaning'.) The 2nd line gives the average score of a triplet (1 is best.) In these lines scores of 5 were not counted. The 3rd line gives the average score given to a full category (10 is best.) Inter-evaluator Kappa between scores 1,2 and 3,4 was 0.56, 0.67 and 0.72 for Dmoz, BNC and Russian respectively.

Our algorithm clearly outperforms k-means, which outperforms random. We believe that the Russian results are better because the percentage of native speakers among our subjects for Russian was larger than that for English.

5.4 WordNet-Based Evaluation

The major guideline in this part of the evaluation was to compare our results with previous work having a similar goal (Widdows and Dorow, 2002). We have followed their methodology as best as we could, using the same WordNet (WN) categories and the same corpus (BNC) in addition to the Dmoz and Russian corpora[5].

The evaluation method is as follows. We took the exact 10 WN subsets referred to as 'subjects' in (Widdows and Dorow, 2002), and removed all multi-word items. We now selected at random 10 pairs of words from each subject. For each pair, we found the largest of our discovered categories containing it (if there isn't one, we pick another pair. This is valid because our Recall is obviously not even close to 100%, so if we did not pick another pair we would seriously harm the validity of the evaluation.) The various morphological forms of the same word were treated as one during the evaluation.

The only difference from the (Widdows and Dorow, 2002) experiment is the usage of pairs rather than single words. We did this in order to disambiguate our categories. This was not needed in (Widdows and Dorow, 2002) because they had directly accessed the word graph, which may be an advantage in some applications.

The Russian evaluation posed a bit of a problem because the Russian WordNet is not readily available and its coverage is rather small. Fortunately, the subject list is such that WordNet words

[5] (Widdows and Dorow, 2002) also reports results for an LSA-based clustering algorithm that are vastly inferior to the pattern-based ones.

	Dmoz			BNC			Russian		
	us	k-means	random	us	k-means	random	us	k-means	random
avg ‘shared meaning’ (%)	**80.53**	18.25	1.43	**86.87**	8.52	0.00	**95.00**	18.96	7.33
avg triplet score (1-4)	**1.74**	3.34	3.88	**1.56**	3.61	3.94	**1.34**	3.32	3.76
avg category score (1-10)	**9.27**	4.00	1.8	**9.31**	4.50	0.00	**8.50**	4.66	3.32

Table 1: Results of evaluation by human judgment of three data sets (ours, that obtained by k-means, and random categories) on the three corpora. See text for detailed explanations.

could be translated unambiguously to Russian and words in our discovered categories could be translated unambiguously into English. This was the methodology taken.

For each found category C containing N words, we computed the following (see Table 2): (1) Precision: the number of words present in both C and WN divided by N; (2) Precision*: the number of correct words divided by N. Correct words are either words that appear in the WN subtree, or words whose entry in the American Heritage Dictionary or the Britannica directly defines them as belonging to the given class (e.g., ‘keyboard’ is defined as ‘a piano’; ‘mitt’ is defined by ‘a type of glove’.) This was done in order to overcome the relative poorness of WordNet; (3) Recall: the number of words present in both C and WN divided by the number of (single) words in WN; (4) The number of correctly discovered words (New) that are not in WN. The Table also shows the number of WN words (:WN), in order to get a feeling by how much WN could be improved here. For each subject, we show the average over the 10 randomly selected pairs.

Table 2 also shows the average of each measure over the subjects, and the two precision measures when computed on the total set of WN words. The (uncorrected) precision is the only metric given in (Widdows and Dorow, 2002), who reported 82% (for the BNC.) Our method gives 90.47% for this metric on the same corpus.

5.5 Summary

Our human-evaluated and WordNet-based results are better than the baseline and previous work respectively. Both are also of good standalone quality. Clearly, evaluation methodology for lexical acquisition tasks should be improved, which is an interesting research direction in itself.

Examining our categories at random, we found a nice example that shows how difficult it is to evaluate the task and how useful automatic category discovery can be, as opposed to manual definition. Consider the following category, discovered in the Dmoz corpus: $\{$*nightcrawlers, chicken, shrimp, liver, leeches*$\}$. We did not know why these words were grouped together; if asked in an evaluation, we would give the category a very low score. However, after some web search, we found that this is a ‘fish bait’ category, especially suitable for catfish.

6 Discussion

We have presented a novel method for pattern-based discovery of lexical semantic categories. It is the first pattern-based lexical acquisition method that is fully unsupervised, requiring no corpus annotation or manually provided patterns or words. Pattern candidates are discovered using meta-patterns of high frequency and content words, and symmetric patterns are discovered using simple graph-theoretic measures. Categories are generated using a novel graph clique-set algorithm. The only other fully unsupervised lexical category acquisition approach is based on decomposition of a matrix defined by context feature vectors, and it has not been shown to scale well yet. Our algorithm was evaluated using both human judgment and automatic comparisons with WordNet, and results were superior to previous work (although it used a POS tagged corpus) and more efficient computationally. Our algorithm is also easy to implement.

Computational efficiency and specifically lack of annotation are important criteria, because they allow usage of huge corpora, which are presently becoming available and growing in size.

There are many directions to pursue in the future: (1) support multi-word lexical items; (2) increase category quality by improved merge algorithms; (3) discover various relationships (e.g., hyponymy) between the discovered categories; (4) discover finer inter-word relationships, such as verb selection preferences; (5) study various properties of discovered patterns in a detailed manner; and (6) adapt the algorithm to morphologically rich languages.

Subject	Prec.	Prec.*	Rec.	New:WN
Dmoz				
instruments	79.25	89.34	34.54	7.2:163
vehicles	80.17	86.84	18.35	6.3:407
academic	78.78	89.32	30.83	15.5:396
body parts	73.85	79.29	5.95	9.1:1491
foodstuff	83.94	90.51	28.41	26.3:1209
clothes	83.41	89.43	10.65	4.5:539
tools	83.99	89.91	21.69	4.3:219
places	76.96	84.45	25.82	6.3:232
crimes	76.32	86.99	31.86	4.7:102
diseases	81.33	88.99	19.58	6.8:332
set avg	79.80	87.51	22.77	9.1:509
all words	79.32	86.94		
BNC				
instruments	92.68	95.43	9.51	0.6:163
vehicles	94.16	95.23	3.81	0.2:407
academic	93.45	96.10	12.02	0.6:396
body parts	96.38	97.60	0.97	0.3:1491
foodstuff	93.76	94.36	3.60	0.6:1209
cloths	93.49	94.90	4.04	0.3:539
tools	96.84	97.24	6.67	0.1:219
places	87.88	97.25	6.42	1.5:232
crimes	83.79	91.99	19.61	2.6:102
diseases	95.16	97.14	5.54	0.5:332
set avg	92.76	95.72	7.22	0.73:509
all words	90.47	93.80		
Russian				
instruments	82.46	89.09	25.28	3.4:163
vehicles	83.16	89.58	16.31	5.1:407
academic	87.27	92.92	15.71	4.9:396
body parts	81.42	89.68	3.94	8.3:1491
foodstuff	80.34	89.23	13.41	24.3:1209
clothes	82.47	87.75	15.94	5.1:539
tools	79.69	86.98	21.14	3.7:219
places	82.25	90.20	33.66	8.5:232
crimes	84.77	93.26	34.22	3.3:102
diseases	80.11	87.70	20.69	7.7:332
set avg	82.39	89.64	20.03	7.43:509
all words	80.67	89.17		

Table 2: WordNet evaluation. Note the BNC 'all words' precision of 90.47%. This metric was reported to be 82% in (Widdows and Dorow, 2002).

It should be noted that our algorithm can be viewed as one for automatic discovery of word senses, because it allows a word to participate in more than a single category. When merged properly, the different categories containing a word can be viewed as the set of its senses. We are planning an evaluation according to this measure after improving the merge stage.

References

Matthew Berland and Eugene Charniak, 1999. Finding parts in very large corpora. ACL '99.

Peter Brown, Vincent Della Pietra, Peter deSouza, Jenifer Lai, Robert Mercer, 1992. Class-based n-gram models for natural language. *Comp. Linguistics*, 18(4):468–479.

Sharon Caraballo, 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. ACL '99.

Timothy Chklovski, Patrick Pantel, 2004. VerbOcean: mining the web for fine-grained semantic verb relations. EMNLP '04.

Philipp Cimiano, Andreas Hotho, Steffen Staab, 2005. Learning concept hierarchies from text corpora using formal concept analysis. *J. of Artificial Intelligence Research*, 24:305–339.

James Curran, Marc Moens, 2002. Improvements in automatic thesaurus extraction. ACL Workshop on Unsupervised Lexical Acquisition, 2002.

Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, Richard Harshman, 1990. Indexing by latent semantic analysis. *J. of the American Society for Info. Science*, 41(6):391–407.

Beate Dorow, Dominic Widdows, Katarina Ling, Jean-Pierre Eckmann, Danilo Sergi, Elisha Moses, 2005. Using curvature and Markov clustering in graphs for lexical acquisition and word sense discrimination. MEANING '05.

Dayne Freitag, 2004. Trained named entity recognition using distributional clusters. EMNLP '04.

Evgeniy Gabrilovich, Shaul Markovitch, 2005. Feature generation for text categorization using world knowledge. IJCAI '05.

Marti Hearst, 1992. Automatic acquisition of hyponyms from large text corpora. COLING '92.

Hang Li, Naoki Abe, 1996. Clustering words with the MDL principle. COLING '96.

Dekang Lin, 1998. Automatic retrieval and clustering of similar words. COLING '98.

Margaret Matlin, 2005. *Cognition, 6th edition.* John Wiley & Sons.

Patrick Pantel, Dekang Lin, 2002. Discovering word senses from text. SIGKDD '02.

Patrick Pantel, Deepak Ravichandran, Eduard Hovy, 2004. Towards terascale knowledge acquisition. COLING '04.

Fernando Pereira, Naftali Tishby, Lillian Lee, 1993. Distributional clustering of English words. ACL '93.

Ellen Riloff, Rosie Jones, 1999. Learning dictionaries for information extraction by multi-level bootstrapping. AAAI '99.

Hinrich Schütze, 1998. Automatic word sense discrimination. *Comp. Linguistics*, 24(1):97–123.

Dominic Widdows, Beate Dorow, 2002. A graph model for unsupervised Lexical acquisition. COLING '02.

Bayesian Query-Focused Summarization

Hal Daumé III and **Daniel Marcu**
Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
me@hal3.name,marcu@isi.edu

Abstract

We present BAYESUM (for "Bayesian summarization"), a model for sentence extraction in query-focused summarization. BAYESUM leverages the common case in which multiple documents are relevant to a single query. Using these documents as reinforcement for query terms, BAYESUM is not afflicted by the paucity of information in short queries. We show that approximate inference in BAYESUM is possible on large data sets and results in a state-of-the-art summarization system. Furthermore, we show how BAYESUM can be understood as a justified query expansion technique in the language modeling for IR framework.

1 Introduction

We describe BAYESUM, an algorithm for performing query-focused summarization in the common case that there are many relevant documents for a given query. Given a query and a collection of relevant documents, our algorithm functions by asking itself the following question: what is it about these relevant documents that differentiates them from the *non*-relevant documents? BAYESUM can be seen as providing a statistical formulation of this exact question.

The key requirement of BAYESUM is that multiple relevant documents are known for the query in question. This is not a severe limitation. In two well-studied problems, it is the de-facto standard. In standard multidocument summarization (with or without a query), we have access to known relevant documents for some user need. Similarly, in the case of a web-search application, an underlying IR engine will retrieve multiple (presumably) relevant documents for a given query. For both of these tasks, BAYESUM performs well, even when the underlying retrieval model is noisy.

The idea of leveraging known relevant documents is known as query expansion in the information retrieval community, where it has been shown to be successful in ad hoc retrieval tasks. Viewed from the perspective of IR, our work can be interpreted in two ways. First, it can be seen as an *application* of query expansion to the summarization task (or, in IR terminology, passage retrieval); see (Liu and Croft, 2002; Murdock and Croft, 2005). Second, and more importantly, it can be seen as a method for query expansion in a non-ad-hoc manner. That is, BAYESUM is a statistically justified query expansion method in the language modeling for IR framework (Ponte and Croft, 1998).

2 Bayesian Query-Focused Summarization

In this section, we describe our Bayesian query-focused summarization model (BAYESUM). This task is very similar to the standard ad-hoc IR task, with the important distinction that we are comparing query models against *sentence models*, rather than against document models. The shortness of sentences means that one must do a good job of creating the query models.

To maintain generality, so that our model is applicable to any problem for which multiple relevant documents are known for a query, we formulate our model in terms of *relevance judgments*. For a collection of D documents and Q queries, we assume we have a $D \times Q$ binary matrix r, where $r_{dq} = 1$ if an only if document d is relevant to query q. In multidocument summarization, r_{dq} will be 1 exactly when d is in the document set corresponding to query q; in search-engine sum-

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 305–312,
Sydney, July 2006. ©2006 Association for Computational Linguistics

marization, it will be 1 exactly when d is returned by the search engine for query q.

2.1 Language Modeling for IR

BAYESUM is built on the concept of language models for information retrieval. The idea behind the language modeling techniques used in IR is to represent either queries or documents (or both) as probability distributions, and then use standard probabilistic techniques for comparing them. These probability distributions are almost always "bag of words" distributions that assign a probability to words from a fixed vocabulary $\mathcal{V}$.

One approach is to build a probability distribution for a given document, $p_d(\cdot)$, and to look at the probability of a query under that distribution: $p_d(q)$. Documents are ranked according to how *likely* they make the query (Ponte and Croft, 1998). Other researchers have built probability distributions over queries $p_q(\cdot)$ and ranked documents according to how likely they look under the query model: $p_q(d)$ (Lafferty and Zhai, 2001). A third approach builds a probability distribution $p_q(\cdot)$ for the query, a probability distribution $p_d(\cdot)$ for the document and then measures the *similarity* between these two distributions using KL divergence (Lavrenko et al., 2002):

$$\mathit{KL}\left(p_q \mid\mid p_d\right) = \sum_{w \in \mathcal{V}} p_q(w) \log \frac{p_q(w)}{p_d(w)} \qquad (1)$$

The KL divergence between two probability distributions is zero when they are identical and otherwise strictly positive. It implicitly assumes that both distributions p_q and p_d have the same *support*: they assign non-zero probability to exactly the same subset of $\mathcal{V}$; in order to account for this, the distributions p_q and p_d are smoothed against a background general English model. This final mode—the KL model—is the one on which BAYESUM is based.

2.2 Bayesian Statistical Model

In the language of information retrieval, the query-focused sentence extraction task boils down to estimating a good *query model*, $p_q(\cdot)$. Once we have such a model, we could estimate sentence models for each sentence in a relevant document, and rank the sentences according to Eq (1).

The BAYESUM system is based on the following model: we hypothesize that a sentence appears in a document because it is relevant to some query, because it provides background information about the document (but is not relevant to a known query) or simply because it contains useless, general English filler. Similarly, we model each word as appearing for one of those purposes. More specifically, our model assumes that each word can be assigned a discrete, exact source, such as "this word is relevant to query q_1" or "this word is general English." At the sentence level, however, sentences are assigned *degrees*: "this sentence is 60% about query q_1, 30% background document information, and 10% general English."

To model this, we define a general English language model, $p^G(\cdot)$ to capture the English filler. Furthermore, for each document d_k, we define a background document language model, $p^{d_k}(\cdot)$; similarly, for each query q_j, we define a query-specific language model $p^{q_j}(\cdot)$. Every word in a document d_k is modeled as being generated from a mixture of p^G, p^{d_k} and $\{p^{q_j} :$ query q_j is relevant to document $d_k\}$. Supposing there are J total queries and K total documents, we say that the nth word from the sth sentence in document d, w_{dsn}, has a corresponding *hidden variable*, z_{dsn} that specifies exactly which of these distributions is used to generate that one word. In particular, z_{dsn} is a vector of length $1 + J + K$, where exactly one element is 1 and the rest are 0.

At the sentence level, we introduce a second layer of hidden variables. For the sth sentence in document d, we let π_{ds} be a vector also of length $1 + J + K$ that represents our degree of belief that this sentence came from any of the models. The π_{ds}s lie in the $J + K$-dimensional simplex $\Delta^{J+K} = \{\boldsymbol{\theta} = \langle \theta_1, \dots, \theta_{J+K+1} \rangle : (\forall i)\ \theta_i \geq 0, \sum_i \theta_i = 1\}$. The interpretation of the π variables is that if the "general English" component of π is 0.9, then 90% of the words in this sentence will be general English. The π and z variables are constrained so that a sentence cannot be generated by a document language model other than its own document and cannot be generated by a query language model for a query to which it is not relevant.

Since the πs are unknown, and it is unlikely that there is a "true" correct value, we place a corpus-level prior on them. Since π is a multinomial distribution over its corresponding zs, it is natural to use a Dirichlet distribution as a prior over π. A Dirichlet distribution is parameterized by a vector α of equal length to the corresponding multinomial parameter, again with the positivity restriction, but no longer required to sum to one. It has continuous density over a variable $\theta_1, \dots, \theta_I$ given by: $\mathcal{D}ir(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \prod_i \theta_i^{\alpha_i - 1}$. The first term is a normalization term that ensures that $\int_{\Delta^I} \mathrm{d}\boldsymbol{\theta}\, \mathcal{D}ir(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) = 1$.

2.3 Generative Story

The generative story for our model defines a distribution over a corpus of queries, $\{q_j\}_{1:J}$, and documents, $\{d_k\}_{1:K}$, as follows:

1. For each query $j = 1 \dots J$: Generate each word q_{jn} in q_j by $p^{q_j}(q_{jn})$
2. For each document $k = 1 \dots K$ and each sentence s in document k:
   (a) Select the current sentence degree π_{ks} by $\mathcal{D}ir(\pi_{ks} \mid \alpha) r_k(\pi_{ks})$
   (b) For each word w_{ksn} in sentence s:
      - Select the word source z_{ksn} according to $\mathcal{M}ult(z \mid \pi_{ks})$
      - Generate the word w_{ksn} by

$$\begin{cases} p^G(w_{ksn}) & \text{if } z_{ksn} = 0 \\ p^{d_k}(w_{ksn}) & \text{if } z_{ksn} = k + 1 \\ p^{q_j}(w_{ksn}) & \text{if } z_{ksn} = j + K + 1 \end{cases}$$

We used r to denote relevance judgments: $r_k(\pi) = 0$ if any document component of π *except* the one corresponding to k is non-zero, or if any query component of π *except* those queries to which document k is deemed relevant is non-zero (this prevents a document using the "wrong" document or query components). We have further assumed that the z vector is laid out so that z_0 corresponds to general English, z_{k+1} corresponds to document d_k for $0 \leq j < J$ and that z_{j+K+1} corresponds to query q_j for $0 \leq k < K$.

2.4 Graphical Model

The graphical model corresponding to this generative story is in Figure 1. This model depicts the four known parameters in square boxes (α, p^Q, p^D and p^G) with the three observed random variables in shaded circles (the queries q, the relevance judgments r and the words w) and two unobserved random variables in empty circles (the word-level indicator variables z and the sentence level degrees π). The rounded plates denote replication: there are J queries and K documents, containing S sentences in a given document and N words in a given sentence. The joint probability over the observed random variables is given in Eq (2):

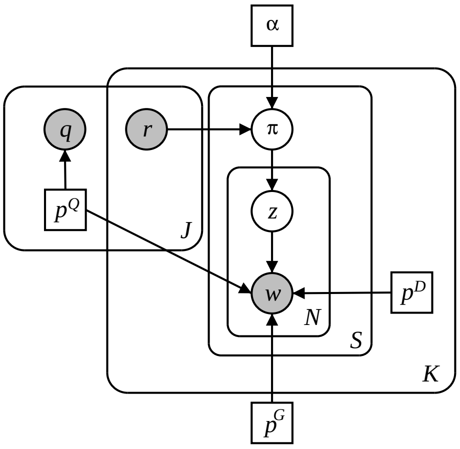


Figure 1: Graphical model for the Bayesian Query-Focused Summarization Model.

$$p\left(\boldsymbol{q}_{1:J}, r, \boldsymbol{d}_{1:K}\right) = \left[\prod_j \prod_n p^{q_j}\left(q_{jn}\right)\right] \times \qquad (2)$$
$$\left[\prod_k \prod_s \int_\Delta \mathrm{d}\pi_{ks}\, p\left(\pi_{ks} \mid \alpha, r\right)\right.$$
$$\left.\prod_n \sum_{z_{ksn}} p\left(z_{ksn} \mid \pi_{ks}\right) p\left(w_{ksn} \mid z_{ksn}\right)\right]$$

This expression computes the probability of the data by integrating out the unknown variables. In the case of the π variables, this is accomplished by integrating over Δ, the multinomial simplex, according to the prior distribution given by α. In the case of the z variables, this is accomplished by summing over all possible (discrete) values. The final word probability is conditioned on the z value by selecting the appropriate distribution from p^G, p^D and p^Q. Computing this expression and finding optimal model parameters is intractable due to the coupling of the variables under the integral.

3 Statistical Inference in BAYESUM

Bayesian inference problems often give rise to intractable integrals, and a large variety of techniques have been proposed to deal with this. The most popular are Markov Chain Monte Carlo (MCMC), the Laplace (or saddle-point) approximation and the variational approximation. A third, less common, but very effective technique, especially for dealing with mixture models, is expectation propagation (Minka, 2001). In this paper, we will focus on expectation propagation; experiments not reported here have shown variational

EM to perform comparably but take roughly 50% longer to converge.

Expectation propagation (EP) is an inference technique introduced by Minka (2001) as a generalization of both belief propagation and assumed density filtering. In his thesis, Minka showed that EP is very effective in mixture modeling problems, and later demonstrated its superiority to variational techniques in the Generative Aspect Model (Minka and Lafferty, 2003). The key idea is to compute an integral of a product of terms by iteratively applying a sequence of "deletion/inclusion" steps. Given an integral of the form: $\int_\Delta \mathrm{d}\boldsymbol{\pi}\ p(\pi) \prod_n t_n(\boldsymbol{\pi})$, EP approximates each term t_n by a simpler term $\tilde{t}_n$, giving Eq (3).

$$\int_\Delta \mathrm{d}\boldsymbol{\pi}\ q(\boldsymbol{\pi}) \qquad q(\boldsymbol{\pi}) = p(\boldsymbol{\pi}) \prod_n \tilde{t}_n(\boldsymbol{\pi}) \tag{3}$$

In each deletion/inclusion step, one of the approximate terms is deleted from $q(\cdot)$, leaving $q^{-n}(\cdot) = q(\cdot)/\tilde{t}_n(\cdot)$. A new approximation for $t_n(\cdot)$ is computed so that $t_n(\cdot)q^{-n}(\cdot)$ has the same integral, mean and variance as $\tilde{t}_n(\cdot)q^{-n}(\cdot)$. This new approximation, $\tilde{t}_n(\cdot)$ is then included back into the full expression for $q(\cdot)$ and the process repeats. This algorithm always has a fixed point and there are methods for ensuring that the approximation remains in a location where the integral is well-defined. Unlike variational EM, the approximation given by EP is global, and often leads to much more reliable estimates of the true integral.

In the case of our model, we follow Minka and Lafferty (2003), who adapts latent Dirichlet allocation of Blei et al. (2003) to EP. Due to space constraints, we omit the inference algorithms and instead direct the interested reader to the description given by Minka and Lafferty (2003).

4 Search-Engine Experiments

The first experiments we run are for query-focused single document summarization, where relevant documents are returned from a search engine, and a short summary is desired of each document.

4.1 Data

The data we use to train and test BAYESUM is drawn from the Text REtrieval Conference (TREC) competitions. This data set consists of queries, documents and relevance judgments, exactly as required by our model. The queries are typically broken down into four fields of increasing length: the title (3-4 words), the summary (1 sentence), the narrative (2-4 sentences) and the concepts (a list of keywords). Obviously, one would expect that the longer the query, the better a model would be able to do, and this is borne out experimentally (Section 4.5).

Of the TREC data, we have trained our model on 350 queries (queries numbered 51-350 and 401-450) and all corresponding relevant documents. This amounts to roughly $43k$ documents, $2.1m$ sentences and $65.8m$ words. The mean number of relevant documents per query is 137 and the median is 81 (the most prolific query has 968 relevant documents). On the other hand, each document is relevant to, on average, 1.11 queries (the median is 5.5 and the most generally relevant document is relevant to 20 different queries). In all cases, we apply stemming using the Porter stemmer; for all other models, we remove stop words.

In order to *evaluate* our model, we had seven human judges manually perform the query-focused sentence extraction task. The judges were supplied with the full TREC query and a single document relevant to that query, and were asked to select up to four sentences from the document that best met the needs given by the query. Each judge annotated 25 queries with some overlap to allow for an evaluation of inter-annotator agreement, yielding annotations for a total of 166 unique query/document pairs. On the doubly annotated data, we computed the inter-annotator agreement using the kappa measure. The kappa value found was 0.58, which is low, but not abysmal (also, keep in mind that this is computed over only 25 of the 166 examples).

4.2 Evaluation Criteria

Since there are differing numbers of sentences selected per document by the human judges, one cannot compute precision and recall; instead, we opt for other standard IR performance measures. We consider three related criteria: mean average precision (MAP), mean reciprocal rank (MRR) and precision at 2 (P@2). MAP is computed by calculating precision at every sentence as ordered by the system up until all relevant sentences are selected and averaged. MRR is the reciprocal of the rank of the first relevant sentence. P@2 is the precision computed at the first point that two relevant sentences have been selected (in the rare case that humans selected only one sentence, we use P@1).

4.3 Baseline Models

As baselines, we consider four strawman models and two state-of-the-art information retrieval models. The first strawman, RANDOM ranks sentences randomly. The second strawman, POSITION, ranks sentences according to their absolute position (in the context of non-query-focused summarization, this is an incredibly powerful baseline). The third and fourth models are based on the vector space interpretation of IR. The third model, JACCARD, uses standard Jaccard distance score (intersection over union) between each sentence and the query to rank sentences. The fourth, COSINE, uses TF-IDF weighted cosine similarity.

The two state-of-the-art IR models used as comparative systems are based on the language modeling framework described in Section 2.1. These systems compute a language model for each query and for each sentence in a document. Sentences are then ranked according to the KL divergence between the query model and the sentence model, smoothed against a general model estimated from the entire collection, as described in the case of document retrieval by Lavrenko et al. (2002). This is the first system we compare against, called KL.

The second true system, KL+REL is based on augmenting the KL system with blind relevance feedback (query expansion). Specifically, we first run each query against the document set returned by the relevance judgments and retrieve the top n sentences. We then expand the query by interpolating the original query model with a query model estimated on these sentences. This serves as a method of query expansion. We ran experiments ranging n in $\{5, 10, 25, 50, 100\}$ and the interpolation parameter λ in $\{0.2, 0.4, 0.6, 0.8\}$ and used oracle selection (on MRR) to choose the values that performed best (the results are thus overly optimistic). These values were $n = 25$ and $\lambda = 0.4$.

Of all the systems compared, only BAYESUM and the KL+REL model use the relevance judgments; however, they both have access to exactly the same information. The other models only run on the subset of the data used for evaluation (the corpus language model for the KL system and the IDF values for the COSINE model are computed on the full data set). EP ran for 2.5 hours.

	MAP	MRR	P@2
RANDOM	19.9	37.3	16.6
POSITION	24.8	41.6	19.9
JACCARD	17.9	29.3	16.7
COSINE	29.6	50.3	23.7
KL	36.6	64.1	27.6
KL+REL	36.3	62.9	29.2
BAYESUM	44.1	70.8	33.6

Table 1: Empirical results for the baseline models as well as BAYESUM, when all query fields are used.

4.4 Performance on all Query Fields

Our first evaluation compares results when all query fields are used (title, summary, description and concepts[1]). These results are shown in Table 1. As we can see from these results, the JACCARD system alone is not sufficient to beat the position-based baseline. The COSINE does beat the position baseline by a bit of a margin (5 points better in MAP, 9 points in MRR and 4 points in P@2), and is in turn beaten by the KL system (which is 7 points, 14 points and 4 points better in MAP, MRR and P@2, respectively). Blind relevance feedback (parameters of which were chosen by an oracle to maximize the P@2 metric) actually hurts MAP and MRR performance by 0.3 and 1.2, respectively, and increases P@2 by 1.5. Over the best performing baseline system (either KL or KL+REL), BAYESUM wins by a margin of 7.5 points in MAP, 6.7 for MRR and 4.4 for P@2.

4.5 Varying Query Fields

Our next experimental comparison has to do with reducing the amount of information given in the query. In Table 2, we show the performance of the KL, KL-REL and BAYESUM systems, as we use different query fields. There are several things to notice in these results. First, the standard KL model without blind relevance feedback performs worse than the position-based model when only the 3-4 word title is available. Second, BAYESUM using *only the title* outperform the KL model *with* relevance feedback using *all fields*. In fact, one can apply BAYESUM without using *any* of the query fields; in this case, only the relevance judgments are available to make sense

[1] A reviewer pointed out that concepts were later removed from TREC because they were "too good." Section 4.5 considers the case without the concepts field.

		MAP	MRR	P@2
Position		24.8	41.6	19.9
Title	KL	19.9	32.6	17.8
	KL-Rel	31.9	53.8	26.1
	BayeSum	41.1	65.7	31.6
+Description	KL	31.5	58.3	24.1
	KL-Rel	32.6	55.0	26.2
	BayeSum	40.9	66.9	31.0
+Summary	KL	31.6	56.9	23.8
	KL-Rel	34.2	48.5	27.0
	BayeSum	42.0	67.8	31.8
+Concepts	KL	36.7	64.2	27.6
	KL-Rel	36.3	62.9	29.2
	BayeSum	44.1	70.8	33.6
No Query	BayeSum	39.4	64.7	30.4

Table 2: Empirical results for the position-based model, the KL-based models and BayeSum, with different inputs.

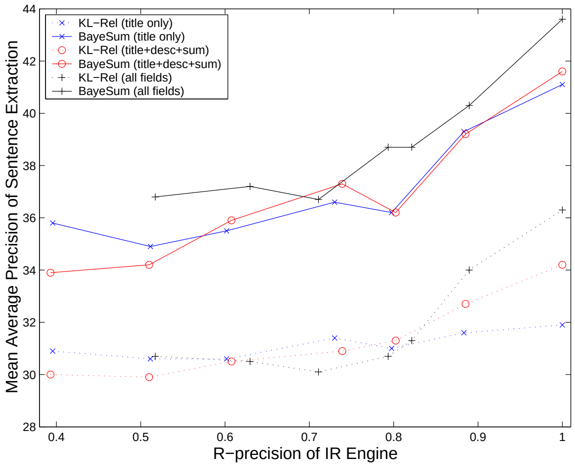


Figure 2: Performance with noisy relevance judgments. The X-axis is the R-precision of the IR engine and the Y-axis is the summarization performance in MAP. Solid lines are BayeSum, dotted lines are KL-Rel. Blue/stars indicate title only, red/circles indicated title+description+summary and black/pluses indicate all fields.

of what the query might be. Even in this circumstance, BayeSum achieves a MAP of 39.4, an MRR of 64.7 and a P@2 of 30.4, still better across the board than KL-Rel with all query fields. While initially this seems counterintuitive, it is actually not so unreasonable: there is significantly more information available in several hundred positive relevance judgments than in a few sentences. However, the simple blind relevance feedback mechanism so popular in IR is unable to adequately model this.

With the exception of the KL model without relevance feedback, adding the description on top of the title does not seem to make any difference for any of the models (and, in fact, occasionally hurts according to some metrics). Adding the summary improves performance in most cases, but not significantly. Adding concepts tends to improve results slightly more substantially than any other.

4.6 Noisy Relevance Judgments

Our model hinges on the assumption that, for a given query, we have access to a collection of *known relevant documents*. In most real-world cases, this assumption is violated. Even in multi-document summarization as run in the DUC competitions, the assumption of access to a collection of documents all relevant to a user need is unrealistic. In the real world, we will have to deal with document collections that "accidentally" contain irrelevant documents. The experiments in this section show that BayeSum is comparatively robust.

For this experiment, we use the IR engine that performed best in the TREC 1 evaluation: Inquery (Callan et al., 1992). We used the official TREC results of Inquery on the subset of the TREC corpus we consider. The Inquery R-precision on this task is 0.39 using title only, and 0.51 using all fields. In order to obtain curves as the IR engine improves, we have linearly interpolated the Inquery rankings with the true relevance judgments. By tweaking the interpolation parameter, we obtain an IR engine with improving performance, but with a reasonable bias. We have run both BayeSum and KL-Rel on the relevance judgments obtained by this method for six values of the interpolation parameter. The results are shown in Figure 2.

As we can observe from the figure, the solid lines (BayeSum) are always above the dotted lines (KL-Rel). Considering the KL-Rel results alone, we can see that for a non-perfect IR engine, it makes little difference what query fields we use for the summarization task: they all obtain roughly equal scores. This is because the performance in KL-Rel is dominated by the performance of the IR engine. Looking only at the BayeSum results, we can see a much stronger, and perhaps surprising difference. For an imperfect IR system, it is better to use only the title than to use the title, description and summary for the summarization component. We believe this is because the title is more on topic than the other fields, which contain terms like "A relevant document should describe . . . ." Nevertheless, BAYESUM has a more upward trend than KL-Rel, which indicates that improved IR will result in improved summarization for BAYESUM but not for KL-Rel.

5 Multidocument Experiments

We present two results using BAYESUM in the multidocument summarization settings, based on the official results from the Multilingual Summarization Evaluation (MSE) and Document Understanding Conference (DUC) competitions in 2005.

5.1 Performance at MSE 2005

We participated in the Multilingual Summarization Evaluation (MSE) workshop with a system based on BAYESUM. The task for this competition was generic (no query) multidocument summarization. Fortunately, not having a query is not a hindrance to our model. To account for the redundancy present in document collections, we applied a greedy selection technique that selects sentences central to the document cluster but far from previously selected sentences (Daumé III and Marcu, 2005a). In MSE, our system performed very well. According to the human "pyramid" evaluation, our system came first with a score of 0.529; the next best score was 0.489. In the automatic "Basic Element" evaluation, our system scored 0.0704 (with a 95% confidence interval of $[0.0429, 0.1057]$), which was the third best score on a site basis (out of 10 sites), and was not statistically significantly different from the best system, which scored 0.0981.

5.2 Performance at DUC 2005

We also participated in the Document Understanding Conference (DUC) competition. The chosen task for DUC was query-focused multidocument summarization. We entered a nearly identical system to DUC as to MSE, with an additional rule-based sentence compression component (Daumé III and Marcu, 2005b). Human evaluators considered both *responsiveness* (how well did the summary answer the query) and *linguistic quality*. Our system achieved the highest responsiveness score in the competition. We scored more poorly on the linguistic quality evaluation, which (only 5 out of about 30 systems performed worse); this is likely due to the sentence compression we performed on top of BAYESUM. On the automatic Rouge-based evaluations, our system performed between third and sixth (depending on the Rouge parameters), but was never statistically significantly worse than the best performing systems.

6 Discussion and Future Work

In this paper we have described a model for automatically generating a query-focused summary, when one has access to multiple relevance judgments. Our Bayesian Query-Focused Summarization model (BAYESUM) consistently outperforms contending, state of the art information retrieval models, even when it is forced to work with significantly less information (either in the complexity of the query terms or the quality of relevance judgments documents). When we applied our system as a stand-alone summarization model in the 2005 MSE and DUC tasks, we achieved among the highest scores in the evaluation metrics. The primary weakness of the model is that it currently only operates in a purely extractive setting.

One question that arises is: why does BAYESUM so strongly outperform KL-Rel, given that BAYESUM can be seen as Bayesian formalism for relevance feedback (query expansion)? Both models have access to exactly the same information: the queries and the true relevance judgments. This is especially interesting due to the fact that the two relevance feedback parameters for KL-Rel were chosen *optimally* in our experiments, yet BAYESUM consistently won out. One explanation for this performance win is that BAYESUM provides a separate weight for each word, for each query. This gives it significantly more flexibility. Doing something similar with ad-hoc query expansion techniques is difficult due to the enormous number of parameters; see, for instance, (Buckley and Salton, 1995).

One significant advantage of working in the Bayesian statistical framework is that it gives us a straightforward way to integrate other sources of knowledge into our model in a coherent manner. One could consider, for instance, to extend this model to the multi-document setting, where one would need to explicitly model redundancy across documents. Alternatively, one could include user models to account for novelty or user preferences along the lines of Zhang et al. (2002).

Our model is similar in spirit to the random-walk summarization model (Otterbacher et al., 2005). However, our model has several advantages over this technique. First, our model has no tunable parameters: the random-walk method has many (graph connectivity, various thresholds, choice of similarity metrics, etc.). Moreover, since our model is properly Bayesian, it is straightforward to extend it to model other aspects of the problem, or to related problems. Doing so in a non ad-hoc manner in the random-walk model would be nearly impossible.

Another interesting avenue of future work is to relax the bag-of-words assumption. Recent work has shown, in related models, how this can be done for moving from bag-of-words models to bag-of-$n$gram models (Wallach, 2006); more interesting than moving to $n$grams would be to move to dependency parse trees, which could likely be accounted for in a similar fashion. One could also potentially relax the assumption that the relevance judgments are known, and attempt to integrate them out as well, essentially simultaneously performing IR and summarization.

Acknowledgments. We thank Dave Blei and Tom Minka for discussions related to topic models, and to the anonymous reviewers, whose comments have been of great benefit. This work was partially supported by the National Science Foundation, Grant IIS-0326276.

References

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research (JMLR)*, 3:993–1022, January.

Chris Buckley and Gerard Salton. 1995. Optimization of relevance feedback weights. In *Proceedings of the Conference on Research and Developments in Information Retrieval (SIGIR)*.

Jamie Callan, Bruce Croft, and Stephen Harding. 1992. The INQUERY retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications*.

Hal Daumé III and Daniel Marcu. 2005a. Bayesian multi-document summarization at MSE. In *ACL 2005 Workshop on Intrinsic and Extrinsic Evaluation Measures*.

Hal Daumé III and Daniel Marcu. 2005b. Bayesian summarization at DUC and a suggestion for extrinsic evaluation. In *Document Understanding Conference*.

John Lafferty and ChengXiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the Conference on Research and Developments in Information Retrieval (SIGIR)*.

Victor Lavrenko, M. Choquette, and Bruce Croft. 2002. Crosslingual relevance models. In *Proceedings of the Conference on Research and Developments in Information Retrieval (SIGIR)*.

Xiaoyong Liu and Bruce Croft. 2002. Passage retrieval based on language models. In *Processing of the Conference on Information and Knowledge Management (CIKM)*.

Thomas Minka and John Lafferty. 2003. Expectation-propagation for the generative aspect model. In *Proceedings of the Converence on Uncertainty in Artificial Intelligence (UAI)*.

Thomas Minka. 2001. *A family of algorithms for approximate Bayesian inference*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Vanessa Murdock and Bruce Croft. 2005. A translation model for sentence retrieval. In *Proceedings of the Joint Conference on Human Language Technology Conference and Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 684–691.

Jahna Otterbacher, Gunes Erkan, and Dragomir R. Radev. 2005. Using random walks for question-focused sentence retrieval. In *Proceedings of the Joint Conference on Human Language Technology Conference and Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

Jay M. Ponte and Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the Conference on Research and Developments in Information Retrieval (SIGIR)*.

Hanna Wallach. 2006. Topic modeling: beyond bag-of-words. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Yi Zhang, Jamie Callan, and Thomas Minka. 2002. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the Conference on Research and Developments in Information Retrieval (SIGIR)*.

Expressing Implicit Semantic Relations without Supervision

Peter D. Turney
Institute for Information Technology
National Research Council Canada
M-50 Montreal Road
Ottawa, Ontario, Canada, K1A 0R6
peter.turney@nrc-cnrc.gc.ca

Abstract

We present an unsupervised learning algorithm that mines large text corpora for patterns that express implicit semantic relations. For a given input word pair $X:Y$ with some unspecified semantic relations, the corresponding output list of patterns $\langle P_1,\ldots,P_m\rangle$ is ranked according to how well each pattern P_i expresses the relations between X and Y. For example, given $X = \text{ostrich}$ and $Y = \text{bird}$, the two highest ranking output patterns are "*X* is the largest *Y*" and "*Y* such as the *X*". The output patterns are intended to be useful for finding further pairs with the same relations, to support the construction of lexicons, ontologies, and semantic networks. The patterns are sorted by *pertinence*, where the pertinence of a pattern P_i for a word pair $X:Y$ is the expected relational similarity between the given pair and typical pairs for P_i. The algorithm is empirically evaluated on two tasks, solving multiple-choice SAT word analogy questions and classifying semantic relations in noun-modifier pairs. On both tasks, the algorithm achieves state-of-the-art results, performing significantly better than several alternative pattern ranking algorithms, based on tf-idf.

1 Introduction

In a widely cited paper, Hearst (1992) showed that the lexico-syntactic pattern "*Y* such as the *X*" can be used to mine large text corpora for word pairs $X:Y$ in which *X* is a hyponym (type) of *Y*. For example, if we search in a large corpus using the pattern "*Y* such as the *X*" and we find the string "bird such as the ostrich", then we can infer that "ostrich" is a hyponym of "bird". Berland and Charniak (1999) demonstrated that the patterns "*Y*'s *X*" and "*X* of the *Y*" can be used to mine corpora for pairs $X:Y$ in which *X* is a meronym (part) of *Y* (e.g., "wheel of the car").

Here we consider the *inverse* of this problem: Given a word pair $X:Y$ with some unspecified semantic relations, can we mine a large text corpus for lexico-syntactic patterns that express the implicit relations between X and Y? For example, if we are given the pair ostrich:bird, can we discover the pattern "*Y* such as the *X*"? We are particularly interested in discovering high quality patterns that are reliable for mining further word pairs with the same semantic relations.

In our experiments, we use a corpus of web pages containing about 5×10^{10} English words (Terra and Clarke, 2003). From co-occurrences of the pair ostrich:bird in this corpus, we can generate 516 patterns of the form "*X* ... *Y*" and 452 patterns of the form "*Y* ... *X*". Most of these patterns are not very useful for text mining. The main challenge is to find a way of ranking the patterns, so that patterns like "*Y* such as the *X*" are highly ranked. Another challenge is to find a way to empirically evaluate the performance of any such pattern ranking algorithm.

For a given input word pair $X:Y$ with some unspecified semantic relations, we rank the corresponding output list of patterns $\langle P_1,\ldots,P_m\rangle$ in order of decreasing *pertinence*. The pertinence of a pattern P_i for a word pair $X:Y$ is the expected relational similarity between the given pair and typical pairs that fit P_i. We define pertinence more precisely in Section 2.

Hearst (1992) suggests that her work may be useful for building a thesaurus. Berland and Charniak (1999) suggest their work may be useful for building a lexicon or ontology, like WordNet. Our algorithm is also applicable to these tasks. Other potential applications and related problems are discussed in Section 3.

To calculate pertinence, we must be able to measure relational similarity. Our measure is based on Latent Relational Analysis (Turney, 2005). The details are given in Section 4.

Given a word pair $X:Y$, we want our algorithm to rank the corresponding list of patterns

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 313–320,
Sydney, July 2006. ©2006 Association for Computational Linguistics

$\langle P_1,\ldots,P_m\rangle$ according to their value for mining text, in support of semantic network construction and similar tasks. Unfortunately, it is difficult to measure performance on such tasks. Therefore our experiments are based on two tasks that provide objective performance measures.

In Section 5, ranking algorithms are compared by their performance on solving multiple-choice SAT word analogy questions. In Section 6, they are compared by their performance on classifying semantic relations in noun-modifier pairs. The experiments demonstrate that ranking by pertinence is significantly better than several alternative pattern ranking algorithms, based on tf-idf. The performance of pertinence on these two tasks is slightly below the best performance that has been reported so far (Turney, 2005), but the difference is not statistically significant.

We discuss the results in Section 7 and conclude in Section 8.

2 Pertinence

The *relational similarity* between two pairs of words, $X_1:Y_1$ and $X_2:Y_2$, is the degree to which their semantic relations are analogous. For example, mason:stone and carpenter:wood have a high degree of relational similarity. Measuring relational similarity will be discussed in Section 4. For now, assume that we have a measure of the relational similarity between pairs of words, $\mathrm{sim_r}(X_1:Y_1, X_2:Y_2) \in \Re$.

Let $W = \{X_1:Y_1,\ldots,X_n:Y_n\}$ be a set of word pairs and let $P = \{P_1,\ldots,P_m\}$ be a set of patterns. The *pertinence* of pattern P_i to a word pair $X_j:Y_j$ is the *expected relational similarity* between a word pair $X_k:Y_k$, randomly selected from W according to the probability distribution $\mathrm{p}(X_k:Y_k|P_i)$, and the word pair $X_j:Y_j$:

$$\mathrm{pertinence}(X_j:Y_j,P_i) = \sum_{k=1}^{n} \mathrm{p}(X_k:Y_k|P_i)\cdot \mathrm{sim_r}(X_j:Y_j, X_k:Y_k)$$

The conditional probability $\mathrm{p}(X_k:Y_k|P_i)$ can be interpreted as the degree to which the pair $X_k:Y_k$ is representative (i.e., *typical*) of pairs that fit the pattern P_i. That is, P_i is pertinent to $X_j:Y_j$ if highly typical word pairs $X_k:Y_k$ for the pattern P_i tend to be relationally similar to $X_j:Y_j$.

Pertinence tends to be highest with patterns that are unambiguous. The maximum value of $\mathrm{pertinence}(X_j:Y_j,P_i)$ is attained when the pair $X_j:Y_j$ belongs to a cluster of highly similar pairs and the conditional probability distribution $\mathrm{p}(X_k:Y_k|P_i)$ is concentrated on the cluster. An ambiguous pattern, with its probability spread over multiple clusters, will have less pertinence.

If a pattern with high pertinence is used for text mining, it will tend to produce word pairs that are very similar to the given word pair; this follows from the definition of pertinence. We believe this definition is the first formal measure of quality for text mining patterns.

Let $f_{k,i}$ be the number of occurrences in a corpus of the word pair $X_k:Y_k$ with the pattern P_i. We could estimate $\mathrm{p}(X_k:Y_k|P_i)$ as follows:

$$\mathrm{p}(X_k:Y_k|P_i) = f_{k,i} \Big/ \sum_{j=1}^{n} f_{j,i}$$

Instead, we first estimate $\mathrm{p}(P_i|X_k:Y_k)$:

$$\mathrm{p}(P_i|X_k:Y_k) = f_{k,i} \Big/ \sum_{j=1}^{m} f_{k,j}$$

Then we apply Bayes' Theorem:

$$\mathrm{p}(X_k:Y_k|P_i) = \frac{\mathrm{p}(X_k:Y_k)\cdot \mathrm{p}(P_i|X_k:Y_k)}{\sum_{j=1}^{n}\mathrm{p}(X_j:Y_j)\cdot \mathrm{p}(P_i|X_j:Y_j)}$$

We assume $\mathrm{p}(X_j:Y_j) = 1/n$ for all pairs in W:

$$\mathrm{p}(X_k:Y_k|P_i) = \mathrm{p}(P_i|X_k:Y_k) \Big/ \sum_{j=1}^{n}\mathrm{p}(P_i|X_j:Y_j)$$

The use of Bayes' Theorem and the assumption that $\mathrm{p}(X_j:Y_j) = 1/n$ for all word pairs is a way of smoothing the probability $\mathrm{p}(X_k:Y_k|P_i)$, similar to Laplace smoothing.

3 Related Work

Hearst (1992) describes a method for finding patterns like "*Y* such as the *X*", but her method requires human judgement. Berland and Charniak (1999) use Hearst's manual procedure.

Riloff and Jones (1999) use a mutual bootstrapping technique that can find patterns automatically, but the bootstrapping requires an initial seed of manually chosen examples for each class of words. Miller et al. (2000) propose an approach to relation extraction that was evaluated in the Seventh Message Understanding Conference (MUC7). Their algorithm requires labeled examples of each relation. Similarly, Zelenko et al. (2003) use a supervised kernel method that requires labeled training examples. Agichtein and Gravano (2000) also require training examples for each relation. Brin (1998) uses bootstrapping from seed examples of author:title pairs to discover patterns for mining further pairs.

Yangarber et al. (2000) and Yangarber (2003) present an algorithm that can find patterns automatically, but it requires an initial seed of manually designed patterns for each semantic relation. Stevenson (2004) uses WordNet to extract relations from text, but also requires initial seed patterns for each relation.

Lapata (2002) examines the task of expressing the implicit relations in nominalizations, which are noun compounds whose head noun is derived from a verb and whose modifier can be interpreted as an argument of the verb. In contrast with this work, our algorithm is not restricted to nominalizations. Section 6 shows that our algorithm works with arbitrary noun compounds and the SAT questions in Section 5 include all nine possible pairings of nouns, verbs, and adjectives.

As far as we know, our algorithm is the first unsupervised learning algorithm that can find patterns for semantic relations, given only a large corpus (e.g., in our experiments, about 5×10^{10} words) and a moderately sized set of word pairs (e.g., 600 or more pairs in the experiments), such that the members of each pair appear together frequently in short phrases in the corpus. These word pairs are not seeds, since the algorithm does not require the pairs to be labeled or grouped; we do not assume they are homogenous.

The word pairs that we need could be generated automatically, by searching for word pairs that co-occur frequently in the corpus. However, our evaluation methods (Sections 5 and 6) both involve a predetermined list of word pairs. If our algorithm were allowed to generate its own word pairs, the overlap with the predetermined lists would likely be small. This is a limitation of our evaluation methods rather than the algorithm.

Since any two word pairs may have some relations in common and some that are not shared, our algorithm generates a unique list of patterns for each input word pair. For example, mason:stone and carpenter:wood share the pattern "*X* carves *Y*", but the patterns "*X* nails *Y*" and "*X* bends *Y*" are unique to carpenter:wood. The ranked list of patterns for a word pair $X:Y$ gives the relations between *X* and *Y* in the corpus, sorted with the most pertinent (i.e., characteristic, distinctive, unambiguous) relations first.

Turney (2005) gives an algorithm for measuring the relational similarity between two pairs of words, called Latent Relational Analysis (LRA). This algorithm can be used to solve multiple-choice word analogy questions and to classify noun-modifier pairs (Turney, 2005), but it does not attempt to express the implicit semantic relations. Turney (2005) maps each pair $X:Y$ to a high-dimensional vector $\vec{v}$. The value of each element v_i in $\vec{v}$ is based on the frequency, for the pair $X:Y$, of a corresponding pattern P_i. The relational similarity between two pairs, $X_1:Y_1$ and $X_2:Y_2$, is derived from the cosine of the angle between their two vectors. A limitation of this approach is that the semantic content of the vectors is difficult to interpret; the magnitude of an element v_i is not a good indicator of how well the corresponding pattern P_i expresses a relation of $X:Y$. This claim is supported by the experiments in Sections 5 and 6.

Pertinence (as defined in Section 2) builds on the measure of relational similarity in Turney (2005), but it has the advantage that the semantic content can be interpreted; we can point to specific patterns and say that they express the implicit relations. Furthermore, we can use the patterns to find other pairs with the same relations.

Hearst (1992) processed her text with a part-of-speech tagger and a unification-based constituent analyzer. This makes it possible to use more general patterns. For example, instead of the literal string pattern "*Y* such as the *X*", where *X* and *Y* are words, Hearst (1992) used the more abstract pattern "NP_0 such as NP_1", where NP_i represents a noun phrase. For the sake of simplicity, we have avoided part-of-speech tagging, which limits us to literal patterns. We plan to experiment with tagging in future work.

4 The Algorithm

The algorithm takes as input a set of word pairs $W = \{X_1:Y_1, \ldots, X_n:Y_n\}$ and produces as output ranked lists of patterns $\langle P_1, \ldots, P_m \rangle$ for each input pair. The following steps are similar to the algorithm of Turney (2005), with several changes to support the calculation of pertinence.

1. Find phrases: For each pair $X_i:Y_i$, make a list of phrases in the corpus that contain the pair. We use the Waterloo MultiText System (Clarke et al., 1998) to search in a corpus of about 5×10^{10} English words (Terra and Clarke, 2003). Make one list of phrases that begin with X_i and end with Y_i and a second list for the opposite order. Each phrase must have one to three intervening words between X_i and Y_i. The first and last words in the phrase do not need to exactly match X_i and Y_i. The MultiText query language allows different suffixes. Veale (2004) has observed that it is easier to identify semantic relations between nouns than between other parts of speech. Therefore we use WordNet 2.0 (Miller, 1995) to guess whether X_i and Y_i are likely to be nouns. When they are nouns, we are relatively strict about suffixes; we only allow variation in pluralization. For all other parts of speech, we are liberal about suffixes. For example, we allow an adjective such as "inflated" to match a noun such as "inflation". With MultiText, the query "inflat*" matches both "inflated" and "inflation".

2. Generate patterns: For each list of phrases, generate a list of patterns, based on the phrases. Replace the first word in each phrase with the generic marker "*X*" and replace the last word with "*Y*". The intervening words in each phrase may be either left as they are or replaced with the wildcard "*". For example, the phrase "carpenter nails the wood" yields the patterns "*X* nails the *Y*", "*X* nails * *Y*", "*X* * the *Y*", and "*X* * * *Y*". Do not allow duplicate patterns in a list, but note the number of times a pattern is generated for each word pair $X_i : Y_i$ in each order (X_i first and Y_i last or vice versa). We call this the *pattern frequency*. It is a local frequency count, analogous to *term frequency* in information retrieval.

3. Count pair frequency: The *pair frequency* for a pattern is the number of lists from the preceding step that contain the given pattern. It is a global frequency count, analogous to *document frequency* in information retrieval. Note that a pair $X_i : Y_i$ yields two lists of phrases and hence two lists of patterns. A given pattern might appear in zero, one, or two of the lists for $X_i : Y_i$.

4. Map pairs to rows: In preparation for building a matrix $\mathbf{X}$, create a mapping of word pairs to row numbers. For each pair $X_i : Y_i$, create a row for $X_i : Y_i$ and another row for $Y_i : X_i$. If *W* does not already contain $\{Y_1 : X_1, \ldots, Y_n : X_n\}$, then we have effectively doubled the number of word pairs, which increases the sample size for calculating pertinence.

5. Map patterns to columns: Create a mapping of patterns to column numbers. For each unique pattern of the form "*X* ... *Y*" from Step 2, create a column for the original pattern "*X* ... *Y*" and another column for the same pattern with *X* and *Y* swapped, "*Y* ... *X*". Step 2 can generate millions of distinct patterns. The experiment in Section 5 results in 1,706,845 distinct patterns, yielding 3,413,690 columns. This is too many columns for matrix operations with today's standard desktop computer. Most of the patterns have a very low pair frequency. For the experiment in Section 5, 1,371,702 of the patterns have a pair frequency of one. To keep the matrix $\mathbf{X}$ manageable, we drop all patterns with a pair frequency less than ten. For Section 5, this leaves 42,032 patterns, yielding 84,064 columns. Turney (2005) limited the matrix to 8,000 columns, but a larger pool of patterns is better for our purposes, since it increases the likelihood of finding good patterns for expressing the semantic relations of a given word pair.

6. Build a sparse matrix: Build a matrix $\mathbf{X}$ in sparse matrix format. The value for the cell in row *i* and column *j* is the pattern frequency of the *j*-th pattern for the the *i*-th word pair.

7. Calculate entropy: Apply log and entropy transformations to the sparse matrix $\mathbf{X}$ (Landauer and Dumais, 1997). Each cell is replaced with its logarithm, multiplied by a weight based on the negative entropy of the corresponding column vector in the matrix. This gives more weight to patterns that vary substantially in frequency for each pair.

8. Apply SVD: After log and entropy transforms, apply the Singular Value Decomposition (SVD) to $\mathbf{X}$ (Golub and Van Loan, 1996). SVD decomposes $\mathbf{X}$ into a product of three matrices $\mathbf{U\Sigma V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are in column orthonormal form (i.e., the columns are orthogonal and have unit length) and Σ is a diagonal matrix of singular values (hence SVD). If $\mathbf{X}$ is of rank r, then Σ is also of rank r. Let Σ_k, where $k < r$, be the diagonal matrix formed from the top k singular values, and let $\mathbf{U}_k$ and $\mathbf{V}_k$ be the matrices produced by selecting the corresponding columns from $\mathbf{U}$ and $\mathbf{V}$. The matrix $\mathbf{U}_k \Sigma_k \mathbf{V}_k^T$ is the matrix of rank k that best approximates the original matrix $\mathbf{X}$, in the sense that it minimizes the approximation errors (Golub and Van Loan, 1996). Following Landauer and Dumais (1997), we use $k = 300$. We may think of this matrix $\mathbf{U}_k \Sigma_k \mathbf{V}_k^T$ as a smoothed version of the original matrix. SVD is used to reduce noise and compensate for sparseness (Landauer and Dumais, 1997).

9. Calculate cosines: The relational similarity between two pairs, $\text{sim}_\text{r}(X_1 : Y_1, X_2 : Y_2)$, is given by the cosine of the angle between their corresponding row vectors in the matrix $\mathbf{U}_k \Sigma_k \mathbf{V}_k^T$ (Turney, 2005). To calculate pertinence, we will need the relational similarity between all possible pairs of pairs. All of the cosines can be efficiently derived from the matrix $\mathbf{U}_k \Sigma_k (\mathbf{U}_k \Sigma_k)^T$ (Landauer and Dumais, 1997).

10. Calculate conditional probabilities: Using Bayes' Theorem (see Section 2) and the raw frequency data in the matrix $\mathbf{X}$ from Step 6, before log and entropy transforms, calculate the conditional probability $\text{p}(X_i : Y_i | P_j)$ for every row (word pair) and every column (pattern).

11. Calculate pertinence: With the cosines from Step 9 and the conditional probabilities from Step 10, calculate $\text{pertinence}(X_i : Y_i, P_j)$ for every row $X_i : Y_i$ and every column P_j for which $\text{p}(X_i : Y_i | P_j) > 0$. When $\text{p}(X_i : Y_i | P_j) = 0$, it is possible that $\text{pertinence}(X_i : Y_i, P_j) > 0$, but we avoid calculating pertinence in these cases for two reasons. First, it speeds computation, because $\mathbf{X}$ is sparse, so $\text{p}(X_i : Y_i | P_j) = 0$ for most rows and columns. Second, $\text{p}(X_i : Y_i | P_j) = 0$ implies that the pattern P_j does not actually appear with the word pair $X_i : Y_i$ in the corpus; we are only guessing that the pattern is appropriate for the word pair, and we could be wrong. Therefore we prefer to limit ourselves to patterns and word pairs that have actually been observed in the corpus. For each pair $X_i : Y_i$ in *W*, output two separate ranked lists, one for patterns of the form "*X* … *Y*" and another for patterns of the form

"Y … X", where the patterns in both lists are sorted in order of decreasing pertinence to $X_i : Y_i$. Ranking serves as a kind of normalization. We have found that the relative rank of a pattern is more reliable as an indicator of its importance than the absolute pertinence. This is analogous to information retrieval, where documents are ranked in order of their relevance to a query. The relative rank of a document is more important than its actual numerical score (which is usually hidden from the user of a search engine). Having two separate ranked lists helps to avoid bias. For example, ostrich:bird generates 516 patterns of the form *"X ... Y"* and 452 patterns of the form *"Y ... X"*. Since there are more patterns of the form *"X ... Y"*, there is a slight bias towards these patterns. If the two lists were merged, the *"Y ... X"* patterns would be at a disadvantage.

5 Experiments with Word Analogies

In these experiments, we evaluate pertinence using 374 college-level multiple-choice word analogies, taken from the SAT test. For each question, there is a target word pair, called the *stem* pair, and five *choice* pairs. The task is to find the choice that is most analogous (i.e., has the highest relational similarity) to the stem. This choice pair is called the *solution* and the other choices are *distractors*. Since there are six word pairs per question (the stem and the five choices), there are $374 \times 6 = 2244$ pairs in the input set *W*. In Step 4 of the algorithm, we double the pairs, but we also drop some pairs because they do not co-occur in the corpus. This leaves us with 4194 rows in the matrix. As mentioned in Step 5, the matrix has 84,064 columns (patterns). The sparse matrix density is 0.91%.

To answer a SAT question, we generate ranked lists of patterns for each of the six word pairs. Each choice is evaluated by taking the intersection of its patterns with the stem's patterns. The shared patterns are scored by the average of their rank in the stem's lists and the choice's lists. Since the lists are sorted in order of decreasing pertinence, a low score means a high pertinence. Our guess is the choice with the lowest scoring shared pattern.

Table 1 shows three examples, two questions that are answered correctly followed by one that is answered incorrectly. The correct answers are in bold font. For the first question, the stem is ostrich:bird and the best choice is (a) lion:cat. The highest ranking pattern that is shared by both of these pairs is *"Y* such as the *X"*. The third question illustrates that, even when the answer is incorrect, the best shared pattern (*"Y* powered * * *X"*) may be plausible.

Word pair	Best shared pattern	Score
1. ostrich:bird		
(a) lion:cat	***"Y* such as the *X"***	**1.0**
(b) goose:flock	*"X* * * breeding *Y"*	43.5
(c) ewe:sheep	*"X* are the only *Y"*	13.5
(d) cub:bear	*"Y* are called *X"*	29.0
(e) primate:monkey	*"Y* is the * *X"*	80.0
2. traffic:street		
(a) ship:gangplank	*"X* * down the *Y"*	53.0
(b) crop:harvest	*"X* * adjacent * *Y"*	248.0
(c) car:garage	*"X* * a residential *Y"*	63.0
(d) pedestrians:feet	*"Y* * accommodate *X"*	23.0
(e) water:riverbed	***"Y* that carry *X"***	**17.0**
3. locomotive:train		
(a) horse:saddle	*"X* carrying * *Y"*	82.0
(b) tractor:plow	***"X* pulled * *Y"***	**7.0**
(c) rudder:rowboat	*"Y* * *X"*	319.0
(d) camel:desert	*"Y* with two *X"*	43.0
(e) gasoline:automobile	*"Y* powered * * *X"*	5.0

Table 1. Three examples of SAT questions.

Table 2 shows the four highest ranking patterns for the stem and solution for the first example. The pattern *"X* lion *Y"* is anomalous, but the other patterns seem reasonable. The shared pattern *"Y* such as the *X"* is ranked 1 for both pairs, hence the average score for this pattern is 1.0, as shown in Table 1. Note that the "ostrich is the largest bird" and "lions are large cats", but the largest cat is the Siberian tiger.

Word pair	*"X ... Y"*	*"Y ... X"*
ostrich:bird	*"X* is the largest *Y"*	***"Y* such as the *X"***
	"X is * largest *Y"*	*"Y* such * the *X"*
lion:cat	*"X* lion *Y"*	***"Y* such as the *X"***
	"X are large *Y"*	*"Y* and mountain *X"*

Table 2. The highest ranking patterns.

Table 3 lists the top five pairs in *W* that match the pattern *"Y* such as the *X"*. The pairs are sorted by $\mathrm{p}(X : Y \mid P)$. The pattern *"Y* such as the *X"* is one of 146 patterns that are shared by ostrich:bird and lion:cat. Most of these shared patterns are not very informative.

Word pair	Conditional probability
heart:organ	0.49342
dodo:bird	0.08888
elbow:joint	0.06385
ostrich:bird	0.05774
semaphore:signal	0.03741

Table 3. The top five pairs for *"Y* such as the *X"*.

In Table 4, we compare ranking patterns by pertinence to ranking by various other measures, mostly based on varieties of tf-idf (term frequency times inverse document frequency, a common way to rank documents in information retrieval). The tf-idf measures are taken from Salton and Buckley (1988). For comparison, we also include three algorithms that do not rank patterns (the bottom three rows in the table). These three algorithms can answer the SAT questions, but they do not provide any kind of explanation for their answers.

	Algorithm	Prec.	Rec.	F
1	pertinence (Step 11)	55.7	53.5	54.6
2	log and entropy matrix (Step 7)	43.5	41.7	42.6
3	TF = f, IDF = $\log((N\text{-}n)/n)$	43.2	41.4	42.3
4	TF = $\log(f+1)$, IDF = $\log(N/n)$	42.9	41.2	42.0
5	TF = f, IDF = $\log(N/n)$	42.9	41.2	42.0
6	TF = $\log(f+1)$, IDF = $\log((N\text{-}n)/n)$	42.3	40.6	41.4
7	TF = 1.0, IDF = $1/n$	41.5	39.8	40.6
8	TF = f, IDF = $1/n$	41.5	39.8	40.6
9	TF = 0.5 + 0.5 * (f/F), IDF = $\log(N/n)$	41.5	39.8	40.6
10	TF = $\log(f+1)$, IDF = $1/n$	41.2	39.6	40.4
11	$p(X{:}Y\|P)$ (Step 10)	39.8	38.2	39.0
12	SVD matrix (Step 8)	35.9	34.5	35.2
13	random	27.0	25.9	26.4
14	TF = $1/f$, IDF = 1.0	26.7	25.7	26.2
15	TF = f, IDF = 1.0 (Step 6)	18.1	17.4	17.7
16	Turney (2005)	56.8	56.1	56.4
17	Turney and Littman (2005)	47.7	47.1	47.4
18	Veale (2004)	42.8	42.8	42.8

Table 4. Performance of various algorithms on SAT.

All of the pattern ranking algorithms are given exactly the same sets of patterns to rank. Any differences in performance are due to the ranking method alone. The algorithms may skip questions when the word pairs do not co-occur in the corpus. All of the ranking algorithms skip the same set of 15 of the 374 SAT questions. *Precision* is defined as the percentage of correct answers out of the questions that were answered (not skipped). *Recall* is the percentage of correct answers out of the maximum possible number correct (374). The F measure is the harmonic mean of precision and recall.

For the tf-idf methods in Table 4, f is the pattern frequency, n is the pair frequency, F is the maximum f for all patterns for the given word pair, and N is the total number of word pairs. By “TF = f, IDF = $1/n$”, for example (row 8), we mean that f plays a role that is analogous to term frequency and $1/n$ plays a role that is analogous to inverse document frequency. That is, in row 8, the patterns are ranked in decreasing order of pattern frequency divided by pair frequency.

Table 4 also shows some ranking methods based on intermediate calculations in the algorithm in Section 4. For example, row 2 in Table 4 gives the results when patterns are ranked in order of decreasing values in the corresponding cells of the matrix $\mathbf{X}$ from Step 7.

Row 12 in Table 4 shows the results we would get using Latent Relational Analysis (Turney, 2005) to rank patterns. The results in row 12 support the claim made in Section 3, that LRA is not suitable for ranking patterns, although it works well for answering the SAT questions (as we see in row 16). The vectors in LRA yield a good measure of relational similarity, but the magnitude of the value of a specific element in a vector is not a good indicator of the quality of the corresponding pattern.

The best method for ranking patterns is pertinence (row 1 in Table 4). As a point of comparison, the performance of the average senior highschool student on the SAT analogies is about 57% (Turney and Littman, 2005). The second best method is to use the values in the matrix $\mathbf{X}$ after the log and entropy transformations in Step 7 (row 2). The difference between these two methods is statistically significant with 95% confidence. Pertinence (row 1) performs slightly below Latent Relational Analysis (row 16; Turney, 2005), but the difference is not significant.

Randomly guessing answers should yield an F of 20% (1 out of 5 choices), but ranking patterns randomly (row 13) results in an F of 26.4%. This is because the stem pair tends to share more patterns with the solution pair than with the distractors. The minimum of a large set of random numbers is likely to be lower than the minimum of a small set of random numbers.

6 Experiments with Noun-Modifiers

In these experiments, we evaluate pertinence on the task of classifying noun-modifier pairs. The problem is to classify a noun-modifier pair, such as “flu virus”, according to the semantic relation between the head noun (virus) and the modifier (flu). For example, “flu virus” is classified as a *causality* relation (the flu is *caused by* a virus). For these experiments, we use a set of 600 manually labeled noun-modifier pairs (Nastase and Szpakowicz, 2003). There are five general classes of labels with thirty subclasses. We present here the results with five classes; the results with thirty subclasses follow the same trends (that is, pertinence performs significantly better than the other ranking methods). The five classes are *causality* (storm cloud), *temporality* (daily exercise), *spatial* (desert storm), *participant* (student protest), and *quality* (expensive book).

The input set W consists of the 600 noun-modifier pairs. This set is doubled in Step 4, but we drop some pairs because they do not co-occur in the corpus, leaving us with 1184 rows in the matrix. There are 16,849 distinct patterns with a pair frequency of ten or more, resulting in 33,698 columns. The matrix density is 2.57%.

To classify a noun-modifier pair, we use a single nearest neighbour algorithm with leave-one-out cross-validation. We split the set 600 times. Each pair gets a turn as the single testing example, while the other 599 pairs serve as training examples. The testing example is classified according to the label of its nearest neighbour in the training set. The distance between two noun-modifier pairs is measured by the average rank of their best shared pattern. Table 5 shows the resulting precision, recall, and F, when ranking patterns by pertinence.

Class name	Prec.	Rec.	F	Class size
causality	37.3	36.0	36.7	86
participant	61.1	64.4	62.7	260
quality	49.3	50.7	50.0	146
spatial	43.9	32.7	37.5	56
temporality	64.7	63.5	64.1	52
all	51.3	49.5	50.2	600

Table 5. Performance on noun-modifiers.

To gain some insight into the algorithm, we examined the 600 best shared patterns for each pair and its single nearest neighbour. For each of the five classes, Table 6 lists the most frequent pattern among the best shared patterns for the given class. All of these patterns seem appropriate for their respective classes.

Class	Most frequent pattern	Example pair
causality	"*Y* * causes *X*"	"cold virus"
participant	"*Y* of his *X*"	"dream analysis"
quality	"*Y* made of *X*"	"copper coin"
spatial	"*X* * * terrestrial *Y*"	"aquatic mammal"
temporality	"*Y* in * early *X*"	"morning frost"

Table 6. Most frequent of the best shared patterns.

Table 7 gives the performance of pertinence on the noun-modifier problem, compared to various other pattern ranking methods. The bottom two rows are included for comparison; they are not pattern ranking algorithms. The best method for ranking patterns is pertinence (row 1 in Table 7). The difference between pertinence and the second best ranking method (row 2) is statistically significant with 95% confidence. Latent Relational Analysis (row 16) performs slightly better than pertinence (row 1), but the difference is not statistically significant.

Row 6 in Table 7 shows the results we would get using Latent Relational Analysis (Turney, 2005) to rank patterns. Again, the results support the claim in Section 3, that LRA is not suitable for ranking patterns. LRA can classify the noun-modifiers (as we see in row 16), but it cannot express the implicit semantic relations that make an unlabeled noun-modifier in the testing set similar to its nearest neighbour in the training set.

	Algorithm	Prec.	Rec.	F
1	pertinence (Step 11)	51.3	49.5	50.2
2	TF = log(*f*+1), IDF = 1/*n*	37.4	36.5	36.9
3	TF = log(*f*+1), IDF = log(*N*/*n*)	36.5	36.0	36.2

4	TF = log(*f*+1),
	IDF = log((*N*-*n*)/*n*)
		36.0	35.4	35.7

5	TF = *f*, IDF = log((*N*-*n*)/*n*)	36.0	35.3	35.6
6	SVD matrix (Step 8)	43.9	33.4	34.8
7	TF = *f*, IDF = 1/*n*	35.4	33.6	34.3
8	log and entropy matrix (Step 7)	35.6	33.3	34.1
9	TF = *f*, IDF = log(*N*/*n*)	34.1	31.4	32.2

10	TF = 0.5 + 0.5 * (*f*/*F*),
	IDF = log(*N*/*n*)
		31.9	31.7	31.6

11	p(*X*:*Y*|*P*) (Step 10)	31.8	30.8	31.2
12	TF = 1.0, IDF = 1/*n*	29.2	28.8	28.7
13	random	19.4	19.3	19.2
14	TF = 1/*f*, IDF = 1.0	20.3	20.7	19.2
15	TF = *f*, IDF = 1.0 (Step 6)	12.8	19.7	8.0
16	Turney (2005)	55.9	53.6	54.6
17	Turney and Littman (2005)	43.4	43.1	43.2

Table 7. Performance on noun-modifiers.

7 Discussion

Computing pertinence took about 18 hours for the experiments in Section 5 and 9 hours for Section 6. In both cases, the majority of the time was spent in Step 1, using MultiText (Clarke et al., 1998) to search through the corpus of 5×10^{10} words. MultiText was running on a Beowulf cluster with sixteen 2.4 GHz Intel Xeon CPUs. The corpus and the search index require about one terabyte of disk space. This may seem computationally demanding by today's standards, but progress in hardware will soon allow an average desktop computer to handle corpora of this size.

Although the performance on the SAT analogy questions (54.6%) is near the level of the average senior highschool student (57%), there is room for improvement. For applications such as building a thesaurus, lexicon, or ontology, this level of performance suggests that our algorithm could assist, but not replace, a human expert.

One possible improvement would be to add part-of-speech tagging or parsing. We have done some preliminary experiments with parsing and plan to explore tagging as well. A difficulty is that much of the text in our corpus does not consist of properly formed sentences, since the text comes from web pages. This poses problems for most part-of-speech taggers and parsers.

8 Conclusion

Latent Relational Analysis (Turney, 2005) provides a way to measure the relational similarity between two word pairs, but it gives us little insight into how the two pairs are similar. In effect, LRA is a black box. The main contribution of this paper is the idea of pertinence, which allows us to take an opaque measure of relational similarity and use it to find patterns that express the implicit semantic relations between two words.

The experiments in Sections 5 and 6 show that ranking patterns by pertinence is superior to ranking them by a variety of tf-idf methods. On the word analogy and noun-modifier tasks, pertinence performs as well as the state-of-the-art, LRA, but pertinence goes beyond LRA by making relations explicit.

Acknowledgements

Thanks to Joel Martin, David Nadeau, and Deniz Yuret for helpful comments and suggestions.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries (ACM DL 2000)*, pages 85-94.

Matthew Berland and Eugene Charniak. 1999. Finding parts in very large corpora. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, pages 57-64.

Sergey Brin. 1998. Extracting patterns and relations from the World Wide Web. In *WebDB Workshop at the 6th International Conference on Extending Database Technology (EDBT-98)*, pages 172-183.

Charles L.A. Clarke, Gordon V. Cormack, and Christopher R. Palmer. 1998. An overview of MultiText. *ACM SIGIR Forum*, 32(2):14-15.

Gene H. Golub and Charles F. Van Loan. 1996. *Matrix Computations*. Third edition. Johns Hopkins University Press, Baltimore, MD.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 539-545.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211-240.

Maria Lapata. 2002. The disambiguation of nominalisations. *Computational Linguistics*, 28(3):357-388.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39-41.

Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP 2000)*, pages 226-233.

Vivi Nastase and Stan Szpakowicz. 2003. Exploring noun-modifier semantic relations. In *Fifth International Workshop on Computational Semantics (IWCS-5)*, pages 285-301.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99)*, pages 474-479.

Gerard Salton and Chris Buckley. 1988. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513-523.

Mark Stevenson. 2004. An unsupervised WordNet-based algorithm for relation extraction. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC) Workshop, Beyond Named Entity Recognition: Semantic Labelling for NLP Tasks*, Lisbon, Portugal.

Egidio Terra and Charles L.A. Clarke. 2003. Frequency estimates for statistical word similarity measures. In *Proceedings of the Human Language Technology and North American Chapter of Association of Computational Linguistics Conference (HLT/NAACL-03)*, pages 244-251.

Peter D. Turney. 2005. Measuring semantic similarity by latent relational analysis. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1136-1141.

Peter D. Turney and Michael L. Littman. 2005. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60(1-3):251-278.

Tony Veale. 2004. WordNet sits the SAT: A knowledge-based approach to lexical analogy. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 606-612.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Unsupervised discovery of scenario-level patterns for information extraction. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP 2000)*, pages 282-289.

Roman Yangarber. 2003. Counter-training in discovery of semantic patterns. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 343-350.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083-1106.

Hybrid Parsing: Using Probabilistic Models as Predictors for a Symbolic Parser

Kilian A. Foth, Wolfgang Menzel
Department of Informatics
Universität Hamburg, Germany
{foth|menzel}@informatik.uni-hamburg.de

Abstract

In this paper we investigate the benefit of stochastic predictor components for the parsing quality which can be obtained with a rule-based dependency grammar. By including a chunker, a supertagger, a PP attacher, and a fast probabilistic parser we were able to improve upon the baseline by 3.2%, bringing the overall labelled accuracy to 91.1% on the German NEGRA corpus. We attribute the successful integration to the ability of the underlying grammar model to combine uncertain evidence in a soft manner, thus avoiding the problem of error propagation.

1 Introduction

There seems to be an upper limit for the level of quality that can be achieved by a parser if it is confined to information drawn from a single source. Stochastic parsers for English trained on the Penn Treebank have peaked their performance around 90% (Charniak, 2000). Parsing of German seems to be even harder and parsers trained on the NEGRA corpus or an enriched version of it still perform considerably worse. On the other hand, a great number of shallow components like taggers, chunkers, supertaggers, as well as general or specialized attachment predictors have been developed that might provide additional information to further improve the quality of a parser's output, as long as their contributions are in some sense complementory. Despite these prospects, such possibilities have rarely been investigated so far.

To estimate the degree to which the desired synergy between heterogeneous knowledge sources can be achieved, we have established an experimental framework for syntactic analysis which allows us to plug in a wide variety of external predictor components, and to integrate their contributions as additional evidence in the general decision-making on the optimal structural interpretation. We refer to this approach as hybrid parsing because it combines different kinds of linguistic models, which have been acquired in totally different ways, ranging from manually compiled rule sets to statistically trained components.

In this paper we investigate the benefit of external predictor components for the parsing quality which can be obtained with a rule-based grammar. For that purpose we trained a range of predictor components and integrated their output into the parser by means of soft constraints. Accordingly, the goal of our research was not to extensively optimize the predictor components themselves, but to quantify their contribution to the overall parsing quality. The results of these experiments not only lead to a better understanding of the utility of the different knowledge sources, but also allow us to derive empirically based priorities for further improving them. We are able to show that the potential of WCDG for information fusion is strong enough to accomodate even rather unreliable information from a wide range of predictor components. Using this potential we were able to reach a quality level for dependency parsing German which is unprecendented so far.

2 Hybrid Parsing

A hybridization seems advantageous even among purely stochastic models. Depending on their degree of sophistication, they can and must be trained on quite different kinds of data collections, which due to the necessary annotation effort are available in vastly different amounts: While training a probabilistic parser or a supertagger usually

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 321–328,
Sydney, July 2006.

requires a fully developed tree bank, in the case of taggers or chunkers a much more shallow and less expensive annotation suffices. Using a set of rather simple heuristics, a PP-attacher can even be trained on huge amounts of plain text.

Another reason for considering hybrid approaches is the influence that contextual factors might exert on the process of determining the most plausible sentence interpretation. Since this influence is dynamically changing with the environment, it can hardly be captured from available corpus data at all. To gain a benefit from such contextual cues, e.g. in a dialogue system, requires to integrate yet another kind of external information.

Unfortunately, stochastic predictor components are usually not perfect, at best producing preferences and guiding hints instead of reliable certainties. Integrating a number of them into a single systems poses the problem of error propagation. Whenever one component decides on the input of another, the subsequent one will most probably fail whenever the decision was wrong; if not, the erroneous information was not crucial anyhow. Dubey (2005) reported how serious this problem can be when he coupled a tagger with a subsequent parser, and noted that tagging errors are by far the most important source of parsing errors.

As soon as more than two components are involved, the combination of different error sources migth easily lead to a substantial decrease of the overall quality instead of achieving the desired synergy. Moreover, the likelihood of conflicting contributions will rise tremendously the more predictor components are involved. Therefore, it is far from obvious that additional information always helps. Certainly, a processing regime is needed which can deal with conflicting information by taking its reliability (or relative strength) into account. Such a preference-based decision procedure would then allow stronger valued evidence to override weaker one.

3 WCDG

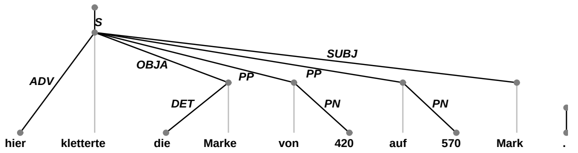
An architecture which fulfills this requirement is *Weighted Constraint Dependency Grammar*, which was based on a model originally proposed by Maruyama (1990) and later extended with weights (Schröder, 2002). A WCDG models natural language as *labelled dependency trees* on words, with no intermediate constituents assumed. It is entirely *declarative*: it only contains rules (called *constraints*) that explicitly describe the properties of well-formed trees, but no derivation rules. For instance, a constraint can state that determiners must precede their regents, or that there cannot be two determiners for the same regent, or that a determiner and its regent must agree in number, or that a countable noun must have a determiner. Further details can be found in (Foth, 2004). There is only a trivial generator component which enumerates all possible combinations of labelled word-to-word subordinations; among these any combination that satisfies the constraints is considered a correct analysis.

Constraints on trees can be *hard* or *soft*. Of the examples above, the first two should probably be considered hard, but the last two could be made defeasible, particularly if a robust coverage of potentially faulty input is desired. When two alternative analyses of the same input violate different constraints, the one that satisfies the more important constraint should be preferred. WCDG ensures this by assigning every analysis a score that is the product of the weights of all instances of constraint failures. Parsing tries to retrieve the analysis with the highest score.

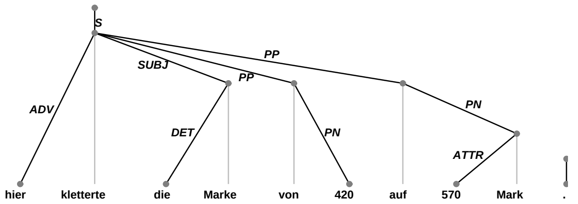
The weight of a constraint is usually determined by the grammar writer as it is formulated. Rules whose violation would produce nonsensical structures are usually made hard, while rules that enforce preferred but not required properties receive less weight. Obviously this classification depends on the purpose of a parsing system; a prescriptive language definition would enforce grammatical principles such as agreement with hard constraints, while a robust grammar must allow violations but disprefer them via soft constraints. In practice, the precise weight of a constraint is not particularly important as long as the relative importance of two rules is clearly reflected in their weights (for instance, a misinflected determiner is a language error, but probably a less severe one than duplicate determiners). There have been attempts to compute the weights of a WCDG automatically by observing which weight vectors perform best on a given corpus (Schröder et al., 2001), but weights computed completely automatically failed to improve on the original, hand-scored grammar.

Weighted constraints provide an ideal interface to integrate arbitrary predictor components in a soft manner. Thus, external predictions are treated the same way as grammar-internal preferences, e.g. on word order or distance. In contrast to a filtering approach such a strong integration does not blindly rely on the available predictions but is able to question them as long as there is strong enough combined evidence from the grammar and the other predictor components.

For our investigations, we used the reference implementation of WCDG available from `http://nats-www.informatik.uni-hamburg.de/download`, which allows constraints to express any formalizable property of a dependency tree. This great expressiveness has the disadvantage that the parsing problem becomes $\mathcal{NP}$-complete and cannot be solved efficiently. However, good success has been achieved with transformation-based solution methods that start out with an educated guess about the optimal tree and use constraint failures as cues where to change labels, subordinations, or lexical readings. As an example we show intermediate and final analyses of a sentence from our test set (negra-s18959): 'Hier kletterte die Marke von 420 auf 570 Mark.' (*Here the figure rose from 420 to 570 DM*).



In the first analysis, subject and object relations are analysed wrongly, and the noun phrase '570 Mark' has not been recognized. The analysis is imperfect because the common noun 'Mark' lacks a Determiner.



The final analysis correctly takes '570 Mark' as the kernel of the last preposition, and 'Marke' as the subject. Altogether, three dependency edges had to be changed to arrive at this solution.

Figure 1 shows the pseudocode of the best solution algorithm for WCDG described so far (Foth et al., 2000). Although it cannot guarantee to find the best solution to the constraint satisfaction problem, it requires only limited space and can be interrupted at any time and still returns a solution. If not interrupted, the algorithm terminates when no constraints with a weight less than a predefined threshold are violated. In contrast, a complete search usually requires more time and space than available, and often fails to return a usable result at all. All experiments described in this paper were conducted with the transformational search.

```
$A$ := the set of levels of analysis
$W$:= the set of all lexical readings of words in the sentence
$L$ := the set of defined dependency labels
$E := A \times W \times W \times L$ = the base set of dependency edges
$D := A \times W$ = the set of domains $d_{a,w}$ of all constraint variables
$B := \emptyset$ = the best analysis found
$C := \emptyset$ = the current analysis

{ Create the search space. }
for $e \in E$
    if $\mathrm{eval}(e) > 0$
    then $d_{a,w} := d_{a,w} \cup \{e\}$

{ Build initial analysis. }
for $d_{a,w} \in D$
    $e_0 = \arg\max_{e \in d_{a,w}} \mathrm{score}(C \cup \{e\})$
    $C := C \cup \{e_0\}$
$B := C$
$T := \emptyset$ = tabu set of conflicts removed so far.
$U := \emptyset$ = set of unremovable conflicts.
$i$ := the penalty threshold above which conflicts are ignored.
$n := 0$

{ Remove conflicts. }
while $\exists\, c \in \mathrm{eval}(C) \setminus U : \mathrm{penalty}(c) > i$
    and no interruption occurred

    { Determine which conflict to resolve. }
    $c_n := \arg\max_{c \in \mathrm{eval}(C) \setminus U} \mathrm{penalty}(c)$
    $T := T \cup \{c\}$

    { Find the best resolution set. }
    $R_n := \arg\max_{R \in \times\, \mathrm{domains}(c_n)} \mathrm{score(replace(C, R))}$
        where replace$(C, R)$ does not cause any $c \in T$
        and $|R \setminus C| <= 2$

    if no $R_n$ can be found

        { Consider $c_0$ unremovable. }
        $n := 0, C := B, T := \emptyset, U := U \cup \{c_0\}$
    else

        { Take a step. }
        $n := n + 1, C := \mathrm{replace}(C, R_n)$
        if $\mathrm{score}(C) > \mathrm{score}(B)$
            $n := 0, B := C, T := \emptyset, U := U \cap \mathrm{eval}(C)$

return $B$
```

Figure 1: Basic algorithm for heuristic transformational search.

For our investigation we use a comprehensive grammar of German expressed in about 1,000 constraints (Foth et al., 2005). It is intended to cover modern German completely and to be robust against many kinds of language error. A large WCDG such as this that is written entirely by hand can describe natural language with great precision, but at the price of very great effort for the grammar writer. Also, because many incorrect analyses are allowed, the space of possible trees becomes even larger than it would be for a prescriptive grammar.

4 Predictor components

Many rules of a language have the character of general preferences so weak that they are easily overlooked even by a language expert; for instance, the ordering of elements in the German *mittelfeld* is subject to several types of preference rules. Other regularities depend crucially on the lexical identity of the words concerned; modelling these fully would require the writing of a specific constraint for each word, which is all but infeasible. Empirically obtained information about the behaviour of a language would be welcome in such cases where manual constraints are not obvious or would require too much effort. This has already been demonstrated for the case of part-of-speech tagging: because contextual cues are very effective in determining the categories of ambiguous words, purely stochastical models can achieve a high accuracy. (Hagenström and Foth, 2002) show that the TnT tagger (Brants, 2000) can be profitably integrated into WCDG parsing: A constraint that prefers analyses which conform to TnT's category predictions can greatly reduce the number of spurious readings of lexically ambiguous words. Due to the soft integration of the tagger, though, the parser is not forced to accept its predictions unchallenged, but can override them if the wider syntactic context suggests this. In our experiments (line 1 in Table 1) this happens 75 times; 52 of these cases were actual errors committed by the tagger. These advantages taken together made the tagger the by far most valuable information source, whithout which the analysis of arbitrary input would not be feasible at all. Therefore, we use this component (POS) in all subsequent experiments.

Starting from this observation, we extended the idea to integrate several other external components that predict particular aspects of syntax analyses. Where possible, we re-used publicly available components to make the predictions rather than construct the best predictors possible; it is likely that better predictors could be found, but components 'off the shelf' or written in the simplest workable way proved enough to demonstrate a positive benefit of the technique in each case.

For the task of predicting the boundaries of major constituents in a sentence (chunk parsing, CP), we used the decision tree model TreeTagger (Schmid, 1994), which was trained on articles from *Stuttgarter Zeitung*. The noun, verb and prepositional chunk boundaries that it predicts are fed into a constraint which requires all chunk heads to be attached outside the current chunk, and all other words within it. Obviously such information can greatly reduce the number of structural alternatives that have to be considered during parsing. On our test set, the TreeTagger achieves a precision of 88.0% and a recall of 89.5%.

Models for category disambiguation can easily be extended to predict not only the syntactic category, but also the local syntactic environment of each word (supertagging). Supertags have been successfully applied to guide parsing in symbolic frameworks such as Lexicalised Tree-Adjoning grammar (Bangalore and Joshi, 1999). To obtain and evaluate supertag predictions, we re-trained the TnT Tagger on the combined NEGRA and TIGER treebanks (1997; 2002). Putting aside the standard NEGRA test set, this amounts to 59,622 sentences with 1,032,091 words as training data. For each word in the training set, the local context was extracted and encoded into a linear representation. The output of the retrained TnT then predicts the label of each word, whether it follows or precedes its regent, and what other types of relations are found below it. Each of these predictions is fed into a constraint which weakly prefers dependencies that do not violate the respective prediction (ST). Due to the high number of 12947 supertags in the maximally detailed model, the accuracy of the supertagger for complete supertags is as low as 67.6%. Considering that a detailed supertag corresponds to several distinct predictions (about label, direction etc.), it might be more appropriate to measure the average accuracy of these distinct predictions; by this measure, the individual predictions of the supertagger are 84.5% accurate; see (Foth et al., 2006) for details.

As with many parsers, the attachment of prepositions poses a particular problem for the base WCDG of German, because it is depends largely upon lexicalized information that is not widely used in its constraints. However, such information

Predictors	Reannotated Dependencies	Transformed Dependencies
1: POS only	89.7%/87.9%	88.3%/85.6%
2: POS+CP	90.2%/88.4%	88.7%/86.0%
3: POS+PP	90.9%/89.1%	89.6%/86.8%
4: POS+ST	92.1%/90.7%	90.7%/88.5%
5: POS+SR	91.4%/90.0%	90.0%/87.7%
6: POS+PP+SR	91.6%/90.2%	90.1%/87.8%
7: POS+ST+SR	92.3%/90.9%	90.8%/88.8%
8: POS+ST+PP	92.1%/90.7%	90.7%/88.5%
9: all five	92.5%/91.1%	91.0%/89.0%

Table 1: Structural/labelled parsing accuracy with various predictor components.

can be automatically extracted from large corpora of trees or even raw text: prepositions that tend to occur in the vicinity of specific nouns or verbs more often than chance would suggest can be assumed to modify those words preferentially (Volk, 2002).

A simple probabilistic model of PP attachment (PP) was used that counts only the occurrences of prepositions and potential attachment words (ignoring the information in the kernel noun of the PP). It was trained on both the available tree banks and on 295,000,000 words of raw text drawn from the taz corpus of German newspaper text. When used to predict the probability of the possible regents of each preposition in each sentence, it achieved an accuracy of 79.4% and 78.3%, respectively (see (Foth and Menzel, 2006) for details). The predictions were integrated into the grammar by another constraint which disprefers all possible regents to the corresponding degree (except for the predicted regent, which is not penalized at all).

Finally, we used a full dependency parser in order to obtain structural predictions for *all* words, and not merely for chunk heads or prepositions. We constructed a probabilistic shift-reduce parser (SR) for labelled dependency trees using the model described by (Nivre, 2003): from all available dependency trees, we reconstructed the series of parse actions (shift, reduce and attach) that would have constructed the tree, and then trained a simple maximum-likelihood model that predicts parse actions based on features of the current state such as the categories of the current and following words, the environment of the top stack word constructed so far, and the distance between the top word and the next word. This oracle parser achieves a structural and labelled accuracy of 84.8%/80.5% on the test set but can only predict projective dependency trees, which causes problems with about 1% of the edges in the 125,000 dependency trees used for training; in the interest of simplicity we did not address this issue specially, instead relying on the ability of the WCDG parser to robustly integrate even predictions which are wrong by definition.

5 Evaluation

Since the WCDG parser never fails on typical treebank sentences, and always delivers an analysis that contains exactly one subordination for each word, the common measures of precision, recall and f-score all coincide; all three are summarized as *accuracy* here. We measure the *structural* (i.e. unlabelled) accuracy as the ratio of correctly attached words to all words; the *labelled* accuracy counts only those words that have the correct regent and also bear the correct label. For comparison with previous work, we used the next-to-last 1,000 sentences of the NEGRA corpus as our test set. Table 1 shows the accuracy obtained.[1]

The gold standard used for evaluation was derived from the annotations of the NEGRA treebank (version 2.0) in a semi-automatic procedure. First, the NEGRA phrase structures were automatically transformed to dependency trees with the DEPSY tool (Daum et al., 2004). However, before the parsing experiments, the results were manually corrected to (1) take care of systematic inconsistencies between the NEGRA annotations and the WCDG annotations (e.g. for nonprojectivities, which in our case are used only if necessary for an ambiguity free attachment of verbal arguments, relative clauses and coordinations, but not for other types of adjuncts) and (2) to remove inconsistencies with NEGRAs own annotation guidelines (e.g. with regard to elliptical and co-ordinated structures, adverbs and subordinated main clauses.) To illustrate the consequences of these corrections we report in Table 1 both kinds of results: those obtained on our WCDG-conform annotations (reannotated) and the others on the raw output of the automatic conversion (trans-

[1] Note that the POS model employed by TnT was trained on the entire NEGRA corpus, so that there is an overlap between the training set of TnT and the test set of the parser. However, control experiments showed that a POS model trained on the NEGRA and TIGER treebanks minus the test set results in the same parsing accuracy, and in fact slightly better POS accuracy. All other statistical predictors were trained on data disjunct from the test set.

formed), although the latter ones introduce a systematic mismatch between the gold standard and the design principles of the grammar.

The experiments 2–5 show the effect of adding the POS tagger and one of the other predictor components to the parser. The chunk parser yields only a slight improvement of about 0.5% accuracy; this is most probably because the baseline parser (line 1) does not make very many mistakes at this level anyway. For instance, the relation type with the highest error rate is prepositional attachment, about which the chunk parser makes no predictions at all. In fact, the benefit of the PP component alone (line 3) is much larger even though it predicts *only* the regents of prepositions. The two other components make predictions about all types of relations, and yield even bigger benefits.

When more than one other predictor is added to the grammar, the beneft is generally higher than that of either alone, but smaller than the sum of both. An exception is seen in line 8, where the combination of POS tagging, supertagging and PP prediction fails to better the results of just POS tagging and supertagging (line 4). Individual inspection of the results suggests that the lexicalized information of the PP attacher is often counteracted by the less informed predictions of the supertagger (this was confirmed in preliminary experiments by a gain in accuracy when prepositions were exempted from the supertag constraint). Finally, combining all five predictors results in the highest accuracy of all, improving over the first experiment by 2.8% and 3.2% for structural and labelled accuracy respectively.

We see that the introduction of stochastical information into the handwritten language model is generally helpful, although the different predictors contribute different types of information. The POS tagger and PP attacher capture lexicalized regularities which are genuinely new to the grammar: in effect, they refine the language model of the grammar in places that would be tedious to describe through individual rules. In contrast, the more global components tend to make the same predictions as the WCDG itself, only explicitly. This guides the parser so that it tends to check the correct alternative first more often, and has a greater chance of finding the global optimum. This explains why their addition increases parsing accuracy even when their own accuracy is markedly lower than even the baseline (line 1).

6 Related work

The idea of integrating knowledge sources of different origin is not particularly new. It has been successfully used in areas like speech recognition or statistical machine translation where acoustic models or bilingual mappings have to be combined with (monolingual) language models. A similar architecture has been adopted by (Wang and Harper, 2004) who train an n-best supertagger and an attachment predictor on the Penn Treebank and obtain an labelled F-score of 92.4%, thus slightly outperforming the results of (Collins, 1999) who obtained 92.0% on the same sentences, but evaluating on transformed phrase structure trees instead on directly computed dependency relations.

Similar to our approach, the result of (Wang and Harper, 2004) was achieved by integrating the evidence of two (stochastic) components into a single decision procedure on the optimal interpretation. Both, however, have been trained on the very same data set. Combining more than two different knowledge sources into a system for syntactic parsing to our knowledge has never been attempted so far. The possible synergy between different knowledge sources is often assumed but viable alternatives to filtering or selection in a pipelined architecture have not yet been been demonstrated successfully. Therefore, external evidence is either used to restrict the space of possibilities for a subsequent component (Clark and Curran, 2004) or to choose among the alternative results which a traditional rule-based parser usually delivers (Malouf and van Noord, 2004). In contrast to these approaches, our system directly integrates the available evidence into the decision procedure of the rule-based parser by modifying the objective function in a way that helps guiding the parsing process towards the desired interpretation. This seems to be crucial for being able to extend the approach to multiple predictors.

An extensive evaluation of probabilistic dependency parsers has recently been carried out within the framework of the 2006 CoNLL shared task (see `http://nextens.uvt.nl/~conll`). Most successful for many of the 13 different languages has been the system described in (McDonald et al., 2005). This approach is based on a procedure for online large margin learning and considers a huge number of locally available features to predict dependency attachments without being restricted to projective structures. For German it achieves 87.34% labelled and 90.38% unlabelled attachment accuracy. These results are particularly impressive, since due to the strictly local evaluation of attachment hypotheses the runtime complexity of the parser is only $\mathcal{O}(n^2)$.

Although a similar source of text has been used for this evaluation (newspaper), the numbers cannot be directly compared to our results since both the test set and the annotation guidelines differ from those used in our experiments. Moreover, the different methodologies adopted for system development clearly favour a manual grammar development, where more lexical resources are available and because of human involvement a perfect isolation between test and training data can only be guaranteed for the probabilistic components. On the other hand CoNLL restricted itself to the easier attachment task and therefore provided the gold standard POS tag as part of the input data, whereas in our case pure word form sequences are analysed and POS disambiguation is part of the task to be solved. Finally, punctuation has been ignored in the CoNLL evaluation, while we included it in the attachment scores. To compensate for the last two effects we re-evaluated our parser without considering punctuation but providing it with perfect POS tags. Thus, under similar conditions as used for the CoNLL evaluation we achieved a labelled accuracy of 90.4% and an unlabelled one of 91.9%.

Less obvious, though, is a comparison with results which have been obtained for phrase structure trees. Here the state of the art for German is defined by a system which applies treebank transformations to the original NEGRA treebank and extends a Collins-style parser with a suffix analysis (Dubey, 2005). Using the same test set as the one described above, but restricting the maximum sentence length to 40 and providing the correct POS tag, the system achieved a labelled bracket F-score of 76.3%.

7 Conclusions

We have presented an architecture for the fusion of information contributed from a variety of components which are either based on expert knowledge or have been trained on quite different data collections. The results of the experiments show that there is a high degree of synergy between these different contributions, even if they themselves are fairly unreliable. Integrating all the available predictors we were able to improve the overall labelled accuracy on a standard test set for German to 91.1%, a level which is as least as good as the results reported for alternative approaches to parsing German.

The result we obtained also challenges the common perception that rule-based parsers are necessarily inferior to stochastic ones. Supplied with appropriate helper components, the WCDG parser not only reached a surprisingly high level of output quality but in addition appears to be fairly stable against changes in the text type it is applied to (Foth et al., 2005).

We attribute the successful integration of different information sources primarily to the fundamental ability of the WCDG grammar to combine evidence in a soft manner. If unreliable information needs to be integrated, this possibility is certainly an undispensible prerequisite for preventing local errors from accumulating and leading to an unacceptably low degree of reliability for the whole system eventually. By integrating the different predictors into the WCDG parsers's general mechanism for evidence arbitration, we not only avoided the adverse effect of individual error rates multiplying out, but instead were able to even raise the degree of output quality substantially.

From the fact that the combination of all predictor components achieved the best results, even if the individual predictions are fairly unreliable, we can also conclude that diversity in the selection of predictor components is more important than the reliability of their contributions. Among the available predictor components which could be integrated into the parser additionally, the approach of (McDonald et al., 2005) certainly looks most promising. Compared to the shift-reduce parser which has been used as one of the predictor components for our experiments, it seems particularly attractive because it is able to predict non-projective structures without any additional provision, thus avoiding the misfit between our (non-projective) gold standard annotations and the restriction to projective structures that our shift-reduce parser suffers from.

Another interesting goal of future work might be to even consider dynamic predictors, which can change their behaviour according to text type and perhaps even to text structure. This, however, would also require extending and adapting the currently dominating standard scenario of parser evaluation substantially.

References

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2):237–265.

Thorsten Brants, Roland Hendriks, Sabine Kramp, Brigitte Krenn, Cordula Preis, Wojciech Skut, and Hans Uszkoreit. 1997. Das NEGRA-Annotationsschema. Negra project report, Universität des Saarlandes, Computerlinguistik, Saarbrücken, Germany.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol.

Thorsten Brants. 2000. TnT – A Statistical Part-of-Speech Tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*, Seattle, WA, USA.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. NAACL-2000*.

Stephen Clark and James R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing. In *Proc. 20th Int. Conf. on Computational Linguistics, Coling-2004*.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Phd thesis, University of Pennsylvania, Philadephia, PA.

Michael Daum, Kilian Foth, and Wolfgang Menzel. 2004. Automatic transformation of phrase treebanks to dependency trees. In *Proc. 4th Int. Conf. on Language Resources and Evaluation, LREC-2004*, Lisbon, Portugal.

Amit Dubey. 2005. What to do when lexicalization fails: parsing German with suffix analysis and smoothing. In *Proc. 43rd Annual Meeting of the ACL*, Ann Arbor, MI.

Kilian Foth and Wolfgang Menzel. 2006. The benefit of stochastic PP-attachment to a rule-based parser. In *Proc. 21st Int. Conf. on Computational Linguistics, Coling-ACL-2006*, Sydney.

Kilian A. Foth, Wolfgang Menzel, and Ingo Schröder. 2000. A Transformation-based Parsing Technique with Anytime Properties. In *4th Int. Workshop on Parsing Technologies, IWPT-2000*, pages 89 – 100.

Kilian Foth, Michael Daum, and Wolfgang Menzel. 2005. Parsing unrestricted German text with defeasible constraints. In H. Christiansen, P. R. Skadhauge, and J. Villadsen, editors, *Constraint Solving and Language Processing*, volume 3438 of *Lecture Notes in Artificial Intelligence*, pages 140–157. Springer-Verlag, Berlin.

Kilian Foth, Tomas By, and Wolfgang Menzel. 2006. Guiding a constraint dependency parser with supertags. In *Proc. 21st Int. Conf. on Computational Linguistics, Coling-ACL-2006*, Sydney.

Kilian Foth. 2004. Writing Weighted Constraints for Large Dependency Grammars. In *Proc. Recent Advances in Dependency Grammars, COLING-Workshop 2004*, Geneva, Switzerland.

Jochen Hagenström and Kilian A. Foth. 2002. Tagging for robust parsers. In *Proc. 2nd. Int. Workshop, Robust Methods in Analysis of Natural Language Data, ROMAND-2002*.

Robert Malouf and Gertjan van Noord. 2004. Wide coverage parsing with stochastic attribute value grammars. In *Proc. IJCNLP-04 Workshop Beyond Shallow Analyses - Formalisms and statistical modeling for deep analyses*, Sanya City, China.

Hiroshi Maruyama. 1990. Structural disambiguation with constraint propagation. In *Proc. 28th Annual Meeting of the ACL (ACL-90)*, pages 31–38, Pittsburgh, PA.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proc. Human Language Technology Conference / Conference on Empirical Methods in Natural Language Processing, HLT/EMNLP-2005*, Vancouver, B.C.

Joakim Nivre. 2003. An Efficient Algorithm for Projective Dependency Parsing. In *Proc. 4th International Workshop on Parsing Technologies, IWPT-2003*, pages 149–160.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Int. Conf. on New Methods in Language Processing*, Manchester, UK.

Ingo Schröder, Horia F. Pop, Wolfgang Menzel, and Kilian Foth. 2001. Learning grammar weights using genetic algorithms. In *Proceedings Euroconference Recent Advances in Natural Language Processing*, pages 235–239, Tsigov Chark, Bulgaria.

Ingo Schröder. 2002. *Natural Language Parsing with Graded Constraints*. Ph.D. thesis, Dept. of Computer Science, University of Hamburg, Germany.

Martin Volk. 2002. Combining unsupervised and supervised methods for pp attachment disambiguation. In *Proc. of COLING-2002*, Taipeh.

Wen Wang and Mary P. Harper. 2004. A statistical constraint dependency grammar (CDG) parser. In *Proc. ACL Workshop Incremental Parsing: Bringing Engineering and Cognition Together*, pages 42–49, Barcelona, Spain.

Error mining in parsing results

Benoît Sagot and Éric de la Clergerie
Projet ATOLL - INRIA
Domaine de Voluceau, B.P. 105
78153 Le Chesnay Cedex, France
{benoit.sagot,eric.de_la_clergerie}@inria.fr

Abstract

We introduce an error mining technique for automatically detecting errors in resources that are used in parsing systems. We applied this technique on parsing results produced on several million words by two distinct parsing systems, which share the syntactic lexicon and the pre-parsing processing chain. We were thus able to identify missing and erroneous information in these resources.

1 Introduction

Natural language parsing is a hard task, partly because of the complexity and the volume of information that have to be taken into account about words and syntactic constructions. However, it is necessary to have access to such information, stored in resources such as lexica and grammars, and to try and minimize the amount of missing and erroneous information in these resources. To achieve this, the use of these resources at a large-scale in parsers is a very promising approach (van Noord, 2004), and in particular the analysis of situations that lead to a parsing failure: one can learn from one's own mistakes.

We introduce a probabilistic model that allows to identify forms and form bigrams that may be the source of errors, thanks to a corpus of parsed sentences. In order to facilitate the exploitation of forms and form bigrams detected by the model, and in particular to identify causes of errors, we have developed a visualization environment. The whole system has been tested on parsing results produced for several multi-million-word corpora and with two different parsers for French, namely SxLFG and FRMG.

However, the error mining technique which is the topic of this paper is fully system- and language-independent. It could be applied without any change on parsing results produced by any system working on any language. The only information that is needed is a boolean value for each sentence which indicates if it has been successfully parsed or not.

2 Principles

2.1 General idea

The idea we implemented is inspired from (van Noord, 2004). In order to identify missing and erroneous information in a parsing system, one can analyze a large corpus and study with statistical tools what differentiates sentences for which parsing succeeded from sentences for which it failed.

The simplest application of this idea is to look for forms, called *suspicious forms*, that are found more frequently in sentences that could not be parsed. This is what van Noord (2004) does, without trying to identify a suspicious form in any sentence whose parsing failed, and thus without taking into account the fact that there is (at least) one cause of error in each unparsable sentence.[1] On the contrary, we will look, in each sentence on which parsing failed, for the form that has the highest probability of being the cause of this failure: it is the *main suspect* of the sentence. This form may be incorrectly or only partially described in the lexicon, it may take part in constructions that are not described in the grammar, or it may exemplify imperfections of the pre-syntactic processing chain. This idea can be easily extended to sequences of forms, which is what we do by tak-

[1] Indeed, he defines the suspicion rate of a form f as the rate of unparsable sentences among sentences that contain f.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 329–336,
Sydney, July 2006.

ing form bigrams into account, but also to lemmas (or sequences of lemmas).

2.2 Form-level probabilistic model

We suppose that the corpus is split in sentences, sentences being segmented in forms. We denote by s_i the i-th sentence. We denote by $o_{i,j}$, $(1 \leq j \leq |s_i|)$ the occurrences of forms that constitute s_i, and by $F(o_{i,j})$ the corresponding forms. Finally, we call error the function that associates to each sentence s_i either 1, if s_i's parsing failed, and 0 if it succeeded.

Let $\mathcal{O}_f$ be the set of the occurrences of a form f in the corpus: $\mathcal{O}_f = \{o_{i,j} | F(o_{i,j}) = f\}$. The number of occurrences of f in the corpus is therefore $|\mathcal{O}_f|$.

Let us define at first the *mean global suspicion rate* $\overline{S}$, that is the mean probability that a given occurrence of a form be the cause of a parsing failure. We make the assumption that the failure of the parsing of a sentence has a unique cause (here, a unique form…). This assumption, which is not necessarily exactly verified, simplifies the model and leads to good results. If we call $\mathrm{occ}_{\mathrm{total}}$ the total amount of forms in the corpus, we have then:

$$\overline{S} = \frac{\Sigma_i \mathrm{error}(s_i)}{\mathrm{occ}_{\mathrm{total}}}$$

Let f be a form, that occurs as the j-th form of sentence s_i, which means that $F(o_{i,j}) = f$. Let us assume that s_i's parsing failed: $\mathrm{error}(s_i) = 1$. We call *suspicion rate* of the j-th form $o_{i,j}$ of sentence s_i the probability, denoted by $S_{i,j}$, that the occurrence $o_{i,j}$ of form form f be the cause of the s_i's parsing failure. If, on the contrary, s_i's parsing succeeded, its occurrences have a suspicion rate that is equal to zero.

We then define the *mean suspicion rate* S_f of a form f as the mean of all suspicion rates of its occurrences:

$$S_f = \frac{1}{|\mathcal{O}_f|} \cdot \sum_{o_{i,j} \in \mathcal{O}_f} S_{i,j}$$

To compute these rates, we use a fix-point algorithm by iterating a certain amount of times the following computations. Let us assume that we just completed the n-th iteration: we know, for each sentence s_i, and for each occurrence $o_{i,j}$ of this sentence, the estimation of its suspicion rate $S_{i,j}$ as computed by the n-th iteration, estimation that is denoted by $S_{i,j}^{(n)}$. From this estimation, we compute the $n+1$-th estimation of the mean suspicion rate of each form f, denoted by $S_f^{(n+1)}$:

$$S_f^{(n+1)} = \frac{1}{|\mathcal{O}_f|} \cdot \sum_{o_{i,j} \in \mathcal{O}_f} S_{i,j}^{(n)}$$

This rate[2] allows us to compute a new estimation of the suspicion rate of all occurrences, by giving to each occurrence if a sentence s_i a suspicion rate $S_{i,j}^{(n+1)}$ that is exactly the estimation $S_f^{(n+1)}$ of the mean suspicion rate of S_f of the corresponding form, and then to perform a sentence-level normalization. Thus:

$$S_{i,j}^{(n+1)} = \mathrm{error}(s_i) \cdot \frac{S_{F(o_{i,j})}^{(n+1)}}{\sum_{1 \leq j \leq |s_i|} S_{F(o_{i,j})}^{(n+1)}}$$

At this point, the $n+1$-th iteration is completed, and we can resume again these computations, until convergence on a fix-point. To begin the whole process, we just say, for an occurrence $o_{i,j}$ of sentence s_i, that $S_{i,j}^{(0)} = \mathrm{error}(s_i)/|s_i|$. This means that for a non-parsable sentence, we start from a baseline where all of its occurrences have an equal probability of being the cause of the failure.

After a few dozens of iterations, we get stabilized estimations of the mean suspicion rate each form, which allows:

- to identify the forms that most probably cause errors,
- for each form f, to identify non-parsable sentences s_i where an occurrence $o_{i,j} \in \mathcal{O}_f$ of f is a main suspect and where $o_{i,j}$ has a very

[2] We also performed experiment in which S_f was estimated by an other estimator, namely the *smoothed mean suspicion rate*, denoted by $\tilde{S}_f^{(n)}$, that takes into account the number of occurrences of f. Indeed, the confidence we can have in the estimation $S_f^{(n)}$ is lower if the number of occurrences of f is lower. Hence the idea to smooth $S_f^{(n)}$ by replacing it with a weighted mean $\tilde{S}_f^{(n)}$ between $S_f^{(n)}$ and $\overline{S}$, where the weights λ and $1-\lambda$ depend on $|\mathcal{O}_f|$: if $|\mathcal{O}_f|$ is high, $\tilde{S}_f^{(n)}$ will be close from $S_f^{(n)}$; if it is low, it will be closer from $\overline{S}$:

$$\tilde{S}_f^{(n)} = \lambda(|\mathcal{O}_f|) \cdot S_f^{(n)} + (1 - \lambda(|\mathcal{O}_f|)) \cdot \overline{S}.$$

In these experiments, we used the smoothing function $\lambda(|\mathcal{O}_f|) = 1 - e^{-\beta|\mathcal{O}_f|}$ with $\beta = 0.1$. But this model, used with the ranking according to $M_f = S_f \cdot \ln|\mathcal{O}_f|$ (see below), leads results that are very similar to those obtained without smoothing. Therefore, we describe the smoothing-less model, which has the advantage not to use an empirically chosen smoothing function.

high suspicion rate among all occurrences of form f.

We implemented this algorithm as a *perl* script, with strong optimizations of data structures so as to reduce memory and time usage. In particular, form-level structures are shared between sentences.

2.3 Extensions of the model

This model gives already very good results, as we shall see in section 4. However, it can be extended in different ways, some of which we already implemented.

First of all, it is possible not to stick to forms. Indeed, we do not only work on forms, but on couples made out of a form (a lexical entry) and one or several token(s) that correspond to this form in the raw text (a token is a portion of text delimited by spaces or punctuation tokens).

Moreover, one can look for the cause of the failure of the parsing of a sentence not only in the presence of a form in this sentence, but also in the presence of a bigram[3] of forms. To perform this, one just needs to extend the notions of *form* and *occurrence*, by saying that a (generalized) form is a unigram or a bigram of forms, and that a (generalized) occurrence is an occurrence of a generalized form, i.e., an occurrence of a unigram or a bigram of forms. The results we present in section 4 includes this extension, as well as the previous one.

Another possible generalization would be to take into account facts about the sentence that are not simultaneous (such as form unigrams and form bigrams) but mutually exclusive, and that must therefore be probabilized as well. We have not yet implemented such a mechanism, but it would be very interesting, because it would allow to go beyond forms or n-grams of forms, and to manipulate also lemmas (since a given form has usually several possible lemmas).

3 Experiments

In order to validate our approach, we applied these principles to look for error causes in parsing results given by two deep parsing systems for French, FRMG and SXLFG, on large corpora.

3.1 Parsers

Both parsing systems we used are based on deep non-probabilistic parsers. They share:

- the Le*fff* 2 syntactic lexicon for French (Sagot et al., 2005), that contains 500,000 entries (representing 400,000 different forms) ; each lexical entry contains morphological information, sub-categorization frames (when relevant), and complementary syntactic information, in particular for verbal forms (controls, attributives, impersonals,. . .),
- the SXPipe pre-syntactic processing chain (Sagot and Boullier, 2005), that converts a raw text in a sequence of DAGs of forms that are present in the Le*fff* ; SXPipe contains, among other modules, a sentence-level segmenter, a tokenization and spelling-error correction module, named-entities recognizers, and a non-deterministic multi-word identifier.

But FRMG and SXLFG use completely different parsers, that rely on different formalisms, on different grammars and on different parser builder. Therefore, the comparison of error mining results on the output of these two systems makes it possible to distinguish errors coming from the Le*fff* or from SXPipe from those coming to one grammar or the other. Let us describe in more details the characteristics of these two parsers.

The FRMG parser (Thomasset and Villemonte de la Clergerie, 2005) is based on a compact TAG for French that is automatically generated from a meta-grammar. The compilation and execution of the parser is performed in the framework of the DYALOG system (Villemonte de la Clergerie, 2005).

The SXLFG parser (Boullier and Sagot, 2005b; Boullier and Sagot, 2005a) is an efficient and robust LFG parser. Parsing is performed in two steps. First, an Earley-like parser builds a shared forest that represents all constituent structures that satisfy the context-free skeleton of the grammar. Then functional structures are built, in one or more bottom-up passes. Parsing efficiency is achieved thanks to several techniques such as compact data representation, systematic use of structure and computation sharing, lazy evaluation and heuristic and almost non-destructive pruning during parsing.

Both parsers implement also advanced error recovery and tolerance techniques, but they were

[3] One could generalize this to n-grams, but as n gets higher the number of occurrences of n-grams gets lower, hence leading to non-significant statistics.

corpus	#sentences	#success (%)	#forms	#occ	$\overline{S}$ (%)	Date
MD/FRMG	330,938	136,885 (41.30%)	255,616	10,422,926	1.86%	Jul. 05
MD/SXLFG	567,039	343,988 (60.66%)	327,785	14,482,059	1.54%	Mar. 05
EASy/FRMG	39,872	16,477 (41.32%)	61,135	878,156	2.66%	Dec. 05
EASy/SXLFG	39,872	21,067 (52.84%)	61,135	878,156	2.15%	Dec. 05

Table 1: General information on corpora and parsing results

useless for the experiments described here, since we want only to distinguish sentences that receive a full parse (without any recovery technique) from those that do not.

3.2 Corpora

We parsed with these two systems the following corpora:

MD corpus : This corpus is made out of 14.5 million words (570,000 sentences) of general journalistic corpus that are articles from the *Monde diplomatique*.

EASy corpus : This is the 40,000-sentence corpus that has been built for the EASy parsing evaluation campaign for French (Paroubek et al., 2005). We only used the raw corpus (without taking into account the fact that a manual parse is available for 10% of all sentences). The EASy corpus contains several sub-corpora of varied style: journalistic, literacy, legal, medical, transcription of oral, e-mail, questions, etc.

Both corpora are raw in the sense that no cleaning whatsoever has been performed so as to eliminate some sequences of characters that can not really be considered as sentences.

Table 1 gives some general information on these corpora as well as the results we got with both parsing systems. It shall be noticed that both parsers did not parse exactly the same set and the same number of sentences for the MD corpus, and that they do not define in the exactly same way the notion of sentence.

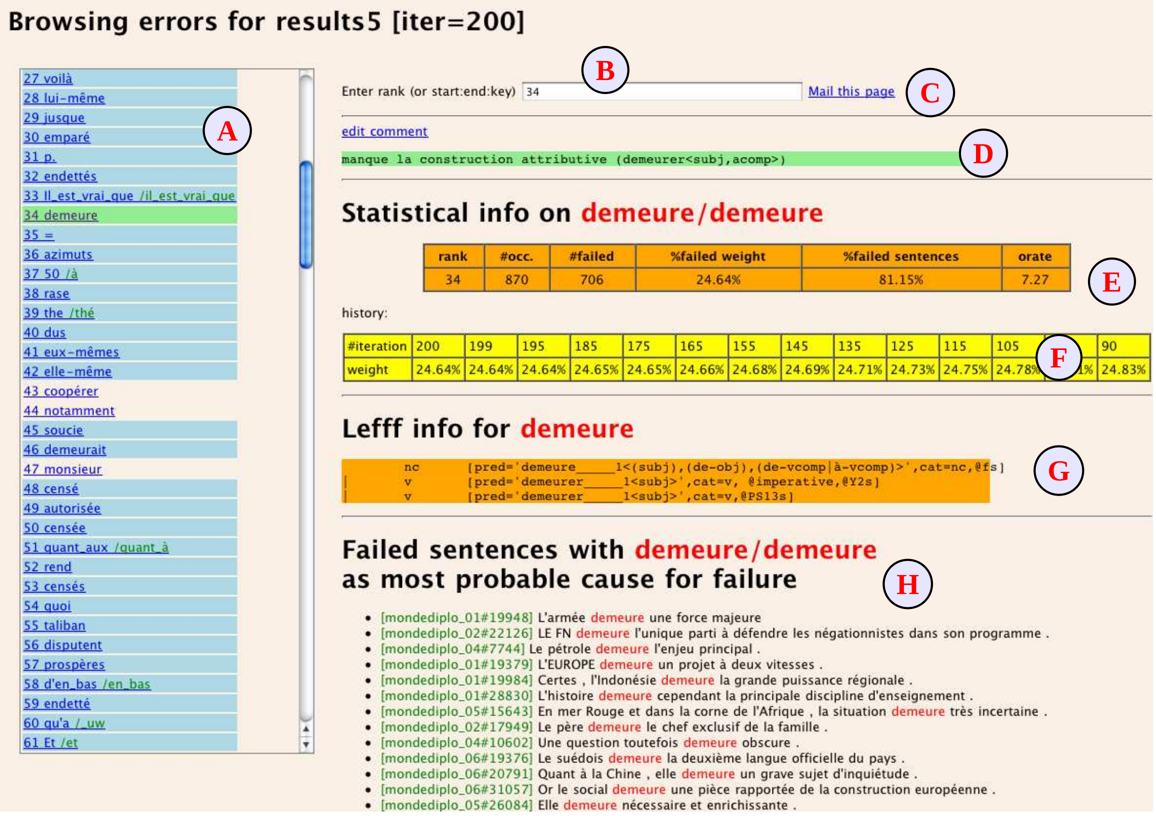
3.3 Results visualization environment

We developed a visualization tool for the results of the error mining, that allows to examine and annotate them. It has the form of an HTML page that uses dynamic generation methods, in particular *javascript*. An example is shown on Figure 1.

To achieve this, suspicious forms are ranked according to a measure M_f that models, for a given form f, the benefit there is to try and correct the (potential) corresponding error in the resources. A user who wants to concentrate on almost certain errors rather than on most frequent ones can visualize suspicious forms ranked according to $M_f = S_f$. On the contrary, a user who wants to concentrate on most frequent potential errors, rather than on the confidence that the algorithm has given to errors, can visualize suspicious forms ranked according to[4] $M_f = S_f|\mathcal{O}_f|$. The default choice, which is adopted to produce all tables shown in this paper, is a balance between these two possibilities, and ranks suspicious forms according to $M_f = S_f \cdot \ln|\mathcal{O}_f|$.

The visualization environment allows to browse through (ranked) suspicious forms in a scrolling list on the left part of the page (**A**). When the suspicious form is associated to a token that is the same as the form, only the form is shown. Otherwise, the token is separated from the form by the symbol “/”. The right part of the page shows various pieces of information about the currently selected form. After having given its rank according to the ranking measure M_f that has been chosen (**B**), a field is available to add or edit an annotation associated with the suspicious form (**D**). These annotations, aimed to ease the analysis of the error mining results by linguists and by the developers of parsers and resources (lexica, grammars), are saved in a database (SQLITE). Statistical information is also given about f (**E**), including its number of occurrences occ_f, the number of occurrences of f in non-parsable sentences, the final estimation of its mean suspicion rate S_f and the rate $\mathrm{err}(f)$ of non-parsable sentences among those where f appears. This indications are complemented by a brief summary of the iterative process that shows the convergence of the successive estimations of S_f. The lower part of the page gives a mean to identify the cause of f-related errors by showing

[4] Let f be a form. The suspicion rate S_f can be considered as the probability for a particular occurrence of f to cause a parsing error. Therefore, $S_f|\mathcal{O}_f|$ models the number of occurrences of f that do cause a parsing error.



Browsing errors for results5 [iter=200]

27 voilà
28 lui-même
29 jusque
30 emparé
31 p.
32 endettés
33 Il_est_vrai_que /il_est_vrai_que
34 demeure
35 =
36 azimuts
37 50 /à
38 rase
39 the /thé
40 dus
41 eux-mêmes
42 elle-même
43 coopérer
44 notamment
45 soucie
46 demeurait
47 monsieur
48 censé
49 autorisée
50 censée
51 quant_aux /quant_à
52 rend
53 censés
54 quoi
55 taliban
56 disputent
57 prospères
58 d'en_bas /en_bas
59 endetté
60 qu'a /_uw
61 Et /et

Enter rank (or start:end:key) 34 Mail this page

edit comment

```
manque la construction attributive (demeurer<subj,acomp>)
```

Statistical info on demeure/demeure

rank	#occ.	#failed	%failed weight	%failed sentences	orate
34	870	706	24.64%	81.15%	7.27

history:

#iteration	200	199	195	185	175	165	155	145	135	125	115	105	[illegible]	90
weight	24.64%	24.64%	24.64%	24.65%	24.65%	24.66%	24.68%	24.69%	24.71%	24.73%	24.75%	24.78%	[illegible]1%	24.83%

Lefff info for demeure

```
        nc      [pred='demeure_____1<(subj),(de-obj),(de-vcomp|à-vcomp)>',cat=nc,@fs]
|       v       [pred='demeurer_____1<subj>',cat=v, @imperative,@Y2s]
|       v       [pred='demeurer_____1<subj>',cat=v,@PS13s]
```

Failed sentences with demeure/demeure as most probable cause for failure

- [mondediplo_01#19948] L'armée demeure une force majeure
- [mondediplo_02#22126] LE FN demeure l'unique parti à défendre les négationnistes dans son programme .
- [mondediplo_04#7744] Le pétrole demeure l'enjeu principal .
- [mondediplo_01#19379] L'EUROPE demeure un projet à deux vitesses .
- [mondediplo_01#19984] Certes , l'Indonésie demeure la grande puissance régionale .
- [mondediplo_01#28830] L'histoire demeure cependant la principale discipline d'enseignement .
- [mondediplo_05#15643] En mer Rouge et dans la corne de l'Afrique , la situation demeure très incertaine .
- [mondediplo_02#17949] Le père demeure le chef exclusif de la famille .
- [mondediplo_04#10602] Une question toutefois demeure obscure .
- [mondediplo_06#19376] Le suédois demeure la deuxième langue officielle du pays .
- [mondediplo_06#20791] Quant à la Chine , elle demeure un grave sujet d'inquiétude .
- [mondediplo_06#31057] Or le social demeure une pièce rapportée de la construction européenne .
- [mondediplo_05#26084] Elle demeure nécessaire et enrichissante .

Figure 1: Error mining results visualization environment (results are shown for MD/FRMG).

f's entries in the Le*fff* lexicon (**G**) as well as non-parsable sentences where f is the main suspect and where one of its occurrences has a particularly high suspicion rate[5] (**H**).

The whole page (with annotations) can be sent by e-mail, for example to the developer of the lexicon or to the developer of one parser or the other (**C**).

4 Results

In this section, we mostly focus on the results of our error mining algorithm on the parsing results provided by SxLfg on the MD corpus. We first present results when only forms are taken into account, and then give an insight on results when both forms and form bigrams are considered.

4.1 Finding suspicious forms

The execution of our error mining script on MD/SxLfg, with $i_{max} = 50$ iterations and when only (isolated) forms are taken into account, takes less than one hour on a 3.2 GHz PC running Linux with a 1.5 Go RAM. It outputs 18,334 *relevant* suspicious forms (out of the 327,785 possible ones), where a relevant suspicious form is defined as a form f that satisfies the following arbitrary constraints:[6] $S_f^{(i_{max})} > 1,5 \cdot \overline{S}$ and $|\mathcal{O}_f| > 5$.

We still can not prove theoretically the convergence of the algorithm.[7] But among the 1000 best-ranked forms, the last iteration induces a mean variation of the suspicion rate that is less than 0.01%.

On a smaller corpus like the EASy corpus, 200 iterations take 260s. The algorithm outputs less than 3,000 relevant suspicious forms (out of the 61,125 possible ones). Convergence information

[5] Such an information, which is extremely valuable for the developers of the resources, can not be obtained by global (form-level and not occurrence-level) approaches such as the $\mathrm{err}(f)$-based approach of (van Noord, 2004). Indeed, enumerating all sentences which include a given form f, and which did not receive a full parse, is not precise enough: it would show at the same time sentences wich fail because of f (e.g., because its lexical entry lacks a given sub-categorization frame) and sentences which fail for an other independent reason.

[6] These constraints filter results, but all forms are taken into account during all iterations of the algorithm.

[7] However, the algorithms shares many common points with iterative algorithm that are known to converge and that have been proposed to find maximum entropy probability distributions under a set of constraints (Berger et al., 1996). Such an algorithm is compared to ours later on in this paper.

is the same as what has been said above for the MD corpus.

Table 2 gives an idea of the repartition of suspicious forms w.r.t. their frequency (for FRMG on MD), showing that rare forms have a greater probability to be suspicious. The most frequent suspicious form is the double-quote, with (only) $S_f = 9\%$, partly because of segmentation problems.

4.2 Analyzing results

Table 3 gives an insight on the output of our algorithm on parsing results obtained by SXLFG on the MD corpus. For each form f (in fact, for each couple of the form *(token,form)*), this table displays its suspicion rate and its number of occurrences, as well as the rate $\mathrm{err}(f)$ of non-parsable sentences among those where f appears and a short manual analysis of the underlying error.

In fact, a more in-depth manual analysis of the results shows that they are very good: errors are correctly identified, that can be associated with four error sources: (1) the Le*fff* lexicon, (2) the SXPipe pre-syntactic processing chain, (3) imperfections of the grammar, but also (4) problems related to the corpus itself (and to the fact that it is a raw corpus, with meta-data and typographic noise).

On the EASy corpus, results are also relevant, but sometimes more difficult to interpret, because of the relative small size of the corpus and because of its heterogeneity. In particular, it contains e-mail and oral transcriptions sub-corpora that introduce a lot of noise. Segmentation problems (caused both by SXPipe and by the corpus itself, which is already segmented) play an especially important role.

4.3 Comparing results with results of other algorithms

In order to validate our approach, we compared our results with results given by two other relevant algorithms:

- van Noord's (van Noord, 2004) (form-level and non-iterative) evaluation of $\mathrm{err}(f)$ (the rate of non-parsable sentences among sentences containing the form f),
- a standard (occurrence-level and iterative) maximum entropy evaluation of each form's contribution to the success or the failure of a sentence (we used the MEGAM package (Daumé III, 2004)).

As done for our algorithm, we do not rank forms directly according to the suspicion rate S_f computed by these algorithms. Instead, we use the M_f measure presented above ($M_f = S_f \cdot \ln|\mathcal{O}_f|$). Using directly van Noord's measure selects as most suspicious words very rare words, which shows the importance of a good balance between suspicion rate and frequency (as noted by (van Noord, 2004) in the discussion of his results). This remark applies to the maximum entropy measure as well.

Table 4 shows for all algorithms the 10 best-ranked suspicious forms, complemented by a manual evaluation of their relevance. One clearly sees that our approach leads to the best results. Van Noord's technique has been initially designed to find errors in resources that already ensured a very high coverage. On our systems, whose development is less advanced, this technique ranks as most suspicious forms those which are simply the most frequent ones. It seems to be the case for the standard maximum entropy algorithm, thus showing the importance to take into account the fact that there is at least one cause of error in any sentence whose parsing failed, not only to identify a main suspicious form in each sentence, but also to get relevant global results.

4.4 Comparing results for both parsers

We complemented the separated study of error mining results on the output of both parsers by an analysis of merged results. We computed for each form the harmonic mean of both measures $M_f = S_f \cdot \ln|\mathcal{O}_f|$ obtained for each parsing system. Results (not shown here) are very interesting, because they identify errors that come mostly from resources that are shared by both systems (the Le*fff* lexicon and the pre-syntactic processing chain SXPipe). Although some errors come from common lacks of coverage in both grammars, it is nevertheless a very efficient mean to get a first repartition between error sources.

4.5 Introducing form bigrams

As said before, we also performed experiments where not only forms but also form bigrams are treated as potential causes of errors. This approach allows to identify situations where a form is not in itself a relevant cause of error, but leads often to a parse failure when immediately followed or preceded by an other form.

Table 5 shows best-ranked form bigrams (forms that are ranked in-between are not shown, to em-

#occ	$> 100\,000$	$> 10\,000$	> 1000	> 100	> 10
#forms	13	84	947	8345	40 393
#suspicious forms (%)	1 (7.6%)	13 (15.5%)	177 (18.7%)	1919 (23%)	12 022 (29.8%)

Table 2: Suspicious forms repartition for MD/FRMG

Rank	Token(s)/form	$S_f^{(50)}$	$\|\mathcal{O}_f\|$	$\mathrm{err}(f)$	M_f	Error cause
1	_____/_UNDERSCORE	100%	6399	100%	8.76	corpus: typographic noise
2	(...)	46%	2168	67%	2.82	SxPipe: should be treated as skippable words
3	2_]/_NUMBER	76%	30	93%	2.58	SxPipe: bad treatment of list constructs
4	privées	39%	589	87%	2.53	Le*fff*: misses as an adjective
5	Haaretz/_Uw	51%	149	70%	2.53	SxPipe: needs local grammars for references
6	contesté	52%	122	90%	2.52	Le*fff*: misses as an adjective
7	occupés	38%	601	86%	2.42	Le*fff*: misses as an adjective
8	privée	35%	834	82%	2.38	Le*fff*: misses as an adjective
9	[...]	44%	193	71%	2.33	SxPipe: should be treated as skippable words
10	faudrait	36%	603	85%	2.32	Le*fff*: can have a nominal object

Table 3: Analysis of the 10 best-ranked forms (ranked according to $M_f = S_f \cdot \ln |\mathcal{O}_f|$)

	this paper		global		maxent	
Rank	Token(s)/form	Eval	Token(s)/form	Eval	Token(s)/form	Eval
1	_____/_UNDERSCORE	++	*	+	pour	-
2	(...)	++	,	-	)	-
3	2_]/_NUMBER	++	livre	-	à	-
4	privées	++	.	-	qu’il/qu’	-
5	Haaretz/_Uw	++	de	-	sont	-
6	contesté	++	;	-	le	-
7	occupés	++	:	-	qu’un/qu’	+
8	privée	++	la	-	qu’un/un	+
9	[...]	++	étrangères	-	que	-
10	faudrait	++	lecteurs	-	pourrait	-

Table 4: The 10 best-ranked suspicious forms, according the the M_f measure, as computed by different algorithms: ours (*this paper*), a standard maximum entropy algorithm (*maxent*) and van Noord’s rate $\mathrm{err}(f)$ (*global*).

Rank	Tokens and forms	M_f	Error cause
4	Toutes/toutes les	2.73	grammar: badly treated pre-determiner adjective
6	y en	2,34	grammar: problem with the construction *il y en a…*
7	in “	1.81	Le*fff*: *in* misses as a preposition, which happends before book titles (hence the “)
10	donne à	1.44	Le*fff*: *donner* should sub-categorize à-vcomps (*donner à voir…*)
11	de demain	1.19	Le*fff*: *demain* misses as common noun (standard adv are not preceded by prep)
16	(22/_NUMBER	0.86	grammar: footnote references not treated
16	22/_NUMBER)	0.86	as above

Table 5: Best ranked form bigrams (forms ranked inbetween are not shown; ranked according to $M_f = S_f \cdot \ln |\mathcal{O}_f|$). *These results have been computed on a subset of the MD corpus (60,000 sentences).*

phasize bigram results), with the same data as in table 3.

5 Conclusions and perspectives

As we have shown, parsing large corpora allows to set up error mining techniques, so as to identify missing and erroneous information in the different resources that are used by full-featured parsing systems. The technique described in this paper and its implementation on forms and form bigrams has already allowed us to detect many errors and omissions in the Le*fff* lexicon, to point out inappropriate behaviors of the SXPipe pre-syntactic processing chain, and to reveal the lack of coverage of the grammars for certain phenomena.

We intend to carry on and extend this work. First of all, the visualization environment can be enhanced, as is the case for the implementation of the algorithm itself.

We would also like to integrate to the model the possibility that facts taken into account (today, forms and form bigrams) are not necessarily certain, because some of them could be the consequence of an ambiguity. For example, for a given form, several lemmas are often possible. The probabilization of these lemmas would thus allow to look for most suspicious lemmas.

We are already working on a module that will allow not only to detect errors, for example in the lexicon, but also to propose a correction. To achieve this, we want to parse anew all non-parsable sentences, after having replaced their main suspects by a special form that receives under-specified lexical information. These information can be either very general, or can be computed by appropriate generalization patterns applied on the information associated by the lexicon with the original form. A statistical study of the new parsing results will make it possible to propose corrections concerning the involved forms.

References

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximun entropy approach to natural language processing. *Computational Linguistics*, 22(1):pp. 39–71.

Pierre Boullier and Benoît Sagot. 2005a. Analyse syntaxique profonde à grande échelle: SXLFG. *Traitement Automatique des Langues (T.A.L.)*, 46(2).

Pierre Boullier and Benoît Sagot. 2005b. Efficient and robust LFG parsing: SxLfg. In *Proceedings of IWPT'05*, Vancouver, Canada, October.

Hal Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://www.isi.edu/~hdaume/docs/daume04cg-bfgs.ps`, implementation available at `http://www.isi.edu/~hdaume/megam/`.

Patrick Paroubek, Louis-Gabriel Pouillot, Isabelle Robba, and Anne Vilnat. 2005. EASy : campagne d'évaluation des analyseurs syntaxiques. In *Proceedings of the EASy workshop of TALN 2005*, Dourdan, France.

Benoît Sagot and Pierre Boullier. 2005. From raw corpus to word lattices: robust pre-parsing processing. In *Proceedings of L&TC 2005*, Poznań, Pologne.

Benoît Sagot, Lionel Clément, Éric Villemonte de la Clergerie, and Pierre Boullier. 2005. Vers un méta-lexique pour le français : architecture, acquisition, utilisation. Journée d'étude de l'ATALA sur l'interface lexique-grammaire et les lexiques syntaxiques et sémantiques, March.

François Thomasset and Éric Villemonte de la Clergerie. 2005. Comment obtenir plus des méta-grammaires. In *Proceedings of TALN'05*, Dourdan, France, June. ATALA.

Gertjan van Noord. 2004. Error mining for wide-coverage grammar engineering. In *Proc. of ACL 2004*, Barcelona, Spain.

Éric Villemonte de la Clergerie. 2005. DyALog: a tabular logic programming based environment for NLP. In *Proceedings of 2nd International Workshop on Constraint Solving and Language Processing (CSLP'05)*, Barcelona, Spain, October.

Reranking and Self-Training for Parser Adaptation

David McClosky, Eugene Charniak, and Mark Johnson
Brown Laboratory for Linguistic Information Processing (BLLIP)
Brown University
Providence, RI 02912
{dmcc|ec|mj}@cs.brown.edu

Abstract

Statistical parsers trained and tested on the Penn Wall Street Journal (WSJ) treebank have shown vast improvements over the last 10 years. Much of this improvement, however, is based upon an ever-increasing number of features to be trained on (typically) the WSJ treebank data. This has led to concern that such parsers may be too finely tuned to this corpus at the expense of portability to other genres. Such worries have merit. The standard "Charniak parser" checks in at a labeled precision-recall f-measure of 89.7% on the Penn WSJ test set, but only 82.9% on the test set from the Brown treebank corpus.

This paper should allay these fears. In particular, we show that the reranking parser described in Charniak and Johnson (2005) improves performance of the parser on Brown to 85.2%. Furthermore, use of the self-training techniques described in (McClosky et al., 2006) raise this to 87.8% (an error reduction of 28%) again without any use of labeled Brown data. This is remarkable since training the parser and reranker on labeled Brown data achieves only 88.4%.

1 Introduction

Modern statistical parsers require treebanks to train their parameters, but their performance declines when one parses genres more distant from the training data's domain. Furthermore, the treebanks required to train said parsers are expensive and difficult to produce.

Naturally, one of the goals of statistical parsing is to produce a broad-coverage parser which is relatively insensitive to textual domain. But the lack of corpora has led to a situation where much of the current work on parsing is performed on a single domain using training data from that domain — the Wall Street Journal (WSJ) section of the Penn Treebank (Marcus et al., 1993). Given the aforementioned costs, it is unlikely that many significant treebanks will be created for new genres. Thus, *parser adaptation* attempts to leverage existing labeled data from one domain and create a parser capable of parsing a different domain.

Unfortunately, the state of the art in parser portability (i.e. using a parser trained on one domain to parse a different domain) is not good. The "Charniak parser" has a labeled precision-recall f-measure of 89.7% on WSJ but a lowly 82.9% on the test set from the Brown corpus treebank. Furthermore, the treebanked Brown data is mostly general non-fiction and much closer to WSJ than, e.g., medical corpora would be. Thus, most work on parser adaptation resorts to using some labeled in-domain data to fortify the larger quantity of out-of-domain data.

In this paper, we present some encouraging results on parser adaptation without any in-domain data. (Though we also present results with in-domain data as a reference point.) In particular we note the effects of two comparatively recent techniques for parser improvement.

The first of these, *parse-reranking* (Collins, 2000; Charniak and Johnson, 2005) starts with a "standard" generative parser, but uses it to generate the n-best parses rather than a single parse. Then a reranking phase uses more detailed features, features which would (mostly) be impossible to incorporate in the initial phase, to reorder

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 337–344,
Sydney, July 2006. ©2006 Association for Computational Linguistics

the list and pick a possibly different best parse. At first blush one might think that gathering even more fine-grained features from a WSJ treebank would not help adaptation. However, we find that reranking improves the parsers performance from 82.9% to 85.2%.

The second technique is *self-training* — parsing unlabeled data and adding it to the training corpus. Recent work, (McClosky et al., 2006), has shown that adding many millions of words of machine parsed and reranked LA Times articles does, in fact, improve performance of the parser on the closely related WSJ data. Here we show that it also helps the father-afield Brown data. Adding it improves performance yet-again, this time from 85.2% to 87.8%, for a net error reduction of 28%. It is interesting to compare this to our results for a completely Brown trained system (i.e. one in which the first-phase parser is trained on just Brown training data, and the second-phase reranker is trained on Brown 50-best lists). This system performs at a 88.4% level — only slightly higher than that achieved by our system with only WSJ data.

2 Related Work

Work in parser adaptation is premised on the assumption that one wants a single parser that can handle a wide variety of domains. While this is the goal of the majority of parsing researchers, it is not quite universal. Sekine (1997) observes that for parsing a specific domain, data from that domain is most beneficial, followed by data from the same class, data from a different class, and data from a different domain. He also notes that different domains have very different structures by looking at frequent grammar productions. For these reasons he takes the position that we should, instead, simply create treebanks for a large number of domains. While this is a coherent position, it is far from the majority view.

There are many different approaches to parser adaptation. Steedman et al. (2003) apply co-training to parser adaptation and find that co-training can work across domains. The need to parse biomedical literature inspires (Clegg and Shepherd, 2005; Lease and Charniak, 2005). Clegg and Shepherd (2005) provide an extensive side-by-side performance analysis of several modern statistical parsers when faced with such data. They find that techniques which combine different parsers such as voting schemes and parse selection can improve performance on biomedical data. Lease and Charniak (2005) use the Charniak parser for biomedical data and find that the use of out-of-domain trees and in-domain vocabulary information can considerably improve performance.

Training	Testing	f-measure	
		Gildea	Bacchiani
WSJ	WSJ	86.4	87.0
WSJ	Brown	80.6	81.1
Brown	Brown	84.0	84.7
WSJ+Brown	Brown	84.3	85.6

Table 1: Gildea and Bacchiani results on WSJ and Brown test corpora using different WSJ and Brown training sets. Gildea evaluates on sentences of length ≤ 40, Bacchiani on all sentences.

However, the work which is most directly comparable to ours is that of (Ratnaparkhi, 1999; Hwa, 1999; Gildea, 2001; Bacchiani et al., 2006). All of these papers look at what happens to modern WSJ-trained statistical parsers (Ratnaparkhi's, Collins', Gildea's and Roark's, respectively) as training data varies in size or usefulness (because we are testing on something other than WSJ). We concentrate particularly on the work of (Gildea, 2001; Bacchiani et al., 2006) as they provide results which are directly comparable to those presented in this paper.

Looking at Table 1, the first line shows us the standard training and testing on WSJ — both parsers perform in the 86-87% range. The next line shows what happens when parsing Brown using a WSJ-trained parser. As with the Charniak parser, both parsers take an approximately 6% hit.

It is at this point that our work deviates from these two papers. Lacking alternatives, both (Gildea, 2001) and (Bacchiani et al., 2006) give up on adapting a pure WSJ trained system, instead looking at the issue of how much of an improvement one gets over a pure Brown system by adding WSJ data (as seen in the last two lines of Table 1). Both systems use a "model-merging" (Bacchiani et al., 2006) approach. The different corpora are, in effect, concatenated together. However, (Bacchiani et al., 2006) achieve a larger gain by weighting the in-domain (Brown) data more heavily than the out-of-domain WSJ data. One can imagine, for instance, five copies of the Brown data concatenated with just one copy of WSJ data.

3 Corpora

We primarily use three corpora in this paper. Self-training requires labeled and unlabeled data. We assume that these sets of data must be in similar domains (e.g. news articles) though the effectiveness of self-training across domains is currently an open question. Thus, we have labeled (WSJ) and unlabeled (NANC) out-of-domain data and labeled in-domain data (BROWN). Unfortunately, lacking a corresponding corpus to NANC for BROWN, we cannot perform the opposite scenario and adapt BROWN to WSJ.

3.1 Brown

The BROWN corpus (Francis and Kučera, 1979) consists of many different genres of text, intended to approximate a "balanced" corpus. While the full corpus consists of fiction and nonfiction domains, the sections that have been annotated in Treebank II bracketing are primarily those containing fiction. Examples of the sections annotated include science fiction, humor, romance, mystery, adventure, and "popular lore." We use the same divisions as Bacchiani et al. (2006), who base their divisions on Gildea (2001). Each division of the corpus consists of sentences from all available genres. The training division consists of approximately 80% of the data, while held-out development and testing divisions each make up 10% of the data. The treebanked sections contain approximately 25,000 sentences (458,000 words).

3.2 Wall Street Journal

Our out-of-domain data is the Wall Street Journal (WSJ) portion of the Penn Treebank (Marcus et al., 1993) which consists of about 40,000 sentences (one million words) annotated with syntactic information. We use the standard divisions: Sections 2 through 21 are used for training, section 24 for held-out development, and section 23 for final testing.

3.3 North American News Corpus

In addition to labeled news data, we make use of a large quantity of unlabeled news data. The unlabeled data is the North American News Corpus, NANC (Graff, 1995), which is approximately 24 million unlabeled sentences from various news sources. NANC contains no syntactic information and sentence boundaries are induced by a simple discriminative model. We also perform some basic cleanups on NANC to ease parsing. NANC contains news articles from various news sources including the Wall Street Journal, though for this paper, we only use articles from the LA Times portion.

To use the data from NANC, we use *self-training* (McClosky et al., 2006). First, we take a WSJ trained reranking parser (i.e. both the parser and reranker are built from WSJ training data) and parse the sentences from NANC with the 50-best (Charniak and Johnson, 2005) parser. Next, the 50-best parses are reordered by the reranker. Finally, the 1-best parses after reranking are combined with the WSJ training set to retrain the first-stage parser.[1] McClosky et al. (2006) find that the self-trained models help considerably when parsing WSJ.

4 Experiments

We use the Charniak and Johnson (2005) reranking parser in our experiments. Unless mentioned otherwise, we use the WSJ-trained reranker (as opposed to a BROWN-trained reranker). To evaluate, we report bracketing f-scores.[2] Parser f-scores reported are for sentences up to 100 words long, while reranking parser f-scores are over all sentences. For simplicity and ease of comparison, most of our evaluations are performed on the development section of BROWN.

4.1 Adapting self-training

Our first experiment examines the performance of the self-trained parsers. While the parsers are created entirely from labeled WSJ data and unlabeled NANC data, they perform extremely well on BROWN development (Table 2). The trends are the same as in (McClosky et al., 2006): Adding NANC data improves parsing performance on BROWN development considerably, improving the f-score from 83.9% to 86.4%. As more NANC data is added, the f-score appears to approach an asymptote. The NANC data appears to help reduce data sparsity and fill in some of the gaps in the WSJ model. Additionally, the reranker provides further benefit and adds an absolute 1-2% to the f-score. The improvements appear to be orthogonal, as our best performance is reached when we use the reranker and add 2,500k self-trained sentences from NANC.

[1] We trained a new reranker from this data as well, but it does not seem to get significantly different performance.

[2] The harmonic mean of labeled precision (P) and labeled recall (R), i.e. $f = \frac{2 \times P \times R}{P + R}$

Sentences added	Parser	Reranking Parser
Baseline BROWN	86.4	87.4
Baseline WSJ	83.9	85.8
WSJ+50k	84.8	86.6
WSJ+250k	85.7	87.2
WSJ+500k	86.0	87.3
WSJ+750k	86.1	87.5
WSJ+1,000k	86.2	87.3
WSJ+1,500k	86.2	87.6
WSJ+2,000k	86.1	87.7
WSJ+2,500k	86.4	87.7

Table 2: Effects of adding NANC sentences to WSJ training data on parsing performance. f-scores for the parser with and without the WSJ reranker are shown when evaluating on BROWN development. For this experiment, we use the WSJ-trained reranker.

The results are even more surprising when we compare against a parser[3] trained on the labeled training section of the BROWN corpus, with parameters tuned against its held-out section. Despite seeing no in-domain data, the WSJ based parser is able to match the BROWN based parser.

For the remainder of this paper, we will refer to the model trained on WSJ+2,500k sentences of NANC as our "best WSJ+NANC" model. We also note that this "best" parser is different from the "best" parser for parsing WSJ, which was trained on WSJ with a relative weight[4] of 5 and 1,750k sentences from NANC. For parsing BROWN, the difference between these two parsers is not large, though.

Increasing the relative weight of WSJ sentences versus NANC sentences when testing on BROWN development does not appear to have a significant effect. While (McClosky et al., 2006) showed that this technique was effective when testing on WSJ, the true distribution was closer to WSJ so it made sense to emphasize it.

4.2 Incorporating In-Domain Data

Up to this point, we have only considered the situation where we have no in-domain data. We now explore different ways of making use of labeled and unlabeled in-domain data.

Bacchiani et al. (2006) applies self-training to parser adaptation to utilize unlabeled in-domain data. The authors find that it helps quite a bit when adapting from BROWN to WSJ. They use a parser trained from the BROWN train set to parse WSJ and add the parsed WSJ sentences to their training set. We perform a similar experiment, using our WSJ-trained reranking parser to parse BROWN train and testing on BROWN development. We achieved a boost from 84.8% to 85.6% when we added the parsed BROWN sentences to our training. Adding in 1,000k sentences from NANC as well, we saw a further increase to 86.3%. However, the technique does not seem as effective in our case. While the self-trained BROWN data helps the parser, it adversely affects the performance of the reranking parser. When self-trained BROWN data is added to WSJ training, the reranking parser's performance drops from 86.6% to 86.1%. We see a similar degradation as NANC data is added to the training set as well. We are not yet able to explain this unusual behavior.

We now turn to the scenario where we have some labeled in-domain data. The most obvious way to incorporate labeled in-domain data is to combine it with the labeled out-of-domain data. We have already seen the results (Gildea, 2001) and (Bacchiani et al., 2006) achieve in Table 1.

We explore various combinations of BROWN, WSJ, and NANC corpora. Because we are mainly interested in exploring techniques with self-trained models rather than optimizing performance, we only consider weighting each corpus with a relative weight of one for this paper. The models generated are tuned on section 24 from WSJ. The results are summarized in Table 3.

While both WSJ and BROWN models benefit from a small amount of NANC data, adding more than 250k NANC sentences to the BROWN or combined models causes their performance to drop. This is not surprising, though, since adding "too much" NANC overwhelms the more accurate BROWN or WSJ counts. By weighting the counts from each corpus appropriately, this problem can be avoided.

Another way to incorporate labeled data is to tune the parser back-off parameters on it. Bacchiani et al. (2006) report that tuning on held-out BROWN data gives a large improvement over tun-

[3] In this case, only the parser is trained on BROWN. In section 4.3, we compare against a fully BROWN-trained reranking parser as well.

[4] A relative weight of n is equivalent to using n copies of the corpus, i.e. an event that occurred x times in the corpus would occur $x \times n$ times in the weighted corpus. Thus, larger corpora will tend to dominate smaller corpora of the same relative weight in terms of event counts.

ing on WSJ data. The improvement is mostly (but not entirely) in precision. We do not see the same improvement (Figure 1) but this is likely due to differences in the parsers. However, we do see a similar improvement for parsing accuracy once NANC data has been added. The reranking parser generally sees an improvement, but it does not appear to be significant.

4.3 Reranker Portability

We have shown that the WSJ-trained reranker is actually quite portable to the BROWN fiction domain. This is surprising given the large number of features (over a million in the case of the WSJ reranker) tuned to adjust for errors made in the 50-best lists by the first-stage parser. It would seem the corrections memorized by the reranker are not as domain-specific as we might expect.

As further evidence, we present the results of applying the WSJ model to the Switchboard corpus — a domain much less similar to WSJ than BROWN. In Table 4, we see that while the parser's performance is low, self-training and reranking provide orthogonal benefits. The improvements represent a 12% error reduction with no additional in-domain data. Naturally, in-domain data and speech-specific handling (e.g. disfluency modeling) would probably help dramatically as well.

Finally, to compare against a model fully trained on BROWN data, we created a BROWN reranker. We parsed the BROWN training set with 20-fold cross-validation, selected features that occurred 5 times or more in the training set, and fed the 50-best lists from the parser to a numerical optimizer to estimate feature weights. The resulting reranker model had approximately 700,000 features, which is about half as many as the WSJ trained reranker. This may be due to the smaller size of the BROWN training set or because the feature schemas for the reranker were developed on WSJ data. As seen in Table 5, the BROWN reranker is not a significant improvement over the WSJ reranker for parsing BROWN data.

5 Analysis

We perform several types of analysis to measure some of the differences and similarities between the BROWN-trained and WSJ-trained reranking parsers. While the two parsers agree on a large number of parse brackets (Section 5.2), there are categorical differences between them (as seen in Section 5.3).

Parser model	Parser f-score	Reranker f-score
WSJ	74.0	75.9
WSJ+NANC	75.6	77.0

Table 4: Parser and reranking parser performance on the SWITCHBOARD development corpus. In this case, WSJ+NANC is a model created from WSJ and 1,750k sentences from NANC.

Model	1-best	10-best	25-best	50-best
WSJ	82.6	88.9	90.7	91.9
WSJ+NANC	86.4	92.1	93.5	94.3
BROWN	86.3	92.0	93.3	94.2

Table 6: Oracle f-scores of top n parses produced by baseline WSJ parser, a combined WSJ and NANC parser, and a baseline BROWN parser.

5.1 Oracle Scores

Table 6 shows the f-scores of an "oracle reranker" — i.e. one which would always choose the parse with the highest f-score in the n-best list. While the WSJ parser has relatively low f-scores, adding NANC data results in a parser with comparable oracle scores as the parser trained from BROWN training. Thus, the WSJ+NANC model has better oracle rates than the WSJ model (McClosky et al., 2006) for both the WSJ and BROWN domains.

5.2 Parser Agreement

In this section, we compare the output of the WSJ+NANC-trained and BROWN-trained reranking parsers. We use *evalb* to calculate how similar the two sets of output are on a bracket level. Table 7 shows various statistics. The two parsers achieved an 88.0% f-score between them. Additionally, the two parsers agreed on all brackets almost half the time. The part of speech tagging agreement is fairly high as well. Considering they were created from different corpora, this seems like a high level of agreement.

5.3 Statistical Analysis

We conducted randomization tests for the significance of the difference in corpus f-score, based on the randomization version of the paired sample t-test described by Cohen (1995). The null hypothesis is that the two parsers being compared are in fact behaving identically, so permuting or swapping the parse trees produced by the parsers for

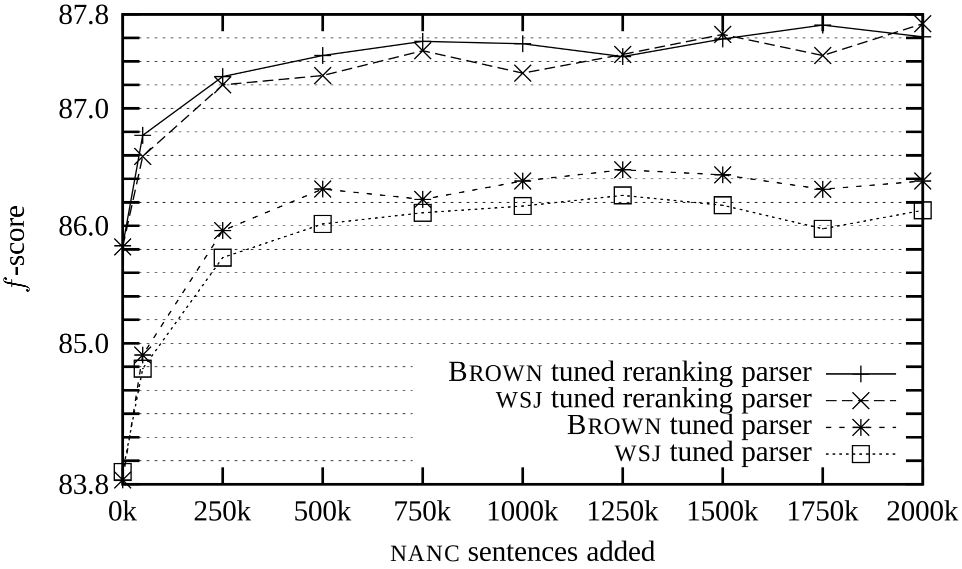


Figure 1: Precision and recall *f*-scores when testing on BROWN development as a function of the number of NANC sentences added under four test conditions. "BROWN tuned" indicates that BROWN training data was used to tune the parameters (since the normal held-out section was being used for testing). For "WSJ tuned," we tuned the parameters from section 24 of WSJ. Tuning on BROWN helps the parser, but not for the reranking parser.

Parser model	Parser alone	Reranking parser
WSJ alone	83.9	85.8
WSJ+2,500k NANC	86.4	87.7
BROWN alone	86.3	87.4
BROWN+50k NANC	86.8	88.0
BROWN+250k NANC	86.8	88.1
BROWN+500k NANC	86.7	87.8
WSJ+BROWN	86.5	88.1
WSJ+BROWN+50k NANC	86.8	88.1
WSJ+BROWN+250k NANC	86.8	88.1
WSJ+BROWN+500k NANC	86.6	87.7

Table 3: *f*-scores from various combinations of WSJ, NANC, and BROWN corpora on BROWN development. The reranking parser used the WSJ-trained reranker model. The BROWN parsing model is naturally better than the WSJ model for this task, but combining the two training corpora results in a better model (as in Gildea (2001)). Adding small amounts of NANC further improves the models.

Parser model	Parser alone	WSJ-reranker	BROWN-reranker
WSJ	82.9	85.2	85.2
WSJ+NANC	87.1	87.8	87.9
BROWN	86.7	88.2	88.4

Table 5: Performance of various combinations of parser and reranker models when evaluated on BROWN test. The WSJ+NANC parser with the WSJ reranker comes close to the BROWN-trained reranking parser. The BROWN reranker provides only a small improvement over its WSJ counterpart, which is not statistically significant.

Bracketing agreement f-score	88.03%
Complete match	44.92%
Average crossing brackets	0.94
POS Tagging agreement	94.85%

Table 7: Agreement between the WSJ+NANC parser with the WSJ reranker and the BROWN parser with the BROWN reranker. Complete match is how often the two reranking parsers returned the exact same parse.

Feature	Estimate	z-value	$\Pr(>\|z\|)$	
(Intercept)	0.054	0.3	0.77	
IN	-0.134	-4.4	8.4e-06	***
ID=G	0.584	2.5	0.011	*
ID=K	0.697	2.9	0.003	**
ID=L	0.552	2.3	0.021	*
ID=M	0.376	0.9	0.33	
ID=N	0.642	2.7	0.0055	**
ID=P	0.624	2.7	0.0069	**
ID=R	0.040	0.1	0.90	

Table 9: The logistic model of BROWN/BROWN f-score $>$ WSJ+NANC/WSJ f-score identified by model selection. The feature IN is the number prepositions in the sentence, while ID identifies the Brown subcorpus that the sentence comes from. Stars indicate significance level.

the same test sentence should not affect the corpus f-scores. By estimating the proportion of permutations that result in an absolute difference in corpus f-scores at least as great as that observed in the actual output, we obtain a distribution-free estimate of significance that is robust against parser and evaluator failures. The results of this test are shown in Table 8. The table shows that the BROWN reranker is not significantly different from the WSJ reranker.

In order to better understand the difference between the reranking parser trained on Brown and the WSJ+NANC/WSJ reranking parser (a reranking parser with the first-stage trained on WSJ+NANC and the second-stage trained on WSJ) on Brown data, we constructed a logistic regression model of the difference between the two parsers' f-scores on the development data using the R statistical package[5]. Of the 2,078 sentences in the development data, 29 sentences were discarded because *evalb* failed to evaluate at least one of the parses.[6] A Wilcoxon signed rank test on the remaining 2,049 paired sentence level f-scores was significant at $p = 0.0003$. Of these 2,049 sentences, there were 983 parse pairs with the same sentence-level f-score. Of the 1,066 sentences for which the parsers produced parses with different f-scores, there were 580 sentences for which the BROWN/BROWN parser produced a parse with a higher sentence-level f-score and 486 sentences for which the WSJ+NANC/WSJ parser produce a parse with a higher f-score. We constructed a generalized linear model with a binomial link with BROWN/BROWN f-score $>$ WSJ+NANC/WSJ f-score as the predicted variable, and sentence length, the number of prepositions (IN), the number of conjunctions (CC) and Brown subcorpus ID as explanatory variables. Model selection (using the "step" procedure) discarded all but the IN and Brown ID explanatory variables. The final estimated model is shown in Table 9. It shows that the WSJ+NANC/WSJ parser becomes more likely to have a higher f-score than the BROWN/BROWN parser as the number of prepositions in the sentence increases, and that the BROWN/BROWN parser is more likely to have a higher f-score on Brown sections K, N, P, G and L (these are the general fiction, adventure and western fiction, romance and love story, letters and memories, and mystery sections of the Brown corpus, respectively). The three sections of BROWN not in this list are F, M, and R (popular lore, science fiction, and humor).

6 Conclusions and Future Work

We have demonstrated that rerankers and self-trained models can work well across domains. Models self-trained on WSJ appear to be better parsing models in general, the benefits of which are not limited to the WSJ domain. The WSJ-trained reranker using out-of-domain LA Times parses (produced by the WSJ-trained reranker) achieves a labeled precision-recall f-measure of 87.8% on Brown data, nearly equal to the performance one achieves by using a purely Brown trained parser-reranker. The 87.8% f-score on Brown represents a 24% error reduction on the corpus.

Of course, as corpora differences go, Brown is relatively close to WSJ. While we also find that our

[5] http://www.r-project.org

[6] This occurs when an apostrophe is analyzed as a possessive marker in the gold tree and a punctuation symbol in the parse tree, or vice versa.

	WSJ+NANC/WSJ	BROWN/WSJ	BROWN/BROWN
WSJ/WSJ	0.025 (0)	0.030 (0)	0.031 (0)
WSJ+NANC/WSJ		0.004 (0.1)	0.006 (0.025)
BROWN/WSJ			0.002 (0.27)

Table 8: The difference in corpus f-score between the various reranking parsers, and the significance of the difference in parentheses as estimated by a randomization test with 10^6 samples. "x/y" indicates that the first-stage parser was trained on data set x and the second-stage reranker was trained on data set y.

"best" WSJ-parser-reranker improves performance on the Switchboard corpus, it starts from a much lower base (74.0%), and achieves a much less significant improvement (3% absolute, 11% error reduction). Bridging these larger gaps is still for the future.

One intriguing idea is what we call "self-trained bridging-corpora." We have not yet experimented with medical text but we expect that the "best" WSJ+NANC parser will not perform very well. However, suppose one does self-training on a biology textbook instead of the LA Times. One might hope that such a text will split the difference between more "normal" newspaper articles and the specialized medical text. Thus, a self-trained parser based upon such text might do much better than our standard "best." This is, of course, highly speculative.

Acknowledgments

This work was supported by NSF grants LIS9720368, and IIS0095940, and DARPA GALE contract HR0011-06-2-0001. We would like to thank the BLLIP team for their comments.

References

Michiel Bacchiani, Michael Riley, Brian Roark, and Richard Sproat. 2006. MAP adaptation of stochastic grammars. *Computer Speech and Language*, 20(1):41–68.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *Proc. of the 2005 Meeting of the Assoc. for Computational Linguistics (ACL)*, pages 173–180.

Andrew B. Clegg and Adrian Shepherd. 2005. Evaluating and integrating treebank parsers on a biomedical corpus. In *Proceedings of the ACL Workshop on Software*.

Paul R. Cohen. 1995. *Empirical Methods for Artificial Intelligence*. The MIT Press, Cambridge, Massachusetts.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML 2000)*, pages 175–182, Stanford, California.

W. Nelson Francis and Henry Kučera. 1979. *Manual of Information to accompany a Standard Corpus of Present-day Edited American English*, for use with Digital Computers. Brown University, Providence, Rhode Island.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 167–202.

David Graff. 1995. *North American News Text Corpus*. Linguistic Data Consortium. LDC95T21.

Rebecca Hwa. 1999. Supervised grammar induction using training data with limited constituent information. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 72–80, University of Maryland.

Matthew Lease and Eugene Charniak. 2005. Parsing biomedical literature. In *Second International Joint Conference on Natural Language Processing (IJCNLP'05)*.

Michell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Comp. Linguistics*, 19(2):313–330.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of HLT-NAACL 2006*.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175.

Satoshi Sekine. 1997. The domain dependence of parsing. In *Proc. Applied Natural Language Processing (ANLP)*, pages 96–102.

Mark Steedman, Miles Osborne, Anoop Sarkar, Stephen Clark, Rebecca Hwa, Julia Hockenmaier, Paul Ruhlen, Steven Baker, and Jeremiah Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proc. of European ACL (EACL)*, pages 331–338.

Automatic Classification of Verbs in Biomedical Texts

Anna Korhonen
University of Cambridge
Computer Laboratory
15 JJ Thomson Avenue
Cambridge CB3 0GD, UK
alk23@cl.cam.ac.uk

Yuval Krymolowski
Dept. of Computer Science
Technion
Haifa 32000
Israel
yuvalkr@cs.technion.ac.il

Nigel Collier
National Institute of Informatics
Hitotsubashi 2-1-2
Chiyoda-ku, Tokyo 101-8430
Japan
collier@nii.ac.jp

Abstract

Lexical classes, when tailored to the application and domain in question, can provide an effective means to deal with a number of natural language processing (NLP) tasks. While manual construction of such classes is difficult, recent research shows that it is possible to automatically induce verb classes from cross-domain corpora with promising accuracy. We report a novel experiment where similar technology is applied to the important, challenging domain of biomedicine. We show that the resulting classification, acquired from a corpus of biomedical journal articles, is highly accurate and strongly domain-specific. It can be used to aid BIO-NLP directly or as useful material for investigating the syntax and semantics of verbs in biomedical texts.

1 Introduction

Lexical classes which capture the close relation between the syntax and semantics of verbs have attracted considerable interest in NLP (Jackendoff, 1990; Levin, 1993; Dorr, 1997; Prescher et al., 2000). Such classes are useful for their ability to capture generalizations about a range of linguistic properties. For example, verbs which share the meaning of 'manner of motion' (such as *travel*, *run*, *walk*), behave similarly also in terms of subcategorization (*I traveled/ran/walked*, *I traveled/ran/walked to London*, *I traveled/ran/walked five miles*). Although the correspondence between the syntax and semantics of words is not perfect and the classes do not provide means for full semantic inferencing, their predictive power is nevertheless considerable.

NLP systems can benefit from lexical classes in many ways. Such classes define the mapping from surface realization of arguments to predicate-argument structure, and are therefore an important component of any system which needs the latter. As the classes can capture higher level abstractions they can be used as a means to abstract away from individual words when required. They are also helpful in many operational contexts where lexical information must be acquired from small application-specific corpora. Their predictive power can help compensate for lack of data fully exemplifying the behavior of relevant words.

Lexical verb classes have been used to support various (multilingual) tasks, such as computational lexicography, language generation, machine translation, word sense disambiguation, semantic role labeling, and subcategorization acquisition (Dorr, 1997; Prescher et al., 2000; Korhonen, 2002). However, large-scale exploitation of the classes in real-world or domain-sensitive tasks has not been possible because the existing classifications, e.g. (Levin, 1993), are incomprehensive and unsuitable for specific domains.

While manual classification of large numbers of words has proved difficult and time-consuming, recent research shows that it is possible to automatically induce lexical classes from corpus data with promising accuracy (Merlo and Stevenson, 2001; Brew and Schulte im Walde, 2002; Korhonen et al., 2003). A number of ML methods have been applied to classify words using features pertaining to mainly syntactic structure (e.g. statistical distributions of subcategorization frames (SCFs) or general patterns of syntactic behaviour, e.g. transitivity, passivisability) which have been extracted from corpora using e.g. part-of-speech tagging or robust statistical parsing techniques.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 345–352,
Sydney, July 2006.

This research has been encouraging but it has so far concentrated on general language. Domain-specific lexical classification remains unexplored, although it is arguably important: existing classifications are unsuitable for domain-specific applications and these often challenging applications might benefit from improved performance by utilizing lexical classes the most.

In this paper, we extend an existing approach to lexical classification (Korhonen et al., 2003) and apply it (without any domain specific tuning) to the domain of biomedicine. We focus on biomedicine for several reasons: (i) NLP is critically needed to assist the processing, mining and extraction of knowledge from the rapidly growing literature in this area, (ii) the domain lexical resources (e.g. UMLS metathesaurus and lexicon[1]) do not provide sufficient information about verbs and (iii) being linguistically challenging, the domain provides a good test case for examining the potential of automatic classification.

We report an experiment where a classification is induced for 192 relatively frequent verbs from a corpus of 2230 biomedical journal articles. The results, evaluated with domain experts, show that the approach is capable of acquiring classes with accuracy higher than that reported in previous work on general language. We discuss reasons for this and show that the resulting classes differ substantially from those in extant lexical resources. They constitute the first syntactic-semantic verb classification for the biomedical domain and could be readily applied to support BIO-NLP.

We discuss the domain-specific issues related to our task in section 2. The approach to automatic classification is presented in section 3. Details of the experimental evaluation are supplied in section 4. Section 5 provides discussion and section 6 concludes with directions for future work.

2 The Biomedical Domain and Our Task

Recent years have seen a massive growth in the scientific literature in the domain of biomedicine. For example, the MEDLINE database[2] which currently contains around 16M references to journal articles, expands with 0.5M new references each year. Because future research in the biomedical sciences depends on making use of all this existing knowledge, there is a strong need for the development of NLP tools which can be used to automatically locate, organize and manage facts related to published experimental results.

In recent years, major progress has been made on information retrieval and on the extraction of specific relations e.g. between proteins and cell types from biomedical texts (Hirschman et al., 2002). Other tasks, such as the extraction of factual information, remain a bigger challenge. This is partly due to the challenging nature of biomedical texts. They are complex both in terms of syntax and semantics, containing complex nominals, modal subordination, anaphoric links, etc.

Researchers have recently began to use deeper NLP techniques (e.g. statistical parsing) in the domain because they are not challenged by the complex structures to the same extent than shallow techniques (e.g. regular expression patterns) are (Lease and Charniak, 2005). However, deeper techniques require richer domain-specific lexical information for optimal performance than is provided by existing lexicons (e.g. UMLS). This is particularly important for verbs, which are central to the structure and meaning of sentences.

Where the lexical information is absent, lexical classes can compensate for it or aid in obtaining it in the ways described in section 1. Consider e.g. the INDICATE and ACTIVATE verb classes in Figure 1. They capture the fact that their members are similar in terms of syntax and semantics: they have similar SCFs and selectional preferences, and they can be used to make similar statements which describe similar events. Such information can be used to build a richer lexicon capable of supporting key tasks such as parsing, predicate-argument identification, event extraction and the identification of biomedical (e.g. interaction) relations.

While an abundance of work has been conducted on semantic classification of biomedical terms and nouns, less work has been done on the (manual or automatic) semantic classification of verbs in the biomedical domain (Friedman et al., 2002; Hatzivassiloglou and Weng, 2002; Spasic et al., 2005). No previous work exists in this domain on the type of *lexical* (i.e. syntactic-semantic) verb classification this paper focuses on.

To get an initial idea about the differences between our target classification and a general language classification, we examined the extent to which individual verbs and their frequencies differ in biomedical and general language texts. We

[1] http://www.nlm.nih.gov/research/umls

[2] http://www.ncbi.nlm.nih.gov/PubMed/

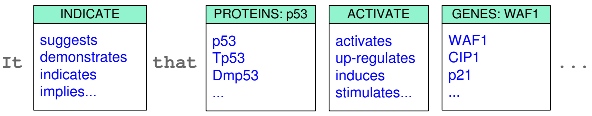


Figure 1: Sample lexical classes

BIO	BNC
show	*do*
suggest	*say*
use	*make*
indicate	*go*
contain	*see*
describe	*take*
express	*get*
bind	*know*
require	*come*
observe	*give*
find	*think*
determine	*use*
demonstrate	*find*
perform	*look*
induce	*want*

Table 1: The 15 most frequent verbs in the biomedical data and in the BNC

created a corpus of 2230 biomedical journal articles (see section 4.1 for details) and compared the distribution of verbs in this corpus with that in the British National Corpus (BNC) (Leech, 1992). We calculated the Spearman rank correlation between the 1165 verbs which occurred in both corpora. The result was only a weak correlation: 0.37 ± 0.03. When the scope was restricted to the 100 most frequent verbs in the biomedical data, the correlation was 0.12 ± 0.10 which is only 1.2σ away from zero. The dissimilarity between the distributions is further indicated by the Kullback-Leibler distance of 0.97. Table 1 illustrates some of these big differences by showing the list of 15 most frequent verbs in the two corpora.

3 Approach

We extended the system of Korhonen et al. (2003) with additional clustering techniques (introduced in sections 3.2.2 and 3.2.4) and used it to obtain the classification for the biomedical domain. The system (i) extracts features from corpus data and (ii) clusters them using five different methods. These steps are described in the following two sections, respectively.

3.1 Feature Extraction

We employ as features distributions of SCFs specific to given verbs. We extract them from corpus data using the comprehensive subcategorization acquisition system of Briscoe and Carroll (1997) (Korhonen, 2002). The system incorporates RASP, a domain-independent robust statistical parser (Briscoe and Carroll, 2002), which tags, lemmatizes and parses data yielding complete though shallow parses and a SCF classifier which incorporates an extensive inventory of 163 verbal SCFs[3]. The SCFs abstract over specific lexically-governed particles and prepositions and specific predicate selectional preferences. In our work, we parameterized two high frequency SCFs for prepositions (PP and NP + PP SCFs). No filtering of potentially noisy SCFs was done to provide clustering with as much information as possible.

3.2 Classification

The SCF frequency distributions constitute the input data to automatic classification. We experiment with five clustering methods: the simple hard nearest neighbours method and four probabilistic methods – two variants of Probabilistic Latent Semantic Analysis and two information theoretic methods (the Information Bottleneck and the Information Distortion).

3.2.1 Nearest Neighbours

The first method collects the nearest neighbours (NN) of each verb. It (i) calculates the Jensen-Shannon divergence (JS) between the SCF distributions of each pair of verbs, (ii) connects each verb with the most similar other verb, and finally (iii) finds all the connected components. The NN method is very simple. It outputs only one clustering configuration and therefore does not allow examining different cluster granularities.

3.2.2 Probabilistic Latent Semantic Analysis

The Probabilistic Latent Semantic Analysis (PLSA, Hoffman (2001)) assumes a generative model for the data, defined by selecting (i) a verb $verb_i$, (ii) a semantic class $class_k$ from the distribution $p(Classes \,|\, verb_i)$, and (iii) a SCF scf_j from the distribution $p(\text{SCF}s \,|\, class_k)$. PLSA uses Expectation Maximization (EM) to find the distribution $\tilde{p}(\text{SCF}s \,|\, Clusters, Verbs)$ which maximises the likelihood of the observed counts. It does this by minimising the cost function

$$\mathcal{F} = -\beta \log \text{Likelihood}(\tilde{p} \,|\, \text{data}) + H(\tilde{p}) \,.$$

[3] See http://www.cl.cam.ac.uk/users/alk23/subcat/subcat.html for further detail.

For $\beta = 1$ minimising $\mathcal{F}$ is equivalent to the standard EM procedure while for $\beta < 1$ the distribution $\tilde{p}$ tends to be more evenly spread. We use $\beta = 1$ (PLSA/EM) and $\beta = 0.75$ (PLSA$_{\beta=0.75}$). We currently "harden" the output and assign each verb to the most probable cluster only[4].

3.2.3 Information Bottleneck

The Information Bottleneck (Tishby et al., 1999) (IB) is an information-theoretic method which controls the balance between: (i) the *loss* of information by representing verbs as clusters ($I(Clusters; Verbs)$), which has to be minimal, and (ii) the *relevance* of the output clusters for representing the SCF distribution ($I(Clusters; \text{SCF}s)$) which has to be maximal. The balance between these two quantities ensures optimal compression of data through clusters. The trade-off between the two constraints is realized through minimising the cost function:

$$\begin{aligned}\mathcal{L}_{\text{IB}} = {} & I(Clusters; Verbs) \\ & - \beta I(Clusters; \text{SCF}s)\,,\end{aligned}$$

where β is a parameter that balances the constraints. IB takes three inputs: (i) SCF-verb distributions, (ii) the desired number of clusters $\mathcal{K}$, and (iii) the initial value of β. It then looks for the minimal β that decreases $\mathcal{L}_{\text{IB}}$ compared to its value with the initial β, using the given $\mathcal{K}$. IB delivers as output the probabilities $p(K|V)$. It gives an indication for the most informative number of output configurations: the ones for which the relevance information increases more sharply between $\mathcal{K} - 1$ and $\mathcal{K}$ clusters than between $\mathcal{K}$ and $\mathcal{K} + 1$.

3.2.4 Information Distortion

The Information Distortion method (Dimitrov and Miller, 2001) (ID) is otherwise similar to IB but $\mathcal{L}_{\text{ID}}$ differs from $\mathcal{L}_{\text{IB}}$ by an additional term that adds a bias towards clusters of similar size:

$$\begin{aligned}\mathcal{L}_{\text{ID}} &= -H(Clusters \,|\, Verbs) \\ &\quad - \beta I(Clusters; \text{SCF}s) \\ &= \mathcal{L}_{\text{IB}} - H(Clusters)\,.\end{aligned}$$

ID yields more evenly divided clusters than IB.

4 Experimental Evaluation

4.1 Data

We downloaded the data for our experiment from the MEDLINE database, from three of the 10 leading journals in biomedicine: 1) *Genes & Development* (molecular biology, molecular genetics), 2) *Journal of Biological Chemistry* (biochemistry and molecular biology) and 3) *Journal of Cell Biology* (cellular structure and function). 2230 full-text articles from years 2003-2004 were used. The data included 11.5M words and 323,307 sentences in total. 192 medium to high frequency verbs (with the minimum of 300 occurrences in the data) were selected for experimentation[5]. This test set was big enough to produce a useful classification but small enough to enable thorough evaluation in this first attempt to classify verbs in the biomedical domain.

4.2 Processing the Data

The data was first processed using the feature extraction module. 233 (preposition-specific) SCF types appeared in the resulting lexicon, 36 per verb on average.[6] The classification module was then applied. NN produced $\mathcal{K}_{\text{nn}} = 42$ clusters. From the other methods we requested $\mathcal{K} = 2$ to 60 clusters. We chose for evaluation the outputs corresponding to the most informative values of $\mathcal{K}$: 20, 33, 53 for IB, and 17, 33, 53 for ID.

4.3 Gold Standard

Because no target lexical classification was available for the biomedical domain, human experts (4 domain experts and 2 linguists) were used to create the gold standard. They were asked to examine whether the test verbs similar in terms of their syntactic properties (i.e. verbs with similar SCF distributions) are similar also in terms of semantics (i.e. they share a common meaning). Where this was the case, a verb class was identified and named.

The domain experts examined the 116 verbs whose analysis required domain knowledge (e.g. *activate, solubilize, harvest*), while the linguists analysed the remaining 76 general or scientific text verbs (e.g. *demonstrate, hypothesize, appear*). The linguists used Levin (1993) classes as gold standard classes whenever possible and created novel ones when needed. The domain experts used two purely semantic classifications of biomedical verbs (Friedman et al., 2002; Spasic et al., 2005)[7] as a starting point where this was pos-

[4] The same approach was used with the information theoretic methods. It made sense in this initial work on biomedical classification. In the future we could use soft clustering a means to investigate polysemy.

[5] 230 verbs were employed initially but 38 were dropped later so that each (coarse-grained) class would have the minimum of 2 members in the gold standard.

[6] This number is high because no filtering of potentially noisy SCFs was done.

[7] See http://www.cbr-masterclass.org.

1 Have an effect on activity (BIO/29)	8 Physical Relation
1.1 Activate / Inactivate	Between Molecules (BIO/20)
1.1.1 Change activity: *activate, inhibit*	**8.1 Binding:** *bind, attach*
1.1.2 Suppress: *suppress, repress*	**8.2 Translocate and Segregate**
1.1.3 Stimulate: *stimulate*	8.2.1 Translocate: *shift, switch*
1.1.4 Inactivate: *delay, diminish*	8.2.2 Segregate: *segregate, export*
1.2 Affect	**8.3 Transmit**
1.2.1 Modulate: *stabilize, modulate*	8.3.1 Transport: *deliver, transmit*
1.2.2 Regulate: *control, support*	8.3.2 Link: *connect, map*
1.3 Increase / decrease: *increase, decrease*	9 Report (GEN/30)
1.4 Modify: *modify, catalyze*	**9.1 Investigate**
2 Biochemical events (BIO/12)	9.1.1 Examine: *evaluate, analyze*
2.1 Express: *express, overexpress*	9.1.2 Establish: *test, investigate*
2.2 Modification	9.1.3 Confirm: *verify, determine*
2.2.1 Biochemical modification:	**9.2 Suggest**
dephosphorylate, phosphorylate	9.2.1 Presentational:
2.2.2 Cleave: *cleave*	*hypothesize, conclude*
2.3 Interact: *react, interfere*	9.2.2 Cognitive:
3 Removal (BIO/6)	*consider, believe*
3.1 Omit: *displace, deplete*	**9.3 Indicate:** *demonstrate, imply*
3.2 Subtract: *draw, dissect*	10 Perform (GEN/10)
4 Experimental Procedures (BIO/30)	**10.1 Quantify**
4.1 Prepare	10.1.1 Quantitate: *quantify, measure*
4.1.1 Wash: *wash, rinse*	10.1.2 Calculate: *calculate, record*
4.1.2 Mix: *mix*	10.1.3 Conduct: *perform, conduct*
4.1.3 Label: *stain, immunoblot*	**10.2 Score:** *score, count*
4.1.4 Incubate: *preincubate, incubate*	11 Release (BIO/4): *detach, dissociate*
4.1.5 Elute: *elute*	12 Use (GEN/4): *utilize, employ*
4.2 Precipitate: *coprecipitate*	13 Include (GEN/11)
coimmunoprecipitate	**13.1 Encompass:** *encompass, span*
4.3 Solubilize: *solubilize,lyse*	**13.2 Include:** *contain, carry*
4.4 Dissolve: *homogenize, dissolve*	14 Call (GEN/3): *name, designate*
4.5 Place: *load, mount*	15 Move (GEN/12)
5 Process (BIO/5): *linearize, overlap*	**15.1 Proceed:**
6 Transfect (BIO/4): *inject, microinject*	*progress, proceed*
7 Collect (BIO/6)	**15.2 Emerge:**
7.1 Collect: *harvest, select*	*arise, emerge*
7.2 Process: *centrifuge, recover*	16 Appear (GEN/6): *appear, occur*

Table 2: The gold standard classification with a few example verbs per class

sible (i.e. where they included our test verbs and also captured their relevant senses)[8].

The experts created a 3-level gold standard which includes both broad and finer-grained classes. Only those classes / memberships were included which all the experts (in the two teams) agreed on.[9] The resulting gold standard including 16, 34 and 50 classes is illustrated in table 2 with 1-2 example verbs per class. The table indicates which classes were created by domain experts (BIO) and which by linguists (GEN). Each class was associated with 1-30 member verbs[10]. The total number of verbs is indicated in the table (e.g. 10 for PERFORM class).

4.4 Measures

The clusters were evaluated against the gold standard using measures which are applicable to all the classification methods and which deliver a numerical value easy to interpret.

The first measure, the *adjusted pairwise precision*, evaluates clusters in terms of verb pairs:

$$\text{APP} = \frac{1}{\mathcal{K}} \sum_{i=1}^{\mathcal{K}} \frac{\text{num. of correct pairs in } k_i}{\text{num. of pairs in } k_i} \cdot \frac{|k_i|-1}{|k_i|+1}$$

APP is the average proportion of all within-cluster pairs that are correctly co-assigned. Multiplied by a factor that increases with cluster size it compensates for a bias towards small clusters.

The second measure is *modified purity*, 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 $n_{\text{prevalent}}(K)$. Verbs that do not take it are considered as errors. Clusters where $n_{\text{prevalent}}(K) = 1$ are disregarded as not to introduce a bias towards singletons:

$$m\text{PUR} = \frac{\sum\limits_{n_{\text{prevalent}}(k_i) \geq 2} n_{\text{prevalent}}(k_i)}{\text{number of verbs}}$$

The third measure is the *weighted class accuracy*, the proportion of members of dominant clusters DOM-CLUST$_i$ within all classes c_i.

$$\text{ACC} = \frac{\sum_{i=1}^{\mathcal{C}} \text{verbs in DOM-CLUST}_i}{\text{number of verbs}}$$

mPUR can be seen to measure the precision of clusters and ACC the recall. We define an F measure as the harmonic mean of mPUR and ACC:

$$F = \frac{2 \cdot m\text{PUR} \cdot \text{ACC}}{m\text{PUR} + \text{ACC}}$$

The statistical significance of the results is measured by randomisation tests where verbs are swapped between the clusters and the resulting clusters are evaluated. The swapping is repeated 100 times for each output and the average av_{swaps} and the standard deviation σ_{swaps} is measured. The significance is the scaled difference $signif = (result - av_{\text{swaps}})/\sigma_{\text{swaps}}$.

4.5 Results from Quantitative Evaluation

Table 3 shows the performance of the five clustering methods for $\mathcal{K} = 42$ clusters (as produced by the NN method) at the 3 levels of gold standard classification. Although the two PLSA variants (particularly PLSA$_{\beta=0.75}$) produce a fairly accurate coarse grained classification, they perform worse than all the other methods at the finer-grained levels of gold standard, particularly according to the global measures. Being based on

[8] Purely semantic classes tend to be finer-grained than lexical classes and not necessarily syntactic in nature. Only these two classifications were found to be similar enough to our target classification to provide a useful starting point. Section 5 includes a summary of the similarities/differences between our gold standard and these other classifications.

[9] Experts were allowed to discuss the problematic cases to obtain maximal accuracy - hence no inter-annotator agreement is reported.

[10] The minimum of 2 member verbs were required at the coarser-grained levels of 16 and 34 classes.

	16 Classes				34 Classes				50 Classes			
	APP	mPUR	ACC	F	APP	mPUR	ACC	F	APP	mPUR	ACC	F
NN	81	86	39	53	64	74	62	67	54	67	73	69
IB	74	88	47	61	61	76	74	75	55	69	87	76
ID	79	89	37	52	63	78	65	70	53	70	77	73
PLSA/EM	55	72	49	58	43	53	61	57	35	47	66	55
PLSA$_{\beta=0.75}$	65	71	68	70	53	48	76	58	41	34	77	47

Table 3: The performance of the NN, PLSA, IB and ID methods with $\mathcal{K}_{\text{nn}} = 42$ clusters

		16 Classes				34 Classes				50 Classes			
$\mathcal{K}$		APP	mPUR	ACC	F	APP	mPUR	ACC	F	APP	mPUR	ACC	F
20	IB	**74**	**77**	**66**	**71**	60	56	86	67	54	48	93	63
17	ID	**67**	**76**	**60**	**67**	43	56	81	66	34	46	91	61
33	IB	78	87	52	65	**69**	**75**	**81**	**77**	61	67	93	77
	ID	81	88	43	57	**65**	**75**	**70**	**72**	54	67	82	73
53	IB	71	87	41	55	61	78	66	71	**54**	**72**	**79**	**75**
	ID	79	89	33	48	66	79	55	64	**53**	**72**	**68**	**69**

Table 4: The performance of IB and ID for the 3 levels of class hierarchy for informative values of $\mathcal{K}$

pairwise similarities, NN shows mostly better performance than IB and ID on the pairwise measure APP but the global measures are better for IB and ID. The differences are smaller in mPUR (yet significant: 2σ between NN and IB and 3σ between NN and ID) but more notable in ACC (which is e.g. $8 - 12\%$ better for IB than for NN). Also the F results suggest that the two information theoretic methods are better overall than the simple NN method.

IB and ID also have the advantage (over NN) that they can be used to produce a hierarchical verb classification. Table 4 shows the results for IB and ID for the informative values of $\mathcal{K}$. The bold font indicates the results when the match between the values of $\mathcal{K}$ and the number of classes at the particular level of the gold standard is the closest.

IB is clearly better than ID at all levels of gold standard. It yields its best results at the medium level (34 classes) with $\mathcal{K} = 33$: $F = 77$ and APP = 69 (the results for ID are $F = 72$ and APP = 65). At the most fine-grained level (50 classes), IB is equally good according to F with $\mathcal{K} = 33$, but APP is 8% lower. Although ID is occasionally better than IB according to APP and mPUR (see e.g. the results for 16 classes with $\mathcal{K} = 53$) this never happens in the case where the correspondence between the number of gold standard classes and the values of $\mathcal{K}$ is the closest. In other words, the informative values of $\mathcal{K}$ prove really informative for IB. The lower performance of ID seems to be due to its tendency to create evenly sized clusters.

All the methods perform significantly better than our random baseline. The significance of the results with respect to two swaps was at the 2σ level, corresponding to a 97% confidence that the results are above random.

4.6 Qualitative Evaluation

We performed further, qualitative analysis of clusters produced by the best performing method IB. Consider the following clusters:

A: *inject, transfect, microinfect, contransfect* (6)
B: *harvest, select, collect* (7.1)
centrifuge, process, recover (7.2)
C: *wash, rinse* (4.1.1)
immunoblot (4.1.3)
overlap (5)
D: *activate* (1.1.1)

When looking at coarse-grained outputs, interestingly, $\mathcal{K}$ as low as 8 learned the broad distinction between biomedical and general language verbs (the two verb types appeared only rarely in the same clusters) and produced large semantically meaningful groups of classes (e.g. the coarse-grained classes EXPERIMENTAL PROCEDURES, TRANSFECT and COLLECT were mapped together). $\mathcal{K} = 12$ was sufficient to identify several classes with very particular syntax One of them was TRANSFECT (see **A** above) whose members were distinguished easily because of their typical SCFs (e.g. *inject /transfect/microinfect/contransfect* X *with/into* Y).

On the other hand, even $\mathcal{K} = 53$ could not identify classes with very similar (yet un-identical) syntax. These included many semantically similar sub-classes (e.g. the two sub-classes of COLLECT shown in **B** whose members take similar NP and PP SCFs). However, also a few semantically different verbs clustered wrongly because of this reason, such as the ones exemplified in **C**. In **C**, *immunoblot* (from the LABEL class) is still somewhat related to *wash* and *rinse* (the WASH class) because they all belong to the larger EXPERIMENTAL PROCEDURES class, but *overlap* (from the PROCESS class) shows up in the cluster merely because of syntactic idiosyncracy.

While parser errors caused by the challenging biomedical texts were visible in some SCFs (e.g. looking at a sample of SCFs, some adjunct instances were listed in the argument slots of the frames), the cases where this resulted in incorrect classification were not numerous[11].

One representative singleton resulting from these errors is exemplified in **D**. *Activate* appears in relatively complicated sentence structures, which gives rise to incorrect SCFs. For example, *MECs cultured on 2D planar substrates transiently* **activate** *MAP kinase in response to EGF, whereas...* gets incorrectly analysed as SCF NP-NP, while *The effect of the constitutively* **activated** *ARF6-Q67L mutant was investigated...* receives the incorrect SCF analysis NP-SCOMP. Most parser errors are caused by unknown domain-specific words and phrases.

5 Discussion

Due to differences in the task and experimental setup, direct comparison of our results with previously published ones is impossible. The closest possible comparison point is (Korhonen et al., 2003) which reported 50-59% mPUR and 15-19% APP on using IB to assign 110 polysemous (general language) verbs into 34 classes. Our results are substantially better, although we made no effort to restrict our scope to monosemous verbs[12] and although we focussed on a linguistically challenging domain.

It seems that our better result is largely due to the higher uniformity of verb senses in the biomedical domain. We could not investigate this effect systematically because no manually sense annotated data (or a comprehensive list of verb senses) exists for the domain. However, examination of a number of corpus instances suggests that the use of verbs is fairly conventionalized in our data[13]. Where verbs show less sense variation, they show less SCF variation, which aids the discovery of verb classes. Korhonen et al. (2003) observed the opposite with general language data.

We examined, class by class, to what extent our domain-specific gold standard differs from the related general (Levin, 1993) and domain classifications (Spasic et al., 2005; Friedman et al., 2002) (recall that the latter were purely semantic classifications as no lexical ones were available for biomedicine):

33 (of the 50) classes in the gold standard are biomedical. Only 6 of these correspond (fully or mostly) to the semantic classes in the domain classifications. 17 are unrelated to any of the classes in Levin (1993) while 16 bear vague resemblance to them (e.g. our TRANSPORT verbs are also listed under Levin's SEND verbs) but are too different (semantically and syntactically) to be combined.

17 (of the 50) classes are general (scientific) classes. 4 of these are absent in Levin (e.g. QUANTITATE). 13 are included in Levin, but 8 of them have a more restricted sense (and fewer members) than the corresponding Levin class. Only the remaining 5 classes are identical (in terms of members and their properties) to Levin classes.

These results highlight the importance of building or tuning lexical resources specific to different domains, and demonstrate the usefulness of automatic lexical acquisition for this work.

6 Conclusion

This paper has shown that current domain-independent NLP and ML technology can be used to automatically induce a relatively high accuracy verb classification from a linguistically challenging corpus of biomedical texts. The lexical classification resulting from our work is strongly domain-specific (it differs substantially from previous ones) and it can be readily used to aid BIO-NLP. It can provide useful material for investigating the syntax and semantics of verbs in biomedical data or for supplementing existing domain lexical resources with additional information (e.g.

[11] This is partly because the mistakes of the parser are somewhat consistent (similar for similar verbs) and partly because the SCFs gather data from hundreds of corpus instances, many of which are analysed correctly.

[12] Most of our test verbs are polysemous according to WordNet (WN) (Miller, 1990), but this is not a fully reliable indication because WN is not specific to this domain.

[13] The different sub-domains of the biomedical domain may, of course, be even more conventionalized (Friedman et al., 2002).

semantic classifications with additional member verbs). Lexical resources enriched with verb class information can, in turn, better benefit practical tasks such as parsing, predicate-argument identification, event extraction, identification of biomedical relation patterns, among others.

In the future, we plan to improve the accuracy of automatic classification by seeding it with domain-specific information (e.g. using named entity recognition and anaphoric linking techniques similar to those of Vlachos et al. (2006)). We also plan to conduct a bigger experiment with a larger number of verbs and demonstrate the usefulness of the bigger classification for practical BIO-NLP application tasks. In addition, we plan to apply similar technology to other interesting domains (e.g. tourism, law, astronomy). This will not only enable us to experiment with cross-domain lexical class variation but also help to determine whether automatic acquisition techniques benefit, in general, from domain-specific tuning.

Acknowledgement

We would like to thank Yoko Mizuta, Shoko Kawamato, Sven Demiya, and Parantu Shah for their help in creating the gold standard.

References

C. Brew and S. Schulte im Walde. 2002. Spectral clustering for German verbs. In *Conference on Empirical Methods in Natural Language Processing*, Philadelphia, USA.

E. J. Briscoe and J. Carroll. 1997. Automatic extraction of subcategorization from corpora. In 5th *ACL Conference on Applied Natural Language Processing*, pages 356–363, Washington DC.

E. J. Briscoe and J. Carroll. 2002. Robust accurate statistical annotation of general text. In 3rd *International Conference on Language Resources and Evaluation*, pages 1499–1504, Las Palmas, Gran Canaria.

A. G. Dimitrov and J. P. Miller. 2001. Neural coding and decoding: communication channels and quantization. *Network: Computation in Neural Systems*, 12(4):441–472.

B. Dorr. 1997. Large-scale dictionary construction for foreign language tutoring and interlingual machine translation. *Machine Translation*, 12(4):271–325.

C. Friedman, P. Kra, and A. Rzhetsky. 2002. Two biomedical sublanguages: a description based on the theories of Zellig Harris. *Journal of Biomedical Informatics*, 35(4):222–235.

V. Hatzivassiloglou and W. Weng. 2002. Learning anchor verbs for biological interaction patterns from published text articles. *International Journal of Medical Inf.*, 67:19–32.

L. Hirschman, J. C. Park, J. Tsujii, L. Wong, and C. H. Wu. 2002. Accomplishments and challenges in literature data mining for biology. *Journal of Bioinformatics*, 18(12):1553–1561.

T. Hoffman. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1):177–196.

R. Jackendoff. 1990. *Semantic Structures*. MIT Press, Cambridge, Massachusetts.

A. Korhonen, Y. Krymolowski, and Z. Marx. 2003. Clustering polysemic subcategorization frame distributions semantically. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 64–71, Sapporo, Japan.

A. Korhonen. 2002. *Subcategorization Acquisition*. Ph.D. thesis, University of Cambridge, UK.

M. Lease and E. Charniak. 2005. Parsing biomedical literature. In *Second International Joint Conference on Natural Language Processing*, pages 58–69.

G. Leech. 1992. 100 million words of English: the British National Corpus. *Language Research*, 28(1):1–13.

B. Levin. 1993. *English Verb Classes and Alternations*. Chicago University Press, Chicago.

P. Merlo and S. Stevenson. 2001. Automatic verb classification based on statistical distributions of argument structure. *Computational Linguistics*, 27(3):373–408.

G. A. Miller. 1990. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312.

D. Prescher, S. Riezler, and M. Rooth. 2000. Using a probabilistic class-based lexicon for lexical ambiguity resolution. In *18th International Conference on Computational Linguistics*, pages 649–655, Saarbrücken, Germany.

I. Spasic, S. Ananiadou, and J. Tsujii. 2005. Masterclass: A case-based reasoning system for the classification of biomedical terms. *Journal of Bioinformatics*, 21(11):2748–2758.

N. Tishby, F. C. Pereira, and W. Bialek. 1999. The information bottleneck method. In *Proc. of the* 37th *Annual Allerton Conference on Communication, Control and Computing*, pages 368–377.

A. Vlachos, C. Gasperin, I. Lewin, and E. J. Briscoe. 2006. Bootstrapping the recognition and anaphoric linking of named entitites in drosophila articles. In *Pacific Symposium in Biocomputing*.

Selection of Effective Contextual Information for Automatic Synonym Acquisition

Masato Hagiwara, Yasuhiro Ogawa, and Katsuhiko Toyama
Graduate School of Information Science,
Nagoya University
Furo-cho, Chikusa-ku, Nagoya, JAPAN 464-8603
{hagiwara, yasuhiro, toyama}@kl.i.is.nagoya-u.ac.jp

Abstract

Various methods have been proposed for automatic synonym acquisition, as synonyms are one of the most fundamental lexical knowledge. Whereas many methods are based on contextual clues of words, little attention has been paid to what kind of categories of contextual information are useful for the purpose. This study has experimentally investigated the impact of contextual information selection, by extracting three kinds of word relationships from corpora: dependency, sentence co-occurrence, and proximity. The evaluation result shows that while dependency and proximity perform relatively well by themselves, combination of two or more kinds of contextual information gives more stable performance. We've further investigated useful selection of dependency relations and modification categories, and it is found that modification has the greatest contribution, even greater than the widely adopted subject-object combination.

1 Introduction

Lexical knowledge is one of the most important resources in natural language applications, making it almost indispensable for higher levels of syntactical and semantic processing. Among many kinds of lexical relations, synonyms are especially useful ones, having broad range of applications such as query expansion technique in information retrieval and automatic thesaurus construction.

Various methods (Hindle, 1990; Lin, 1998; Hagiwara et al., 2005) have been proposed for synonym acquisition. Most of the acquisition methods are based on distributional hypothesis (Harris, 1985), which states that semantically similar words share similar contexts, and it has been experimentally shown considerably plausible.

However, whereas many methods which adopt the hypothesis are based on contextual clues concerning words, and there has been much consideration on the language models such as Latent Semantic Indexing (Deerwester et al., 1990) and Probabilistic LSI (Hofmann, 1999) and synonym acquisition method, almost no attention has been paid to what kind of categories of contextual information, or their combinations, are useful for word featuring in terms of synonym acquisition.

For example, Hindle (1990) used co-occurrences between verbs and their subjects and objects, and proposed a similarity metric based on mutual information, but no exploration concerning the effectiveness of other kinds of word relationship is provided, although it is extendable to any kinds of contextual information. Lin (1998) also proposed an information theory-based similarity metric, using a broad-coverage parser and extracting wider range of grammatical relationship including modifications, but he didn't further investigate what kind of relationships actually had important contributions to acquisition, either. The selection of useful contextual information is considered to have a critical impact on the performance of synonym acquisition. This is an independent problem from the choice of language model or acquisition method, and should therefore be examined by itself.

The purpose of this study is to experimentally investigate the impact of contextual information selection for automatic synonym acquisition. Because nouns are the main target of

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 353–360,
Sydney, July 2006. ©2006 Association for Computational Linguistics

synonym acquisition, here we limit the target of acquisition to nouns, and firstly extract the co-occurrences between nouns and three categories of contextual information — dependency, sentence co-occurrence, and proximity — from each of three different corpora, and the performance of individual categories and their combinations are evaluated. Since dependency and modification relations are considered to have greater contributions in contextual information and in the dependency category, respectively, these categories are then broken down into smaller categories to examine the individual significance.

Because the consideration on the language model and acquisition methods is not the scope of the current study, widely used vector space model (VSM), tf·idf weighting scheme, and cosine measure are adopted for similarity calculation. The result is evaluated using two automatic evaluation methods we proposed and implemented: discrimination rate and correlation coefficient based on the existing thesaurus WordNet.

This paper is organized as follows: in Section 2, three kinds of contextual information we use are described, and the following Section 3 explains the synonym acquisition method. In Section 4 the evaluation method we employed is detailed, which consists of the calculation methods of reference similarity, discrimination rate, and correlation coefficient. Section 5 provides the experimental conditions and results of contextual information selection, followed by dependency and modification selection. Section 6 concludes this paper.

2 Contextual Information

In this study, we focused on three kinds of contextual information: dependency between words, sentence co-occurrence, and proximity, that is, co-occurrence with other words in a window, details of which are provided the following sections.

2.1 Dependency

The first category of the contextual information we employed is the dependency between words in a sentence, which we suppose is most commonly used for synonym acquisition as the context of words. The dependency here includes predicate-argument structure such as subjects and objects of verbs, and modifications of nouns. As the extraction of accurate and comprehensive grammatical relations is in itself a difficult task, the sophisticated parser RASP Toolkit (Briscoe and Carroll, 2002) was utilized to extract this kind of word relations. RASP analyzes input sentences and provides wide variety of grammatical information such as POS tags, dependency structure, and parsed trees as output, among which we paid attention to dependency structure called grammatical relations (GRs) (Briscoe et al., 2002).

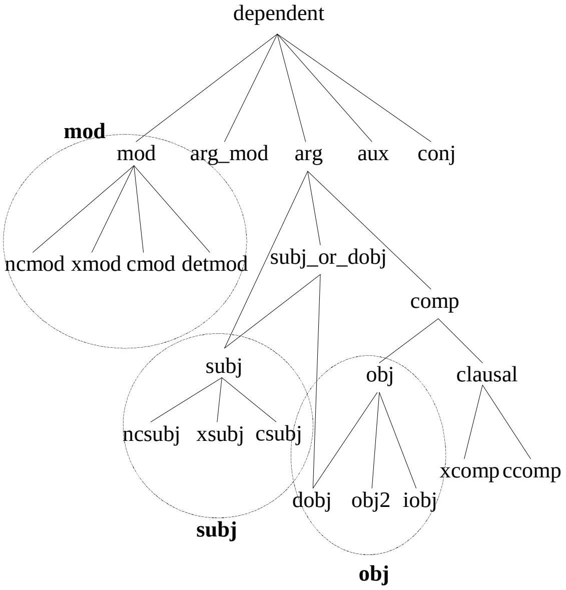


Figure 1: Hierarchy of grammatical relations and groups

GRs represent relationship among two or more words and are specified by the labels, which construct the hierarchy shown in Figure 1. In this hierarchy, the upper levels correspond to more general relations whereas the lower levels to more specific ones. Although the most general relationship in GRs is “dependent”, more specific labels are assigned whenever possible. The representation of the contextual information using GRs is as follows. Take the following sentence for example:

> Shipments have been relatively level since January, the Commerce Department noted.

RASP outputs the extracted GRs as n-ary relations as follows:

```
(ncsubj note Department obj)
(ncsubj be Shipment _)
(xcomp _ be level)
(mod _ level relatively)
(aux _ be have)
(ncmod since be January)
(mod _ Department note)
(ncmod _ Department Commerce)
```

```
(detmod _ Department the)
(ncmod _ be Department)
```

While most of GRs extracted by RASP are binary relations of head and dependent, there are some relations that contain additional slot or extra information regarding the relations, as shown "ncsubj" and "ncmod" in the above example. To obtain the final representation that we require for synonym acquisition, that is, the co-occurrence between words and their contexts, these relationships must be converted to binary relations, i.e., co-occurrence. We consider the concatenation of all the rest of the target word as context:

```
Department    ncsubj:note:*:obj
shipment      ncsubj:be:*:_
January       ncmod:since:be:*
Department    mod:_:*:note
Department    ncmod:_:*:Commerce
Commerce      ncmod:_:Department:*
Department    detmod:_:*:the
Department    ncmod:_:be:*
```

The slot for the target word is replaced by "*" in the context. Note that only the contexts for nouns are extracted because our purpose here is the automatic extraction of synonymous nouns.

2.2 Sentence Co-occurrence

As the second category of contextual information, we used the sentence co-occurrence, i.e., which sentence words appear in. Using this context is, in other words, essentially the same as featuring words with the sentences in which they occur. Treating single sentences as documents, this featuring corresponds to exploiting transposed term-document matrix in the information retrieval context, and the underlying assumption is that words that commonly appear in the similar documents or sentences are considered semantically similar.

2.3 Proximity

The third category of contextual information, proximity, utilizes tokens that appear in the vicinity of the target word in a sentence. The basic assumption here is that the more similar the distribution of proceeding and succeeding words of the target words are, the more similar meaning these two words possess, and its effectiveness has been previously shown (Macro Baroni and Sabrina Bisi, 2004). To capture the word proximity, we consider a window with a certain radius, and treat the label of the word and its position within the window as context. The contexts for the previous example sentence, when the window radius is 3, are then:

```
shipment     R1:have
shipment     R2:be
shipment     R3:relatively
January      L1:since
January      L2:level
January      L3:relatively
January      R1:,
January      R2:the
January      R3:Commerce
Commerce     L1:the
Commerce     L2:,
Commerce     L3:January
Commerce     R1:Department
...
```

Note that the proximity includes tokens such as punctuation marks as context, because we suppose they offer useful contextual information as well.

3 Synonym Acquisition Method

The purpose of the current study is to investigate the impact of the contextual information selection, not the language model itself, we employed one of the most commonly used method: vector space model (VSM) and tf·idf weighting scheme. In this framework, each word is represented as a vector in a vector space, whose dimensions correspond to contexts. The elements of the vectors given by tf·idf are the co-occurrence frequencies of words and contexts, weighted by normalized idf. That is, denoting the number of distinct words and contexts as N and M, respectively,

$$\boldsymbol{w}_i = {}^t[\text{tf}(w_i, c_1)\cdot\text{idf}(c_1)\ ...\ \text{tf}(w_i, c_M)\cdot\text{idf}(c_M)], \tag{1}$$

where $\text{tf}(w_i, c_j)$ is the co-occurrence frequency of word w_i and context c_j. $\text{idf}(c_j)$ is given by

$$\text{idf}(c_j) = \frac{\log(N/\text{df}(c_j))}{\max_k \log(N/\text{df}(v_k))}, \tag{2}$$

where $\text{df}(c_j)$ is the number of distinct words that co-occur with context c_j.

Although VSM and tf·idf are naive and simple compared to other language models like LSI and PLSI, they have been shown effective enough for the purpose (Hagiwara et al., 2005). The similarity between two words are then calculated as the cosine value of two corresponding vectors.

4 Evaluation

This section describes the evaluation methods we employed for automatic synonym acquisition. The evaluation is to measure how similar the obtained similarities are to the "true" similarities. We firstly prepared the reference similarities from the existing thesaurus WordNet as described in Section 4.1, and by comparing the reference and obtained similarities, two evaluation measures, discrimination rate and correlation coefficient, are calculated automatically as described in Sections 4.2 and 4.3.

4.1 Reference similarity calculation using WordNet

As the basis for automatic evaluation methods, the reference similarity, which is the answer value that similarity of a certain pair of words "should take," is required. We obtained the reference similarity using the calculation based on thesaurus tree structure (Nagao, 1996). This calculation method requires no other resources such as corpus, thus it is simple to implement and widely used.

The similarity between word sense w_i and word sense v_j is obtained using tree structure as follows. Let the depth[1] of node w_i be d_i, the depth of node v_j be d_j, and the maximum depth of the common ancestors of both nodes be d_{dca}. The similarity between w_i and v_j is then calculated as

$$sim(w_i, v_j) = \frac{2 \cdot d_{\mathrm{dca}}}{d_i + d_j}, \tag{3}$$

which takes the value between 0.0 and 1.0.

Figure 2 shows the example of calculating the similarity between the word senses "hill" and "coast." The number on the side of each word sense represents the word's depth. From this tree structure, the similarity is obtained:

$$sim(\text{“hill”}, \text{“coast”}) = \frac{2 \cdot 3}{5 + 5} = 0.6. \tag{4}$$

The similarity between word w with senses $w_1, ..., w_n$ and word v with senses $v_1, ..., v_m$ is defined as the maximum similarity between all the pairs of word senses:

$$sim(w, v) = \max_{i,j} sim(w_i, v_j), \tag{5}$$

whose idea came from Lin's method (Lin, 1998).

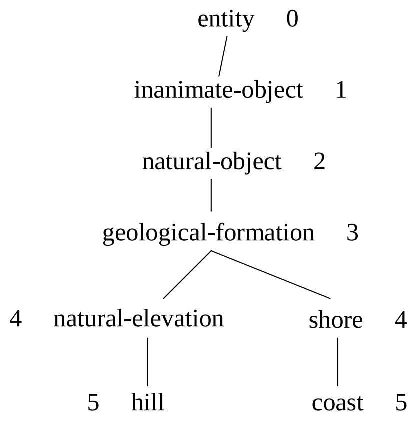


Figure 2: Example of automatic similarity calculation based on tree structure

highly related	unrelated
(answer, reply) (phone, telephone) (sign, signal) (concern, worry) ⋮	(animal, coffee) (him, technology) (track, vote) (path, youth) ⋮

Figure 3: Test-sets for discrimination rate calculation.

4.2 Discrimination Rate

The following two sections describe two evaluation measures based on the reference similarity. The first one is discrimination rate (DR). DR, originally proposed by Kojima et al. (2004), is the rate (percentage) of pairs (w_1, w_2) whose degree of association between two words w_1, w_2 is successfully discriminated by the similarity derived by the method under evaluation. Kojima et al. dealt with three-level discrimination of a pair of words, that is, highly related (synonyms or nearly synonymous), moderately related (a certain degree of association), and unrelated (irrelevant). However, we omitted the moderately related level and limited the discrimination to two-level: high or none, because of the difficulty of preparing a test set that consists of moderately related pairs.

The calculation of DR follows these steps: first, two test sets, one of which consists of highly related word pairs and the other of unrelated ones, are prepared, as shown in Figure 3. The similarity between w_1 and w_2 is then calculated for each pair (w_1, w_2) in both test sets via the method under evaluation, and the pair is labeled highly related when similarity exceeds a given threshold t and unrelated when the similarity is lower than t. The number of pairs labeled highly related in the highly related test set and unrelated in the unrelated test set are denoted n_a and n_b, respectively.

[1] To be precise, the structure of WordNet, where some word senses have more than one parent, isn't a tree but a DAG. The depth of a node is, therefore, defined here as the "maximum distance" from the root node.

DR is then given by:

$$\frac{1}{2}\left(\frac{n_a}{N_a}+\frac{n_b}{N_b}\right), \tag{6}$$

where N_a and N_b are the numbers of pairs in highly related and unrelated test sets, respectively. Since DR changes depending on threshold t, maximum value is adopted by varying t.

We used the reference similarity to create these two test sets. Firstly, $N_p = 100,000$ pairs of words are randomly created using the target vocabulary set for synonym acquisition. Proper nouns are omitted from the choice here because of their high ambiguity. The two testsets are then created extracting $n = 2,000$ most related (with high reference similarity) and unrelated (with low reference similarity) pairs.

4.3 Correlation coefficient

The second evaluation measure is correlation coefficient (CC) between the obtained similarity and the reference similarity. The higher CC value is, the more similar the obtained similarities are to WordNet, thus more accurate the synonym acquisition result is.

The value of CC is calculated as follows. Let the set of the sample pairs be P_s, the sequence of the reference similarities calculated for the pairs in P_s be $\boldsymbol{r} = (r_1, r_2, ..., r_n)$, the corresponding sequence of the target similarity to be evaluated be $\boldsymbol{r} = (s_1, s_2, ..., s_n)$, respectively. Correlation coefficient ρ is then defined by:

$$\rho = \frac{\frac{1}{n}\sum_{i=1}^{n}(r_i - \bar{r})(s_i - \bar{s})}{\sigma_r \sigma_s}, \tag{7}$$

where $\bar{r}, \bar{s}, \sigma_r$, and σ_s represent the average of $\boldsymbol{r}$ and $\boldsymbol{s}$ and the standard deviation of $\boldsymbol{r}$ and $\boldsymbol{s}$, respectively. The set of the sample pairs P_s is created in a similar way to the preparation of highly related test set used in DR calculation, except that we employed $N_p = 4,000, n = 2,000$ to avoid extreme nonuniformity.

5 Experiments

Now we desribe the experimental conditions and results of contextual information selection.

5.1 Condition

We used the following three corpora for the experiment: (1) Wall Street Journal (WSJ) corpus (approx. 68,000 sentences, 1.4 million tokens), (2) Brown Corpus (BROWN) (approx. 60,000 sentences, 1.3 million tokens), both of which are contained in Treebank 3 (Marcus, 1994), and (3) written sentences in WordBank (WB) (approx. 190,000 sentences, 3.5 million words) (Hyper-Collins, 2002). No additional annotation such as POS tags provided for Treebank was used, which means that we gave the plain texts stripped off any additional information to RASP as input.

To distinguish nouns, using POS tags annotated by RASP, any words with POS tags APP, ND, NN, NP, PN, PP were labeled as nouns. The window radius for proximity is set to 3. We also set a threshold t_f on occurrence frequency in order to filter out any words or contexts with low frequency and to reduce computational cost. More specifically, any words w such that $\sum_c \text{tf}(w,c) < t_f$ and any contexts c such that $\sum_w \text{tf}(w,c) < t_f$ were removed from the co-occurrence data. t_f was set to $t_f = 5$ for WSJ and BROWN, and $t_f = 10$ for WB in Sections 5.2 and 5.3, and $t_f = 2$ for WSJ and BROWN and $t_f = 5$ for WB in Section 5.4.

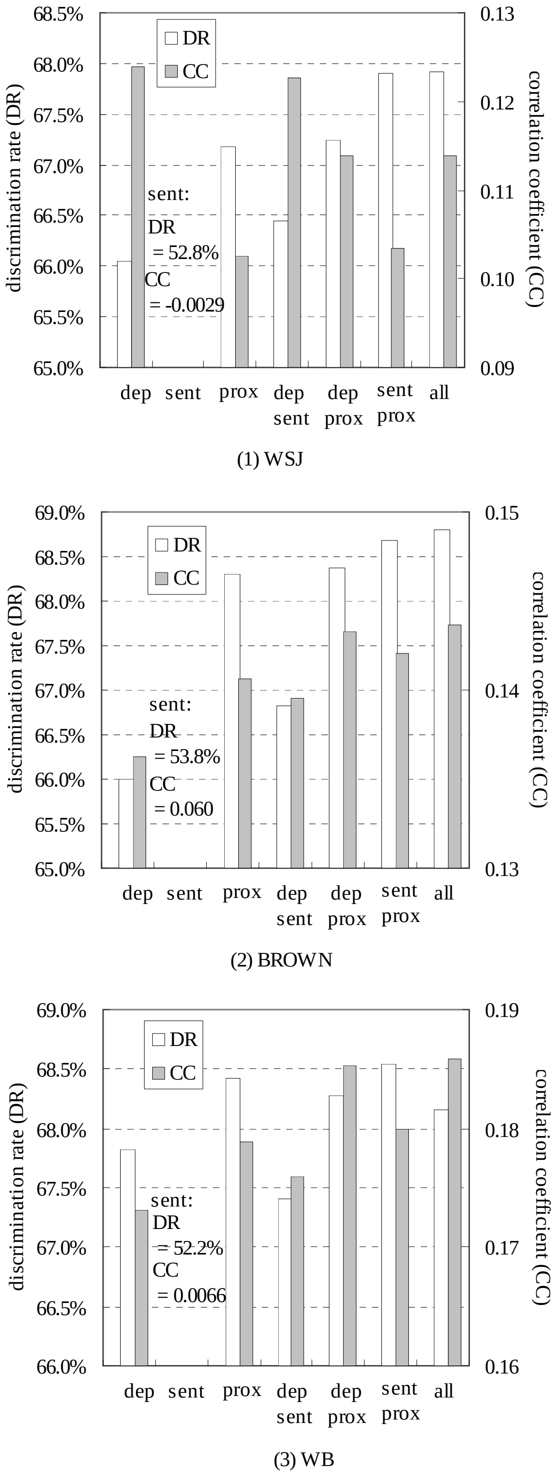
5.2 Contextual Information Selection

In this section, we experimented to discover what kind of contextual information extracted in Section 2 is useful for synonym extraction. The performances, i.e. DR and CC are evaluated for each of the three categories and their combinations.

The evaluation result for three corpora is shown in Figure 4. Notice that the range and scale of the vertical axes of the graphs vary according to corpus. The result shows that dependency and proximity perform relatively well alone, while sentence co-occurrence has almost no contributions to performance. However, when combined with other kinds of context information, every category, even sentence co-occurrence, serves to “stabilize” the overall performance, although in some cases combination itself decreases individual measures slightly. It is no surprise that the combination of all categories achieves the best performance. Therefore, in choosing combination of different kinds of context information, one should take into consideration the economical efficiency and trade-off between computational complexity and overall performance stability.

5.3 Dependency Selection

We then focused on the contribution of individual categories of dependency relation, i.e. groups of grammatical relations. The following four groups



(1) WSJ

(2) BROWN

(3) WB

Figure 4: Contextual information selection performances

Discrimination rate (DR) and correlation coefficient (CC) for (1) Wall Street Journal corpus, (2) Brown Corpus, and (3) WordBank.

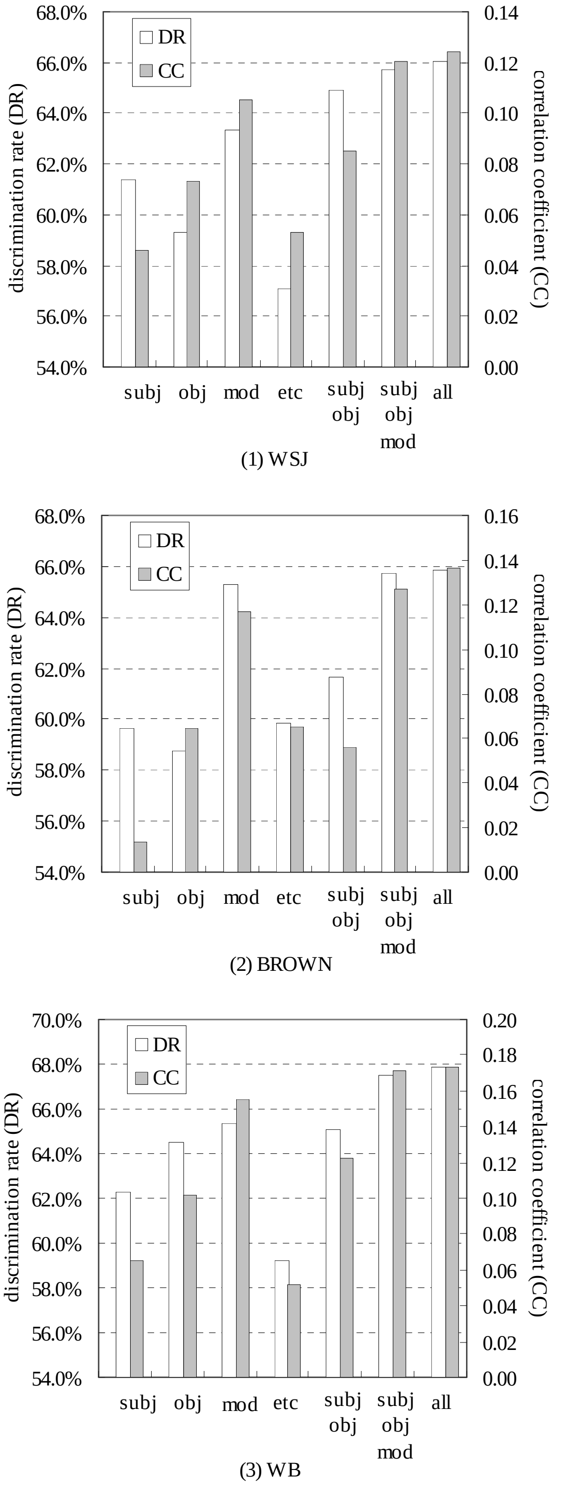
of GRs are considered for comparison convenience: (1) subj group ("subj", "ncsubj", "xsubj", and "csubj"), (2) obj group ("obj", "dobj", "obj2", and "iobj"), (3) mod group ("mod", "ncmod", "xmod", "cmod", and "detmod"), and (4) etc group (others), as shown in the circles in Figure 1. This is because distinction between relations in a group is sometimes unclear, and is considered to strongly depend on the parser implementation. The final target is seven kinds of combinations of the above four groups: subj, obj, mod, etc, subj+obj, subj+obj+mod, and all.

The two evaluation measures are similarly calculated for each group and combination, and shown in Figure 5. Although subjects, objects, and their combination are widely used contextual information, the performances for subj and obj categories, as well as their combination subj+obj, were relatively poor. On the contrary, the result clearly shows the importance of modification, which alone is even better than widely adopted subj+obj. The "stabilization effect" of combinations observed in the previous experiment is also confirmed here as well.

Because the size of the co-occurrence data varies from one category to another, we conducted another experiment to verify that the superiority of the modification category is simply due to the difference in the quality (content) of the group, not the quantity (size). We randomly extracted 100,000 pairs from each of mod and subj+obj categories to cancel out the quantity difference and compared the performance by calculating averaged DR and CC of ten trials. The result showed that, while the overall performances substantially decreased due to the size reduction, the relation between groups was preserved before and after the extraction throughout all of the three corpora, although the detailed result is not shown due to the space limitation. This means that what essentially contributes to the performance is not the size of the modification category but its content.

5.4 Modification Selection

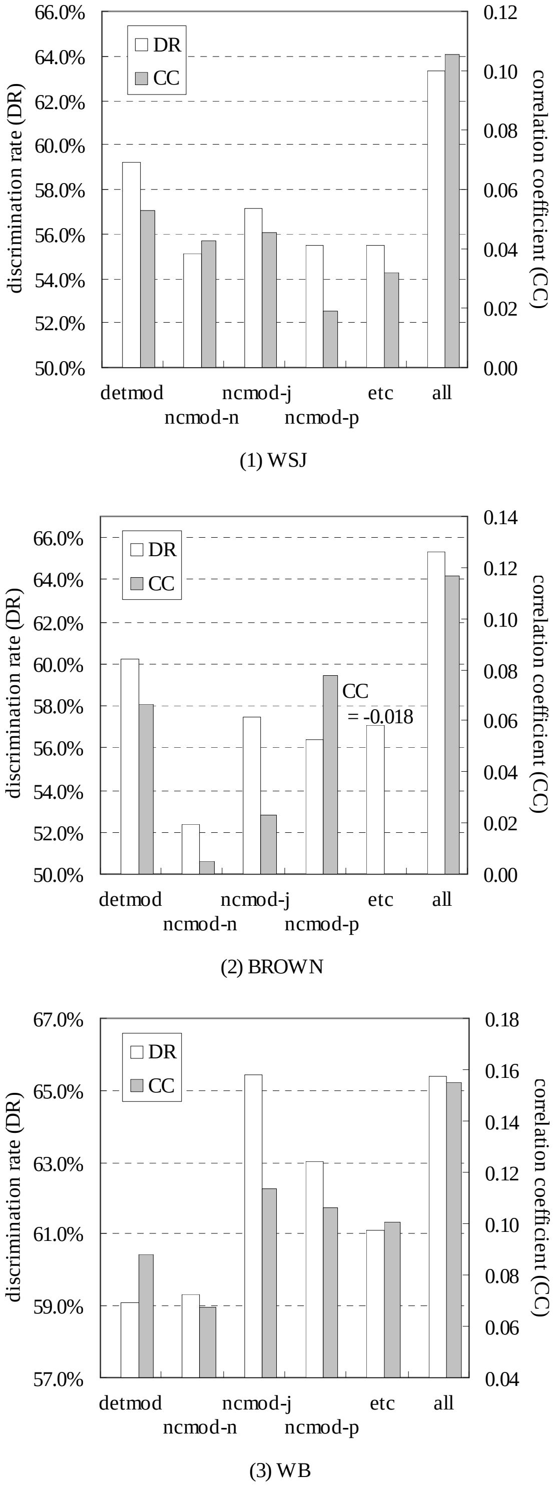
As the previous experiment shows that modifications have the biggest significance of all the dependency relationship, we further investigated what kind of modifications is useful for the purpose. To do this, we broke down the mod group into these five categories according to modifying word's category: (1) detmod, when the GR label is "det-



(1) WSJ

(2) BROWN

(3) WB



(1) WSJ

(2) BROWN

(3) WB

Figure 5: Dependency selection performances
Discrimination rate (DR) and correlation coefficient (CC) for (1) Wall Street Journal corpus, (2) Brown Corpus, and (3) WordBank.

Figure 6: Modification selection performances
Discrimination rate (DR) and correlation coefficient (CC) for (1) Wall Street Journal corpus, (2) Brown Corpus, and (3) WordBank.

mod", i.e., the modifying word is a determiner, (2) ncmod-n, when the GR label is "ncmod" and the modifying word is a noun, (3) ncmod-j, when the GR label is "ncmod" and the modifying word is an adjective or number, (4) ncmod-p, when the GR label is "ncmod" and the modification is through a preposition (e.g. "state" and "affairs" in "state of affairs"), and (5) etc (others).

The performances for each modification category are evaluated and shown in Figure 6. Although some individual modification categories such as detmod and ncmod-j outperform other categories in some cases, the overall observation is that all the modification categories contribute to synonym acquisition to some extent, and the effect of individual categories are accumulative. We therefore conclude that the main contributing factor on utilizing modification relationship in synonym acquisition isn't the type of modification, but the diversity of the relations.

6 Conclusion

In this study, we experimentally investigated the impact of contextual information selection, by extracting three kinds of contextual information — dependency, sentence co-occurrence, and proximity — from three different corpora. The acquisition result was evaluated using two evaluation measures, DR and CC using the existing thesaurus WordNet. We showed that while dependency and proximity perform relatively well by themselves, combination of two or more kinds of contextual information, even with the poorly performing sentence co-occurrence, gives more stable result. The selection should be chosen considering the trade-off between computational complexity and overall performance stability. We also showed that modification has the greatest contribution to the acquisition of all the dependency relations, even greater than the widely adopted subject-object combination. It is also shown that all the modification categories contribute to the acquisition to some extent.

Because we limited the target to nouns, the result might be specific to nouns, but the same experimental framework is applicable to any other categories of words. Although the result also shows the possibility that the bigger the corpus is, the better the performance will be, the contents and size of the corpora we used are diverse, so their relationship, including the effect of the window radius, should be examined as the future work.

References

Marco Baroni and Sabrina Bisi 2004. Using cooccurrence statistics and the web to discover synonyms in a technical language. *Proc. of the Fourth International Conference on Language Resources and Evaluation (LREC 2004).*

Ted Briscoe and John Carroll. 2002. Robust Accurate Statistical Annotation of General Text. *Proc. of the Third International Conference on Language Resources and Evaluation (LREC 2002),* 1499–1504.

Ted Briscoe, John Carroll, Jonathan Graham and Ann Copestake 2002. Relational evaluation schemes. *Proc. of the Beyond PARSEVAL Workshop at the Third International Conference on Language Resources and Evaluation*, 4–8.

Scott Deerwester, et al. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Christiane Fellbaum. 1998. *WordNet: an electronic lexical database.* MIT Press.

Masato Hagiwara, Yasuhiro Ogawa, Katsuhiko Toyama. 2005. PLSI Utilization for Automatic Thesaurus Construction. *Proc. of The Second International Joint Conference on Natural Language Processing (IJCNLP-05),* 334–345.

Zellig Harris. 1985. Distributional Structure. Jerrold J. Katz (ed.) *The Philosophy of Linguistics*. Oxford University Press. 26–47.

Donald Hindle. 1990. Noun classification from predicate-argument structures. *Proc. of the 28th Annual Meeting of the ACL*, 268–275.

Thomas Hofmann. 1999. Probabilistic Latent Semantic Indexing. *Proc. of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR '99),* 50–57.

Kazuhide Kojima, Hirokazu Watabe, and Tsukasa Kawaoka. 2004. Existence and Application of Common Threshold of the Degree of Association. *Proc. of the Forum on Information Technology (FIT2004)* F-003.

Collins. 2002. Collins Cobuild Mld Major New Edition CD-ROM. HarperCollins Publishers.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. *Proc. of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational linguistics (COLING-ACL '98),* 786–774.

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.

Makoto Nagao (ed.). 1996. *Shizengengoshori*. The Iwanami Software Science Series 15, Iwanami Shoten Publishers.

Scaling Distributional Similarity to Large Corpora

James Gorman and **James R. Curran**
School of Information Technologies
University of Sydney
NSW 2006, Australia
{jgorman2,james}@it.usyd.edu.au

Abstract

Accurately representing synonymy using distributional similarity requires large volumes of data to reliably represent infrequent words. However, the naïve nearest-neighbour approach to comparing context vectors extracted from large corpora scales poorly ($O(n^2)$ in the vocabulary size).

In this paper, we compare several existing approaches to approximating the nearest-neighbour search for distributional similarity. We investigate the trade-off between efficiency and accuracy, and find that SASH (Houle and Sakuma, 2005) provides the best balance.

1 Introduction

It is a general property of Machine Learning that increasing the volume of training data increases the accuracy of results. This is no more evident than in Natural Language Processing (NLP), where massive quantities of text are required to model rare language events. Despite the rapid increase in computational power available for NLP systems, the volume of raw data available still outweighs our ability to process it. *Unsupervised learning*, which does not require the expensive and time-consuming human annotation of data, offers an opportunity to use this wealth of data. Curran and Moens (2002) show that synonymy extraction for lexical semantic resources using *distributional similarity* produces continuing gains in accuracy as the volume of input data increases.

Extracting synonymy relations using distributional similarity is based on the *distributional hypothesis* that *similar words appear in similar contexts*. Terms are described by collating information about their occurrence in a corpus into vectors. These *context vectors* are then compared for similarity. Existing approaches differ primarily in their definition of "context", e.g. the surrounding words or the entire document, and their choice of distance metric for calculating similarity between the context vectors representing each term.

Manual creation of lexical semantic resources is open to the problems of bias, inconsistency and limited coverage. It is difficult to account for the needs of the many domains in which NLP techniques are now being applied and for the rapid change in language use. The assisted or automatic creation and maintenance of these resources would be of great advantage.

Finding synonyms using distributional similarity requires a nearest-neighbour search over the context vectors of each term. This is computationally intensive, scaling to $O(n^2m)$ for the number of terms n and the size of their context vectors m. Increasing the volume of input data will increase the size of both n and m, decreasing the efficiency of a naïve nearest-neighbour approach.

Many approaches to reduce this complexity have been suggested. In this paper we evaluate state-of-the-art techniques proposed to solve this problem. We find that the Spatial Approximation Sample Hierarchy (Houle and Sakuma, 2005) provides the best accuracy/efficiency trade-off.

2 Distributional Similarity

Measuring distributional similarity first requires the extraction of context information for each of the vocabulary terms from raw text. These terms are then compared for similarity using a nearest-neighbour search or clustering based on distance calculations between the statistical descriptions of their contexts.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 361–368,
Sydney, July 2006. ©2006 Association for Computational Linguistics

2.1 Extraction

A *context relation* is defined as a tuple (w, r, w') where w is a term, which occurs in some grammatical relation r with another word w' in some sentence. We refer to the tuple (r, w') as an *attribute* of w. For example, (dog, direct-obj, walk) indicates that dog was the direct object of walk in a sentence.

In our experiments context extraction begins with a Maximum Entropy POS tagger and chunker. The SEXTANT relation extractor (Grefenstette, 1994) produces context relations that are then lemmatised. The relations for each term are collected together and counted, producing a vector of attributes and their frequencies in the corpus.

2.2 Measures and Weights

Both nearest-neighbour and cluster analysis methods require a distance measure to calculate the similarity between context vectors. Curran (2004) decomposes this into *measure* and *weight* functions. The *measure* calculates the similarity between two weighted context vectors and the *weight* calculates the informativeness of each context relation from the raw frequencies.

For these experiments we use the Jaccard (1) measure and the TTest (2) weight functions, found by Curran (2004) to have the best performance.

$$\frac{\sum_{(r,w')} \min(\mathrm{w}(w_m, r, w'), \mathrm{w}(w_n, r, w'))}{\sum_{(r,w')} \max(\mathrm{w}(w_m, r, w'), \mathrm{w}(w_n, r, w'))} \quad (1)$$

$$\frac{p(w, r, w') - p(*, r, w')p(w, *, *)}{\sqrt{p(*, r, w')p(w, *, *)}} \quad (2)$$

2.3 Nearest-neighbour Search

The simplest algorithm for finding synonyms is a k-nearest-neighbour (k-NN) search, which involves pair-wise vector comparison of the target term with every term in the vocabulary. Given an n term vocabulary and up to m attributes for each term, the asymptotic time complexity of nearest-neighbour search is $O(n^2m)$. This is very expensive, with even a moderate vocabulary making the use of huge datasets infeasible. Our largest experiments used a vocabulary of over 184,000 words.

3 Dimensionality Reduction

Using a cut-off to remove low frequency terms can significantly reduce the value of n. Unfortunately, reducing m by eliminating low frequency contexts has a significant impact on the quality of the results. There are many techniques to reduce dimensionality while avoiding this problem. The simplest methods use feature selection techniques, such as information gain, to remove the attributes that are less informative. Other techniques smooth the data while reducing dimensionality.

Latent Semantic Analysis (LSA, Landauer and Dumais, 1997) is a smoothing and dimensionality reduction technique based on the intuition that the true dimensionality of data is *latent* in the surface dimensionality. Landauer and Dumais admit that, from a pragmatic perspective, the same effect as LSA can be generated by using large volumes of data with very long attribute vectors. Experiments with LSA typically use attribute vectors of a dimensionality of around 1000. Our experiments have a dimensionality of 500,000 to 1,500,000. Decompositions on data this size are computationally difficult. Dimensionality reduction is often used before using LSA to improve its scalability.

3.1 Heuristics

Another technique is to use an initial heuristic comparison to reduce the number of full $O(m)$ vector comparisons that are performed. If the heuristic comparison is sufficiently fast and a sufficient number of full comparisons are avoided, the cost of an additional check will be easily absorbed by the savings made.

Curran and Moens (2002) introduces a vector of *canonical* attributes (of bounded length $k \ll m$), selected from the full vector, to represent the term. These attributes are the most strongly weighted verb attributes, chosen because they constrain the semantics of the term more and partake in fewer idiomatic collocations. If a pair of terms share at least one canonical attribute then a full similarity comparison is performed, otherwise the terms are not compared. They show an 89% reduction in search time, with only a 3.9% loss in accuracy.

There is a significant improvement in the computational complexity. If a maximum of p positive results are returned, our complexity becomes $O(n^2k + npm)$. When $p \ll n$, the system will be faster as many fewer full comparisons will be made, but at the cost of accuracy as more possibly near results will be discarded out of hand.

4 Randomised Techniques

Conventional dimensionality reduction techniques can be computationally expensive: a more scalable solution is required to handle the volumes of data we propose to use. Randomised techniques provide a possible solution to this.

We present two techniques that have been used recently for distributional similarity: Random Indexing (Kanerva et al., 2000) and Locality Sensitive Hashing (LSH, Broder, 1997).

4.1 Random Indexing

Random Indexing (RI) is a hashing technique based on *Sparse Distributed Memory* (Kanerva, 1993). Karlgren and Sahlgren (2001) showed RI produces results similar to LSA using the *Test of English as a Foreign Language* (TOEFL) evaluation. Sahlgren and Karlgren (2005) showed the technique to be successful in generating bilingual lexicons from parallel corpora.

In RI, we first allocate a d length *index vector* to each unique attribute. The vectors consist of a large number of 0s and small number (ϵ) number of randomly distributed ±1s. *Context vectors*, identifying terms, are generated by summing the index vectors of the attributes for each non-unique context in which a term appears. The context vector for a term t appearing in contexts $c_1 = [1, 0, 0, -1]$ and $c_2 = [0, 1, 0, -1]$ would be $[1, 1, 0, -2]$. The distance between these context vectors is then measured using the cosine measure:

$$\cos(\theta(u, v)) = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}| \; |\vec{v}|} \tag{3}$$

This technique allows for incremental sampling, where the index vector for an attribute is only generated when the attribute is encountered. Construction complexity is $O(nmd)$ and search complexity is $O(n^2d)$.

4.2 Locality Sensitive Hashing

LSH is a probabilistic technique that allows the approximation of a similarity function. Broder (1997) proposed an approximation of the Jaccard similarity function using min-wise independent functions. Charikar (2002) proposed an approximation of the cosine measure using random hyperplanes Ravichandran et al. (2005) used this cosine variant and showed it to produce over 70% accuracy in extracting synonyms when compared against Pantel and Lin (2002).

Given we have n terms in an m' dimensional space, we create $d \ll m'$ unit random vectors also of m' dimensions, labelled $\{\vec{r_1}, \vec{r_2}, ..., \vec{r_d}\}$. Each vector is created by sampling a Gaussian function m' times, with a mean of 0 and a variance of 1.

For each term w we construct its bit signature using the function

$$h_{\vec{r}}(\vec{w}) = \begin{cases} 1 & : \vec{r}.\vec{w} \geq 0 \\ 0 & : \vec{r}.\vec{w} < 0 \end{cases}$$

where $\vec{r}$ is a spherically symmetric random vector of length d. The signature, $\bar{w}$, is the d length bit vector:

$$\bar{w} = \{h_{\vec{r_1}}(\vec{w}), h_{\vec{r_2}}(\vec{w}), \ldots, h_{\vec{r_d}}(\vec{w})\}$$

The cost to build all n signatures is $O(nm'd)$.

For terms u and v, Goemans and Williamson (1995) approximate the angular similarity by

$$p(h_{\vec{r}}(\vec{u}) = h_{\vec{r}}(\vec{v})) = 1 - \frac{\theta(\vec{u}, \vec{u})}{\pi} \tag{4}$$

where $\theta(\vec{u}, \vec{u})$ is the angle between $\vec{u}$ and $\vec{u}$. The angular similarity gives the cosine by

$$\begin{aligned} cos(\theta(\vec{u}, \vec{u})) = \\ cos((1 - p(h_{\vec{r}}(\vec{u}) = h_{\vec{r}}(\vec{v})))\pi) \end{aligned} \tag{5}$$

The probability can be derived from the Hamming distance:

$$p(h_r(u) = h_r(v)) = 1 - \frac{\mathcal{H}(\bar{u}, \bar{v})}{d} \tag{6}$$

By combining equations 5 and 6 we get the following approximation of the cosine distance:

$$cos(\theta(\vec{u}, \vec{u})) = cos\left(\left(\frac{\mathcal{H}(\bar{u}, \bar{v})}{d}\right)\pi\right) \tag{7}$$

That is, the cosine of two context vectors is approximated by the cosine of the Hamming distance between their two signatures normalised by the size of the signatures. Search is performed using Equation 7 and scales to $O(n^2d)$.

5 Data Structures

The methods presented above fail to address the n^2 component of the search complexity. Many data structures have been proposed that can be used to address this problem in similarity searching. We present three data structures: the *vantage point tree* (VPT, Yianilos, 1993), which indexes points in a metric space, *Point Location in Equal*

Balls (PLEB, Indyk and Motwani, 1998), a probabilistic structure that uses the bit signatures generated by LSH, and the *Spatial Approximation Sample Hierarchy* (SASH, Houle and Sakuma, 2005), which approximates a k-NN search.

Another option inspired by IR is attribute indexing (INDEX). In this technique, in addition to each term having a reference to its attributes, each attribute has a reference to the terms referencing it. Each term is then only compared with the terms with which it shares attributes. We will give a theoretically comparison against other techniques.

5.1 Vantage Point Tree

Metric space data structures provide a solution to near-neighbour searches in very high dimensions. These rely solely on the existence of a comparison function that satisfies the conditions of metricality: non-negativity, equality, symmetry and the triangle inequality.

VPT is typical of these structures and has been used successfully in many applications. The VPT is a binary tree designed for range searches. These are searches limited to some distance from the target term but can be modified for k-NN search.

VPT is constructed recursively. Beginning with a set of U terms, we take any term to be our vantage point p. This becomes our root. We now find the median distance m_p of all other terms to p: $m_p = median\{dist(p,u)|u \in U\}$. Those terms u such that $dist(p,u) \leq m_p$ are inserted into the left sub-tree, and the remainder into the right subtree. Each sub-tree is then constructed as a new VPT, choosing a new vantage point from within its terms, until all terms are exhausted.

Searching a VPT is also recursive. Given a term q and radius r, we begin by measuring the distance to the root term p. If $dist(q,p) \leq r$ we enter p into our list of near terms. If $dist(q,p) - r \leq m_p$ we enter the left sub-tree and if $dist(q,p) + r > mp$ we enter the right sub-tree. Both sub-trees may be entered. The process is repeated for each entered subtree, taking the vantage point of the sub-tree to be the new root term.

To perform a k-NN search we use a *backtracking decreasing radius search* (Burkhard and Keller, 1973). The search begins with $r = \infty$, and terms are added to a list of the closest k terms. When the k^{th} closest term is found, the radius is set to the distance between this term and the target. Each time a new, closer element is added to the list, the radius is updated to the distance from the target to the new k^{th} closest term.

Construction complexity is $O(n \log n)$. Search complexity is claimed to be $O(\log n)$ for small radius searches. This does not hold for our decreasing radius search, whose worst case complexity is $O(n)$.

5.2 Point Location in Equal Balls

PLEB is a randomised structure that uses the bit signatures generated by LSH. It was used by Ravichandran et al. (2005) to improve the efficiency of distributional similarity calculations.

Having generated our d length bit signatures for each of our n terms, we take these signatures and randomly permute the bits. Each vector has the same permutation applied. This is equivalent to a column reordering in a matrix where the rows are the terms and the columns the bits. After applying the permutation, the list of terms is sorted lexicographically based on the bit signatures. The list is scanned sequentially, and each term is compared to its B nearest neighbours in the list. The choice of B will effect the accuracy/efficiency trade-off, and need not be related to the choice of k. This is performed q times, using a different random permutation function each time. After each iteration, the current closest k terms are stored.

For a fixed d, the complexity for the permutation step is $O(qn)$, the sorting $O(qn \log n)$ and the search $O(qBn)$.

5.3 Spatial Approximation Sample Hierarchy

SASH approximates a k-NN search by precomputing some near neighbours for each node (terms in our case). This produces multiple paths between terms, allowing SASH to shape itself to the data set (Houle, 2003). The following description is adapted from Houle and Sakuma (2005).

The SASH is a directed, edge-weighted graph with the following properties (see Figure 1):

- Each term corresponds to a unique node.
- The nodes are arranged into a hierarchy of levels, with the bottom level containing $\frac{n}{2}$ nodes and the top containing a single root node. Each level, except the top, will contain half as many nodes as the level below.
- Edges between nodes are linked to consecutive levels. Each node will have at most p *parent* nodes in the level above, and c *child* nodes in the level below.

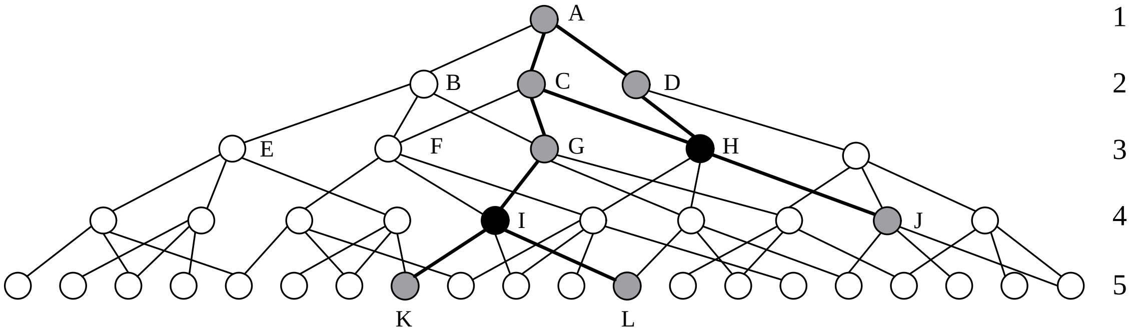


Figure 1: A SASH, where $p = 2$, $c = 3$ and $k = 2$

- Every node must have at least one parent so that all nodes are reachable from the root.

Construction begins with the nodes being randomly distributed between the levels. SASH is then constructed iteratively by each node finding its closest p parents in the level above. The parent will keep the closest c of these children, forming edges in the graph, and reject the rest. Any nodes without parents after being rejected are then assigned as children of the nearest node in the previous level with fewer than c children.

Searching is performed by finding the k nearest nodes at each level, which are added to a set of near nodes. To limit the search, only those nodes whose parents were found to be nearest at the previous level are searched. The k closest nodes from the set of near nodes are then returned. The search complexity is $O(ck \log n)$.

In Figure 1, the filled nodes demonstrate a search for the near-neighbours of some node q, using $k = 2$. Our search begins with the root node A. As we are using $k = 2$, we must find the two nearest children of A using our similarity measure. In this case, C and D are closer than B. We now find the closest two children of C and D. E is not checked as it is only a child of B. All other nodes are checked, including F and G, which are shared as children by B and C. From this level we chose G and H. The final levels are considered similarly.

At this point we now have the list of near nodes A, C, D, G, H, I, J, K and L. From this we chose the two nodes nearest q, H and I marked in black, which are then returned.

k can be varied at each level to force a larger number of elements to be tested at the base of the SASH using, for instance, the equation:

$$k_i = \max\{\, k^{1-\frac{h-i}{\log n}}, \frac{1}{2}pc\,\} \tag{8}$$

This changes our search complexity to:

$$\frac{k^{1+\frac{1}{\log n}}}{k^{\frac{1}{\log n}}-1} + \frac{pc^2}{2}\log n \tag{9}$$

We use this geometric function in our experiments.

Gorman and Curran (2005a; 2005b) found the performance of SASH for distributional similarity could be improved by replacing the initial random ordering with a frequency based ordering. In accordance with Zipf's law, the majority of terms have low frequencies. Comparisons made with these low frequency terms are unreliable (Curran and Moens, 2002). Creating SASH with high frequency terms near the root produces more reliable initial paths, but comparisons against these terms are more expensive.

The best accuracy/efficiency trade-off was found when using *more* reliable initial paths rather than the *most* reliable. This is done by *folding* the data around some mean number of relations. For each term, if its number of relations m_i is greater than some chosen number of relations $\mathcal{M}$, it is given a new ranking based on the score $\frac{\mathcal{M}^2}{m_i}$. Otherwise its ranking based on its number of relations. This has the effect of pushing very high and very low frequency terms away from the root.

6 Evaluation Measures

The simplest method for evaluation is the direct comparison of extracted synonyms with a manually created gold standard (Grefenstette, 1994). To reduce the problem of limited coverage, our evaluation combines three electronic thesauri: the Macquarie, Roget's and Moby thesauri.

We follow Curran (2004) and use two performance measures: direct matches (DIRECT) and inverse rank (INVR). DIRECT is the percentage of returned synonyms found in the gold standard. INVR is the sum of the inverse rank of each matching synonym, e.g. matches at ranks 3, 5 and 28

CORPUS	CUT-OFF	TERMS	AVERAGE RELATIONS PER TERM
BNC	0	246,067	43
	5	88,926	116
	100	14,862	617
LARGE	0	541,722	97
	5	184,494	281
	100	35,618	1,400

Table 1: Extracted Context Information

give an inverse rank score of $\frac{1}{3} + \frac{1}{5} + \frac{1}{28}$. With at most 100 matching synonyms, the maximum INVR is 5.187. This more fine grained as it incorporates the both the number of matches and their ranking. The same 300 single word nouns were used for evaluation as used by Curran (2004) for his large scale evaluation. These were chosen randomly from WordNet such that they covered a range over the following properties: *frequency*, *number of senses*, *specificity* and *concreteness*. For each of these terms, the closest 100 terms and their similarity scores were extracted.

7 Experiments

We use two corpora in our experiments: the smaller is the non-speech portion of the British National Corpus (BNC), 90 million words covering a wide range of domains and formats; the larger consists of the BNC, the Reuters Corpus Volume 1 and most of the English news holdings of the LDC in 2003, representing over 2 billion words of text (LARGE, Curran, 2004).

The semantic similarity system implemented by Curran (2004) provides our baseline. This performs a brute-force k-NN search (NAIVE). We present results for the canonical attribute heuristic (HEURISTIC), RI, LSH, PLEB, VPT and SASH.

We take the optimal canonical attribute vector length of 30 for HEURISTIC from Curran (2004). For SASH we take optimal values of $p = 4$ and $c = 16$ and use the folded ordering taking $\mathcal{M} = 1000$ from Gorman and Curran (2005b).

For RI, LSH and PLEB we found optimal values experimentally using the BNC. For LSH we chose $d = 3,000$ (LSH$_{3,000}$) and $10,000$ (LSH$_{10,000}$), showing the effect of changing the dimensionality. The frequency statistics were weighted using mutual information, as in Ravichandran et al. (2005):

$$\log(\frac{p(w,r,w')}{p(w,*,*)p(*,r,w')}) \qquad (10)$$

PLEB used the values $q = 500$ and $B = 100$.

	CUT-OFF 5	CUT-OFF 100
NAIVE	**1.72**	**1.71**
HEURISTIC	1.65	1.66
RI	0.80	0.93
LSH$_{10,000}$	1.26	1.31
SASH	**1.73**	**1.71**

Table 2: INVR vs frequency cut-off

The initial experiments on RI produced quite poor results. The intuition was that this was caused by the lack of smoothing in the algorithm. Experiments were performed using the weights given in Curran (2004). Of these, mutual information (10), evaluated with an extra $\log_2(f(w,r,w') + 1)$ factor and limited to positive values, produced the best results (RI$_{\text{MI}}$). The values $d = 1000$ and $\epsilon = 5$ were found to produce the best results.

All experiments were performed on 3.2GHz Xeon P4 machines with 4GB of RAM.

8 Results

As the accuracy of comparisons between terms increases with frequency (Curran, 2004), applying a frequency cut-off will both reduce the size of the vocabulary (n) and increase the average accuracy of comparisons. Table 1 shows the reduction in vocabulary and increase in average context relations per term as cut-off increases. For LARGE, the initial 541,722 word vocabulary is reduced by 66% when a cut-off of 5 is applied and by 86% when the cut-off is increased to 100. The average number of relations increases from 97 to 1400.

The work by Curran (2004) largely uses a frequency cut-off of 5. When this cut-off was used with the randomised techniques RI and LSH, it produced quite poor results. When the cut-off was increased to 100, as used by Ravichandran et al. (2005), the results improved significantly. Table 2 shows the INVR scores for our various techniques using the BNC with cut-offs of 5 and 100.

Table 3 shows the results of a full thesaurus extraction using the BNC and LARGE corpora using a cut-off of 100. The average DIRECT score and INVR are from the 300 test words. The total execution time is extrapolated from the average search time of these test words and includes the setup time. For LARGE, extraction using NAIVE takes 444 hours: over 18 days. If the 184,494 word vocabulary were used, it would take over 7000 hours, or nearly 300 days. This gives some indication of

	BNC			LARGE		
	DIRECT	INVR	Time	DIRECT	INVR	Time
NAIVE	5.23	1.71	38.0hr	5.70	1.93	444.3hr
HEURISTIC	4.94	1.66	2.0hr	5.51	1.93	30.2hr
RI	2.97	0.93	0.4hr	2.42	0.85	1.9hr
RI$_{\text{MI}}$	**3.49**	**1.41**	**0.4hr**	**4.58**	**1.75**	**1.9hr**
LSH$_{3,000}$	2.00	0.76	0.7hr	2.92	1.07	3.6hr
LSH$_{10,000}$	3.68	1.31	2.3hr	3.77	1.40	8.4hr
PLEB$_{3,000}$	2.00	0.76	1.2hr	2.85	1.07	4.1hr
PLEB$_{10,000}$	3.66	1.30	3.9hr	3.63	1.37	11.8hr
VPT	5.23	1.71	15.9hr	5.70	1.93	336.1hr
SASH	**5.17**	**1.71**	**2.0hr**	**5.29**	**1.89**	**23.7hr**

Table 3: Full thesaurus extraction

	LARGE		
CUT-OFF	0	5	100
NAIVE	541,721	184,493	35,617
SASH	10,599	8,796	6,231
INDEX	5,844	13,187	32,663

Table 4: Average number of comparisons per term

the scale of the problem.

The only technique to become less accurate when the corpus size is increased is RI; it is likely that RI is sensitive to high frequency, low information contexts that are more prevalent in LARGE. Weighting reduces this effect, improving accuracy.

The importance of the choice of d can be seen in the results for LSH. While much slower, LSH$_{10,000}$ is also much more accurate than LSH$_{3,000}$, while still being much faster than NAIVE. Introducing the PLEB data structure does not improve the efficiency while incurring a small cost on accuracy. We are not using large enough datasets to show the improved time complexity using PLEB.

VPT is only slightly faster slightly faster than NAIVE. This is not surprising in light of the original design of the data structure: decreasing radius search does not guarantee search efficiency.

A significant influence in the speed of the randomised techniques, RI and LSH, is the fixed dimensionality. The randomised techniques use a fixed length vector which is not influenced by the size of m. The drawback of this is that the size of the vector needs to be tuned to the dataset.

It would seem at first glance that HEURISTIC and SASH provide very similar results, with HEURISTIC slightly slower, but more accurate. This misses the difference in time complexity between the methods: HEURISTIC is n^2 and SASH $n \log n$. The improvement in execution time over NAIVE decreases as corpus size increases and this would be expected to continue. Further tuning of SASH parameters may improve its accuracy.

RI$_{\text{MI}}$ produces similar result using LARGE to SASH using BNC. This does not include the cost of extracting context relations from the raw text, so the true comparison is much worse. SASH allows the free use of weight and measure functions, but RI is constrained by having to transform any context space into a RI space. This is important when considering that different tasks may require different weights and measures (Weeds and Weir, 2005). RI also suffers n^2 complexity, where as SASH is $n \log n$. Taking these into account, and that the improvements are barely significant, SASH is a better choice.

The results for LSH are disappointing. It performs consistently worse than the other methods except VPT. This could be improved by using larger bit vectors, but there is a limit to the size of these as they represent a significant memory overhead, particularly as the vocabulary increases.

Table 4 presents the theoretical analysis of attribute indexing. The average number of comparisons made for various cut-offs of LARGE are shown. NAIVE and INDEX are the actual values for those techniques. The values for SASH are *worst case*, where the maximum number of terms are compared at each level. The actual number of comparisons made will be much less. The efficiency of INDEX is sensitive to the density of attributes and increasing the cut-off increases the density. This is seen in the dramatic drop in performance as the cut-off increases. This problem of density will increase as volume of raw input data increases, further reducing its effectiveness. SASH is only dependent on the number of terms, not the density.

Where the need for computationally efficiency out-weighs the need for accuracy, RI$_{\text{MI}}$ provides better results. SASH is the most balanced of the techniques tested and provides the most scalable, high quality results.

9 Conclusion

We have evaluated several state-of-the-art techniques for improving the efficiency of distributional similarity measurements. We found that, in terms of raw efficiency, Random Indexing (RI) was significantly faster than any other technique, but at the cost of accuracy. Even after our modifications to the RI algorithm to significantly improve its accuracy, SASH still provides a better accuracy/efficiency trade-off. This is more evident when considering the time to extract context information from the raw text. SASH, unlike RI, also allows us to choose both the weight and the measure used. LSH and PLEB could not match either the efficiency of RI or the accuracy of SASH.

We intend to use this knowledge to process even larger corpora to produce more accurate results.

Having set out to improve the efficiency of distributional similarity searches while limiting any loss in accuracy, we are producing full nearest-neighbour searches 18 times faster, with only a 2% loss in accuracy.

Acknowledgements

We would like to thank our reviewers for their helpful feedback and corrections. This work has been supported by the Australian Research Council under Discovery Project DP0453131.

References

Andrei Broder. 1997. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences*, pages 21–29, Salerno, Italy.

Walter A. Burkhard and Robert M. Keller. 1973. Some approaches to best-match file searching. *Communications of the ACM*, 16(4):230–236, April.

Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 380–388, Montreal, Quebec, Canada, 19–21 May.

James Curran and Marc Moens. 2002. Improvements in automatic thesaurus extraction. In *Proceedings of the Workshop of the ACL Special Interest Group on the Lexicon*, pages 59–66, Philadelphia, PA, USA, 12 July.

James Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

Michel X. Goemans and David P. Williamson. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of Association for Computing Machinery*, 42(6):1115–1145, November.

James Gorman and James Curran. 2005a. Approximate searching for distributional similarity. In *ACL-SIGLEX 2005 Workshop on Deep Lexical Acquisition*, Ann Arbor, MI, USA, 30 June.

James Gorman and James Curran. 2005b. Augmenting approximate similarity searching with lexical information. In *Australasian Language Technology Workshop*, Sydney, Australia, 9–11 November.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Boston.

Michael E. Houle and Jun Sakuma. 2005. Fast approximate similarity search in extremely high-dimensional data sets. In *Proceedings of the 21st International Conference on Data Engineering*, pages 619–630, Tokyo, Japan.

Michael E. Houle. 2003. Navigating massive data sets via local clustering. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 547–552, Washington, DC, USA.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the 30th annual ACM Symposium on Theory of Computing*, pages 604–613, New York, NY, USA, 24–26 May. ACM Press.

Pentti Kanerva, Jan Kristoferson, and Anders Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, page 1036, Mahwah, NJ, USA.

Pentti Kanerva. 1993. Sparse distributed memory and related models. In M.H. Hassoun, editor, *Associative Neural Memories: Theory and Implementation*, pages 50–76. Oxford University Press, New York, NY, USA.

Jussi Karlgren and Magnus Sahlgren. 2001. From words to understanding. In Y. Uesaka, P. Kanerva, and H Asoh, editors, *Foundations of Real-World Intelligence*, pages 294–308. CSLI Publications, Stanford, CA, USA.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240, April.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of ACM SIGKDD-02*, pages 613–619, 23–26 July.

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy. 2005. Randomized algorithms and NLP: Using locality sensitive hash functions for high speed noun clustering. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 622–629, Ann Arbor, USA.

Mangus Sahlgren and Jussi Karlgren. 2005. Automatic bilingual lexicon acquisition using random indexing of parallel corpora. *Journal of Natural Language Engineering, Special Issue on Parallel Texts*, 11(3), June.

Julie Weeds and David Weir. 2005. Co-occurance retrieval: A flexible framework for lexical distributional similarity. *Computational Linguistics*, 31(4):439–475, December.

Peter N. Yianilos. 1993. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 311–321, Philadelphia.

Extractive Summarization using Inter- and Intra- Event Relevance

Wenjie Li, Mingli Wu and Qin Lu
Department of Computing
The Hong Kong Polytechnic University
{cswjli,csmlwu,csluqin}@comp.polyu.edu.hk

Wei Xu and Chunfa Yuan
Department of Computer Science and Technology, Tsinghua University
{vivian00,cfyuan}@mail.tsinghua.edu.cn

Abstract

Event-based summarization attempts to select and organize the sentences in a summary with respect to the events or the sub-events that the sentences describe. Each event has its own internal structure, and meanwhile often relates to other events semantically, temporally, spatially, causally or conditionally. In this paper, we define an event as one or more event terms along with the named entities associated, and present a novel approach to derive intra- and inter- event relevance using the information of internal association, semantic relatedness, distributional similarity and named entity clustering. We then apply PageRank ranking algorithm to estimate the significance of an event for inclusion in a summary from the event relevance derived. Experiments on the DUC 2001 test data shows that the relevance of the named entities involved in events achieves better result when their relevance is derived from the event terms they associate. It also reveals that the topic-specific relevance from documents themselves outperforms the semantic relevance from a general purpose knowledge base like Word-Net.

1. Introduction

Extractive summarization selects sentences which contain the most salient concepts in documents. Two important issues with it are how the concepts are defined and what criteria should be used to judge the salience of the concepts. Existing work has typically been based on techniques that extract key textual elements, such as keywords (also known as significant terms) as weighed by their tf*idf score, or concepts (such as events or entities) with linguistic and/or statistical analysis. Then, sentences are selected according to either the important textual units they contain or certain types of inter-sentence relations they hold.

Event-based summarization which has emerged recently attempts to select and organize sentences in a summary with respect to events or sub-events that the sentences describe. With regard to the concept of events, people do not have the same definition when introducing it in different domains. While traditional linguistics work on semantic theory of events and the semantic structures of verbs, studies in information retrieval (IR) within topic detection and tracking framework look at events as narrowly defined topics which can be categorized or clustered as a set of related documents (TDT). IR events are broader (or to say complex) events in the sense that they may include happenings and their causes, consequences or even more extended effects. In the information extraction (IE) community, events are defined as the pre-specified and structured templates that relate an action to its participants, times, locations and other entities involved (MUC-7). IE defines what people call atomic events.

Regardless of their distinct perspectives, people all agree that events are collections of activities together with associated entities. To apply the concept of events in the context of text summarization, we believe it is more appropriate to consider events at the sentence level, rather than at the document level. To avoid the complexity of deep semantic and syntactic processing, we complement the advantages of statistical techniques from the IR community and structured information provided by the IE community.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 369–376,
Sydney, July 2006. ©2006 Association for Computational Linguistics

We propose to extract semi-structured events with shallow natural language processing (NLP) techniques and estimate their importance for inclusion in a summary with IR techniques.

Though it is most likely that documents narrate more than one similar or related event, most event-based summarization techniques reported so far explore the importance of the events independently. Motivated by this observation, this paper addresses the task of event-relevance based summarization and explores what sorts of relevance make a contribution. To this end, we investigate intra-event relevance, that is action-entity relevance, and inter-event relevance, that is event-event relevance. While intra-event relevance is measured with frequencies of the associated events and entities directly, inter-event relevance is derived indirectly from a general WordNet similarity utility, distributional similarity in the documents to be summarized, named entity clustering and so on. Pagerank ranking algorithm is then applied to estimate the event importance for inclusion in a summary using the aforesaid relevance.

The remainder of this paper is organized as follows. Section 2 introduces related work. Sections 3 introduces our proposed event-based summarization approaches which make use of intra- and inter- event relevance. Section 4 presents experiments and evaluates different approaches. Finally, Section 5 concludes the paper.

2. Related Work

Event-based summarization has been investigated in recent research. It was first presented in (Daniel, Radev and Allison, 2003), who treated a news topic in multi-document summarization as a series of sub-events according to human understanding of the topic. They determined the degree of sentence relevance to each sub-event through human judgment and evaluated six extractive approaches. Their paper concluded that recognizing the sub-events that comprise a single news event is essential for producing better summaries. However, it is difficult to automatically break a news topic into sub-events.

Later, atomic events were defined as the relationships between the important named entities (Filatova and Hatzivassiloglou, 2004), such as participants, locations and times (which are called relations) through the verbs or action nouns labeling the events themselves (which are called connectors). They evaluated sentences based on co-occurrence statistics of the named entity relations and the event connectors involved. The proposed approach claimed to outperform conventional tf*idf approach. Apparently, named entities are key elements in their model. However, the constraints defining events seemed quite stringent.

The application of dependency parsing, anaphora and co-reference resolution in recognizing events were presented involving NLP and IE techniques more or less (Yoshioka and Haraguchi, 2004), (Vanderwende, Banko and Menezes, 2004) and (Leskovec, Grobelnik and Fraling, 2004). Rather than pre-specifying events, these efforts extracted (verb)-(dependent relation)-(noun) triples as events and took the triples to form a graph merged by relations.

As a matter of fact, events in documents are related in some ways. Judging whether the sentences are salient or not and organizing them in a coherent summary can take advantage from event relevance. Unfortunately, this was neglected in most previous work. Barzilay and Lapata (2005) exploited the use of the distributional and referential information of discourse entities to improve summary coherence. While they captured text relatedness with entity transition sequences, i.e. entity-based summarization, we are particularly interested in relevance between events in event-based summarization.

Extractive summarization requires ranking sentences with respect to their importance. Successfully used in Web-link analysis and more recently in text summarization, Google's PageRank (Brin and Page, 1998) is one of the most popular ranking algorithms. It is a kind of graph-based ranking algorithm deciding on the importance of a node within a graph by taking into account the global information recursively computed from the entire graph, rather than relying on only the local node-specific information. A graph can be constructed by adding a node for each sentence, phrase or word. Edges between nodes are established using inter-sentence similarity relations as a function of content overlap or grammatically relations between words or phrases.

The application of PageRank in sentence extraction was first reported in (Erkan and Radev, 2004). The similarity between two sentence nodes according to their term vectors was used to generate links and define link strength. The same idea was followed and investigated extensively (Mihalcea, 2005). Yoshioka and Haraguchi (2004) went one step further toward event-based summarization. Two sentences were linked if they shared similar events. When tested on TSC-3, the approach favoured longer summaries. In contrast, the importance of the verbs and nouns constructing events was evaluated with PageRank as individual nodes aligned by their dependence relations (Vanderwende, 2004; Leskovec, 2004).

Although we agree that the fabric of event constitutions constructed by their syntactic relations can help dig out the important events, we have two comments. First, not all verbs denote event happenings. Second, semantic similarity or relatedness between action words should be taken into account.

3. Event-based Summarization

3.1. Event Definition and Event Map

Events can be broadly defined as "Who did What to Whom When and Where". Both linguistic and empirical studies acknowledge that event arguments help characterize the effects of a verb's event structure even though verbs or other words denoting event determine the semantics of an event. In this paper, we choose verbs (such as "elect") and action nouns (such as "supervision") as event terms that can characterize or partially characterize actions or incident occurrences. They roughly relate to "did What". One or more associated named entities are considered as what are denoted by linguists as event arguments. Four types of named entities are currently under the consideration. These are <Person>, <Organization>, <Location> and <Date>. They convey the information of "Who", "Whom", "When" and "Where". A verb or an action noun is deemed as an event term only when it presents itself at least once between two named entities.

Events are commonly related with one another semantically, temporally, spatially, causally or conditionally, especially when the documents to be summarized are about the same or very similar topics. Therefore, all event terms and named entities involved can be explicitly connected or implicitly related and weave a document or a set of documents into an event fabric, i.e. an event graphical representation (see Figure 1). The nodes in the graph are of two types. Event terms (*ET*) are indicated by rectangles and named entities (*NE*) are indicated by ellipses. They represent concepts rather than instances. Words in either their original form or morphological variations are represented with a single node in the graph regardless of how many times they appear in documents. We call this representation an event map, from which the most important concepts can be pick out in the summary.

<Organization> America Online </Organization> was to buy <Organization> Netscape </Organization> and forge a partnership with <Organization> Sun </Organization>, benefiting all three and giving technological independence from <Organization> Microsoft </Organization>.

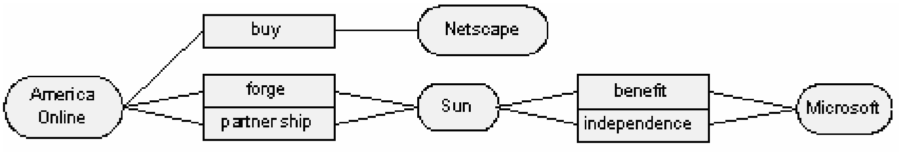


Figure 1 Sample sentences and their graphical representation

The advantage of representing with separated action and entity nodes over simply combining them into one event or sentence node is to provide a convenient way for analyzing the relevance among event terms and named entities either by their semantic or distributional similarity. More importantly, this favors extraction of concepts and brings the conceptual compression available.

We then integrate the strength of the connections between nodes into this graphical model in terms of the relevance defined from different perspectives. The relevance is indicated by $r(node_i, node_j)$, where $node_i$ and $node_j$ represent two nodes, and are either event terms (et_i) or named entities (ne_j). Then, the significance of each node, indicated by $w(node_i)$, is calculated with PageRank ranking algorithm. Sections 3.2 and 3.3 address the issues of deriving $r(node_i, node_j)$ according to intra- or/and inter-event relevance and calculating $w(node_i)$ in detail.

3.2 Intra- and Inter- Event Relevance

We consider both intra-event and inter-event relevance for summarization. Intra-event relevance measures how an action itself is associated with its associated arguments. It is indicated as $R(ET, NE)$ and $R(NE, ET)$ in Table 1 below. This is a kind of direct relevance as the connections between actions and arguments are established from the text surface directly. No inference or background knowledge is required. We consider that when the connection between an event term et_i and a named entity ne_j is symmetry, then $R(NE, ET) = R(ET, NE)^T$. Events are related as explained in Section 2. By means of inter-event relevance, we consider how an event term (or a named entity involved in an event) associate to another event term (or another named entity involved in the same or different events) syntactically, semantically and distributionally. It is indicated by $R(ET, ET)$ or $R(NE, NE)$ in Table 1 and measures an indirect connection which is not explicit in the event map needing to be derived from the external resource or overall event distribution.

	Event Term (*ET*)	Named Entity (*NE*)
Event Term (*ET*)	$R(ET, ET)$	$R(ET, NE)$
Named Entity (*NE*)	$R(NE, ET)$	$R(NE, NE)$

Table 1 Relevance Matrix

The complete relevance matrix is:

$$R = \begin{bmatrix} R(ET, ET) & R(ET, NE) \\ R(NE, ET) & R(NE, NE) \end{bmatrix}$$

The intra-event relevance $R(ET, NE)$ can be simply established by counting how many times et_i and ne_j are associated, i.e.

$$r_{Document}(et_i, ne_j) = freq(et_i, ne_j) \qquad \text{(E1)}$$

One way to measure the term relevance is to make use of a general language knowledge base, such as WordNet (Fellbaum 1998). WordNet::Similarity is a freely available software package that makes it possible to measure the semantic relatedness between a pair of concepts, or in our case event terms, based on WordNet (Pedersen, Patwardhan and Michelizzi, 2004). It supports three measures. The one we choose is the function *lesk*.

$$r_{WordNet}(et_i, et_j) = similarity(et_i, et_j) = lesk(et_i, et_j) \qquad \text{(E2)}$$

Alternatively, term relevance can be measured according to their distributions in the specified documents. We believe that if two events are concerned with the same participants, occur at same location, or at the same time, these two events are interrelated with each other in some ways. This observation motivates us to try deriving event term relevance from the number of name entities they share.

$$r_{Document}(et_i, et_j) = | NE(et_i) \cap NE(et_j) | \qquad \text{(E3)}$$

Where $NE(et_i)$ is the set of named entities et_i associate. | | indicates the number of the elements in the set. The relevance of named entities can be derived in a similar way.

$$r_{Document}(ne_i, ne_j) = | ET(ne_i) \cap ET(ne_j) | \qquad \text{(E4)}$$

The relevance derived with (E3) and (E4) are indirect relevance. In previous work, a clustering algorithm, shown in Figure 2, has been proposed (Xu et al, 2006) to merge the named entity that refer to the same person (such as Ranariddh, Prince Norodom Ranariddh and President Prince Norodom Ranariddh). It is used for co-reference resolution and aims at joining the same concept into a single node in the event map. The experimental result suggests that merging named entity improves performance in some extend but not evidently. When applying the same algorithm for clustering all four types of name entities in DUC data, we observe that the name entities in the same cluster do not always refer to the same objects, even when they are indeed related in some way. For example, “Mississippi” is a state in the southeast United States, while “Mississippi River” is the second-longest rever in the United States and flows through “Mississippi”.

```
Step1: Each name entity is represented by
        $ne_i = w_{i1} w_{i2} ... w_{ik}$, where $w_i$ is the ith
        word in it. The cluster it belongs to, in-
        dicated by $C(ne_i)$, is initialled by
        $w_{i1} w_{i2} ... w_{ik}$ itself.
Step2: For each name entity
        $ne_i = w_{i1} w_{i2} ... w_{ik}$
            For each name entity
```

$ne_j = w_{j1}w_{j2}...w_{jl}$, if $C(ne_i)$ is a sub-string of $C(ne_j)$, then $C(ne_i) = C(ne_j)$.

Continue Step 2 until no change occurs.

Figure 2 The algorithm proposed to merge the named entities

Location	Person	Date	Organization
Mississippi	Professor Sir Richard Southwood	first six months of last year	Long Beach City Council
Mississippi River	Sir Richard Southwood	last year	San Jose City Council
	Richard Southwood		City Council

Table 2 Some results of the named entity merged

It therefore provides a second way to measure named entity relevance based on the clusters found. It is actually a kind of measure of lexical similarity.

$$r_{Cluster}(ne_i, ne_j) = \begin{cases} 1, & ne_i, ne_j \text{ are in the same cluster} \\ 0, & \text{otherwise} \end{cases}$$

(E5)

In addition, the relevance of the named entities can be sometimes revealed by sentence context. Take the following most frequently used sentence patterns as examples:

<Person>, a-position-name of <Organization>, does something.
<Person> and another <Person> do something.

Figure 3 The example patterns

Considering that two neighbouring name entities in a sentence are usually relevant, the following window-based relevance is also experimented with.

$$r_{Pattern}(ne_i, ne_j) = \begin{cases} 1, & ne_i, ne_j \text{ are within a pre-specified window size} \\ 0, & \text{otherwise} \end{cases}$$

(E6)

3.3 Significance of Concepts

The significance score, i.e. the weight $w(node_i)$ of each $node_i$, is then estimated recursively with PageRank ranking algorithm which assigns the significance score to each node according to the number of nodes connecting to it as well as the strength of their connections. The equation calculating $w(node_i)$ using PageRank of a certain $node_i$ is shown as follows.

$$w(node_i) = (1-d) + d\left(\frac{w(node_1)}{r(node_i, node_1)} + ... + \frac{w(node_j)}{r(node_i, node_j)} + ... + \frac{w(node_t)}{r(node_i, node_t)}\right) \quad \text{(E7)}$$

In (E7), $node_j$ ($j = 1,2,...t$, $j \neq i$) are the nodes linking to $node_i$. d is the factor used to avoid the limitation of loop in the map structure. It is set to 0.85 experimentally. The significance of each sentence to be included in the summary is then obtained from the significance of the events it contains. The sentences with higher significance are picked up into the summary as long as they are not exactly the same sentences. We are aware of the important roles of information fusion and sentence compression in summary generation. However, the focus of this paper is to evaluate event-based approaches in extracting the most important sentences. Conceptual extraction based on event relevance is our future direction.

4. Experiments and Discussions

To evaluate the event based summarization approaches proposed, we conduct a set of experiments on 30 English document sets provide by the DUC 2001 multi-document summarization task. The documents are pre-processed with GATE to recognize the previously mentioned four types of name entities. On average, each set contains 10.3 documents, 602 sentences, 216 event terms and 148.5 name entities.

To evaluate the quality of the generated summaries, we choose an automatic summary evaluation metric ROUGE, which has been used in DUCs. ROUGE is a recall-based metric for fixed length summaries. It bases on *N*-gram co-occurrence and compares the system generated summaries to human judges (Lin and Hovy, 2003). For each DUC document set, the system creates a summary of 200 word length and present three of the ROUGE metrics: ROUGE-1 (unigram-based), ROUGE-2 (bigram-based), and ROUGE-W (based on longest common subsequence weighed by the length) in the following experiments and evaluations.

We first evaluate the summaries generated based on $R(ET, NE)$ itself. In the pre-evaluation experiments, we have observed that some frequently occurring nouns, such as “doctors” and “hospitals”, by themselves are not marked by general NE taggers. But they indicate persons, organizations or locations. We compare the ROUGE scores of adding frequent nouns or not to the set of named entities in Table 3. A noun is considered as a frequent noun when its frequency is larger than 10. Roughly 5% improvement is achieved when high frequent nouns are taken into the consideration. Hereafter, when we mention *NE* in latter experiments, the high frequent nouns are included.

$R(ET,NE)$	*NE* Without High Frequency Nouns	*NE* With High Frequency Nouns
ROUGE-1	0.33320	0.34859
ROUGE-2	0.06260	0.07157
ROUGE-W	0.12965	0.13471

Table 3 ROUGE scores using $R(ET,NE)$ itself

Table 4 below then presents the summarization results by using $R(ET,ET)$ itself. It compares two relevance derivation approaches, $R_{WordNet}$ and $R_{Document}$. The topic-specific relevance derived from the documents to be summarized outperforms the general purpose Word-Net relevance by about 4%. This result is reasonable as WordNet may introduce the word relatedness which is not necessary in the topic-specific documents. When we examine the relevance matrix from the event term pairs with the highest relevant, we find that the pairs, like “abort” and “confirm”, “vote” and confirm”, do reflect semantics (antonymous) and associated (causal) relations to some degree.

$R(ET,ET)$	Semantic Relevance from Word-Net	Topic-Specific Relevance from Documents
ROUGE-1	0.32917	0.34178
ROUGE-2	0.05737	0.06852
ROUGE-W	0.11959	0.13262

Table 4 ROUGE scores using $R(ET,ET)$ itself

Surprisingly, the best individual result is from document distributional similarity $R_{Document}$ (NE,NE) in Table 5. Looking more closely, we conclude that compared to event terms, named entities are more representative of the documents in which they are included. In other words, event terms are more likely to be distributed around all the document sets, whereas named entities are more topic-specific and therefore cluster in a particular document set more. Examples of high related named entities in relevance matrix are “Andrew” and “Florida”, “Louisiana” and “Florida”. Although their relevance is not as explicit as the same of event terms (their relevance is more contextual than semantic), we can still deduce that some events may happen in both Louisiana and Florida, or about Andrew in Florida. In addition, it also shows that the relevance we would have expected to be derived from patterns and clustering can also be discovered by $R_{Document}(NE,NE)$. The window size is set to 5 experimentally in window-based practice.

$R(NE,NE)$	Relevance from Documents	Relevance from Clustering	Relevance from Window-based Context
ROUGE-1	0.35212	0.33561	0.34466
ROUGE-2	0.07107	0.07286	0.07508
ROUGE-W	0.13603	0.13109	0.13523

Table 5 ROUGE scores using $R(NE,NE)$ itself

Next, we evaluate the integration of $R(ET,NE)$, $R(ET,ET)$ and $R(NE,NE)$. As DUC 2001 provides 4 different summary sizes for evaluation, it satisfies our desire to test the sensibility of the proposed event-based summarization techniques to the length of summaries. While the previously presented results are evaluated on 200 word summaries, now we move to check the results in four different sizes, i.e. 50, 100, 200 and 400 words. The experiments results show that the event-based approaches indeed prefer longer summaries. This is coincident with what we have hypothesized. For this set of experiments, we choose to integrate the best method from each individual evaluation presented previously. It appears that using the named entities relevance which is derived from the event terms gives the best ROUGE scores in almost all the summery sizes. Compared with the results provided in (Filatova and Hatzivassiloglou, 2004) whose average ROUGE-1 score is below 0.3 on the same data set, the significant improvement is revealed. Of course, we need to test on more data in the future.

$R(NE,NE)$	50	100	200	400
ROUGE-1	0.22383	0.28584	0.35212	0.41612
ROUGE-2	0.03376	0.05489	0.07107	0.10275
ROUGE-W	0.10203	0.11610	0.13603	0.13877
$R(ET,NE)$	50	100	200	400
ROUGE-1	0.22224	0.27947	0.34859	0.41644
ROUGE-2	0.03310	0.05073	0.07157	0.10369
ROUGE-W	0.10229	0.11497	0.13471	0.13850
$R(ET,ET)$	50	100	200	400

ROUGE-1	0.20616	0.26923	0.34178	0.41201
ROUGE-2	0.02347	0.04575	0.06852	0.10263
ROUGE-W	0.09212	0.11081	0.13262	0.13742
$R(ET,NE)+$ $R(ET,ET)+$ $R(NE,NE)$	50	100	200	400
ROUGE-1	0.21311	0.27939	0.34630	0.41639
ROUGE-2	0.03068	0.05127	0.07057	0.10579
ROUGE-W	0.09532	0.11371	0.13416	0.13913

Table 6 ROUGE scores using complete *R* matrix and with different summary lengths

As discussed in Section 3.2, the named entities in the same cluster may often be relevant but not always be co-referred. In the following last set of experiments, we evaluate the two ways to use the clustering results. One is to consider them as related as if they are in the same cluster and derive the *NE-NE* relevance with (E5). The other is to merge the entities in one cluster as one reprehensive named entity and then use it in *ET-NE* with (E1). The rationality of the former approach is validated.

	Clustering is used to derive *NE-NE*	Clustering is used to merge entities and then to derive *ET-NE*
ROUGE-1	0.34072	0.33006
ROUGE-2	0.06727	0.06154
ROUGE-W	0.13229	0.12845

Table 7 ROUGE scores with regard to how to use the clustering information

5. Conclusion

In this paper, we propose to integrate event-based approaches to extractive summarization. Both inter-event and intra-event relevance are investigated and PageRank algorithm is used to evaluate the significance of each concept (including both event terms and named entities). The sentences containing more concepts and highest significance scores are chosen in the summary as long as they are not the same sentences.

To derive event relevance, we consider the associations at the syntactic, semantic and contextual levels. An important finding on the DUC 2001 data set is that making use of named entity relevance derived from the event terms they associate with achieves the best result. The result of 0.35212 significantly outperforms the one reported in the closely related work whose average is below 0.3. We are interested in the issue of how to improve an event representation in order to build a more powerful event-based summarization system. This would be one of our future directions. We also want to see how concepts rather than sentences are selected into the summary in order to develop a more flexible compression technique and to know what characteristics of a document set is appropriate for applying event-based summarization techniques.

Acknowledgements

The work presented in this paper is supported partially by Research Grants Council on Hong Kong (reference number CERG PolyU5181/03E) and partially by National Natural Science Foundation of China (reference number: NSFC 60573186).

References

Chin-Yew Lin and Eduard Hovy. 2003. Automatic Evaluation of Summaries using N-gram Co-occurrence Statistics. In Proceedings of HLT-NAACL 2003, pp71-78.

Christiane Fellbaum. 1998, WordNet: An Electronic Lexical Database. MIT Press.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-based Extractive summarization. In Proceedings of ACL 2004 Workshop on Summarization, pp104-111.

Gunes Erkan and Dragomir Radev. 2004. LexRank: Graph-based Centrality as Salience in Text Summarization. Journal of Artificial Intelligence Research.

Jure Leskovec, Marko Grobelnik and Natasa Milic-Frayling. 2004. Learning Sub-structures of Document Semantic Graphs for Document Summarization. In LinkKDD 2004.

Lucy Vanderwende, Michele Banko and Arul Menezes. 2004. Event-Centric Summary Generation. In Working Notes of DUC 2004.

Masaharu Yoshioka and Makoto Haraguchi. 2004. Multiple News Articles Summarization based on Event Reference Information. In Working Notes of NTCIR-4, Tokyo.

MUC-7. http://www-nlpir.nist.gov/related_projects/ muc/proceeings/ muc_7_toc.html

Naomi Daniel, Dragomir Radev and Timothy Allison. 2003. Sub-event based Multi-document Summarization. In Proceedings of the HLT-NAACL 2003 Workshop on Text Summarization, pp9-16.

Page Lawrence, Brin Sergey, Motwani Rajeev and Winograd Terry. 1998. The PageRank Citation Ranking: Bring Order to the Web. Technical Report, Stanford University.

Rada Mihalcea. 2005. Language Independent Extractive Summarization. ACL 2005 poster.

Regina Barzilay and Michael Elhadad. 2005. Modelling Local Coherence: An Entity-based Approach. In Proceedings of ACL, pp141-148.

TDT. http://projects.ldc.upenn.edu/TDT.

Ted Pedersen, Siddharth Patwardhan and Jason Michelizzi. 2004. WordNet::Similarity – Measuring the Relatedness of Concepts. In Proceedings of AAAI, pp25-29.

Wei Xu, Wenjie Li, Mingli Wu, Wei Li and Chunfa Yuan. 2006. Deriving Event Relevance from the Ontology Constructed with Formal Concept Analysis, in Proceedings of CiCling’06, pp480-489.

Models for Sentence Compression: A Comparison across Domains, Training Requirements and Evaluation Measures

James Clarke and **Mirella Lapata**
School of Informatics, University of Edinburgh
2 Bucclecuch Place, Edinburgh EH8 9LW, UK
jclarke@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

Sentence compression is the task of producing a summary at the sentence level. This paper focuses on three aspects of this task which have not received detailed treatment in the literature: training requirements, scalability, and automatic evaluation. We provide a novel comparison between a supervised constituent-based and an weakly supervised word-based compression algorithm and examine how these models port to different domains (written vs. spoken text). To achieve this, a human-authored compression corpus has been created and our study highlights potential problems with the automatically gathered compression corpora currently used. Finally, we assess whether automatic evaluation measures can be used to determine compression quality.

1 Introduction

Automatic sentence compression has recently attracted much attention, in part because of its affinity with summarisation. The task can be viewed as producing a summary of a single sentence that retains the most important information while remaining grammatically correct. An ideal compression algorithm will involve complex text rewriting operations such as word reordering, paraphrasing, substitution, deletion, and insertion. In default of a more sophisticated compression algorithm, current approaches have simplified the problem to a single rewriting operation, namely word deletion. More formally, given an input sentence of words $W = w_1, w_2, \ldots, w_n$, a compression is formed by dropping any subset of these words. Viewing the task as word removal reduces the number of possible compressions to 2^n; naturally, many of these compressions will not be reasonable or grammatical (Knight and Marcu 2002).

Sentence compression could be usefully employed in wide range of applications. For example, to automatically generate subtitles for television programs; the transcripts cannot usually be used verbatim due to the rate of speech being too high (Vandeghinste and Pan 2004). Other applications include compressing text to be displayed on small screens (Corston-Oliver 2001) such as mobile phones or PDAs, and producing audio scanning devices for the blind (Grefenstette 1998).

Algorithms for sentence compression fall into two broad classes depending on their training requirements. Many algorithms exploit parallel corpora (Jing 2000; Knight and Marcu 2002; Riezler et al. 2003; Nguyen et al. 2004a; Turner and Charniak 2005; McDonald 2006) to learn the correspondences between long and short sentences in a supervised manner, typically using a rich feature space induced from parse trees. The learnt rules effectively describe which constituents should be deleted in a given context. Approaches that do not employ parallel corpora require minimal or no supervision. They operationalise compression in terms of word deletion without learning specific rules and can therefore rely on little linguistic knowledge such as part-of-speech tags or merely the lexical items alone (Hori and Furui 2004). Alternatively, the rules of compression are approximated from a non-parallel corpus (e.g., the Penn Treebank) by considering context-free grammar derivations with matching expansions (Turner and Charniak 2005).

Previous approaches have been developed and tested almost exclusively on written text, a notable exception being Hori and Furui (2004) who focus on spoken language. While parallel corpora of original-compressed sentences are not naturally available in the way multilingual corpora are, researchers have obtained such corpora automatically by exploiting documents accompanied by abstracts. Automatic corpus creation affords the opportunity to study compression mechanisms

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 377–384,
Sydney, July 2006.

cheaply, yet these mechanisms may not be representative of human performance. It is unlikely that authors routinely carry out sentence compression while creating abstracts for their articles. Collecting human judgements is the method of choice for evaluating sentence compression models. However, human evaluations tend to be expensive and cannot be repeated frequently; furthermore, comparisons across different studies can be difficult, particularly if subjects employ different scales, or are given different instructions.

In this paper we examine some aspects of the sentence compression task that have received little attention in the literature. First, we provide a novel comparison of supervised and weakly supervised approaches. Specifically, we study how constituent-based and word-based methods port to different domains and show that the latter tend to be more robust. Second, we create a corpus of human-authored compressions, and discuss some potential problems with currently used compression corpora. Finally, we present automatic evaluation measures for sentence compression and examine whether they correlate reliably with behavioural data.

2 Algorithms for Sentence Compression

In this section we give a brief overview of the algorithms we employed in our comparative study. We focus on two representative methods, Knight and Marcu's (2002) decision-based model and Hori and Furui's (2004) word-based model.

The decision-tree model operates over parallel corpora and offers an intuitive formulation of sentence compression in terms of tree rewriting. It has inspired many discriminative approaches to the compression task (Riezler et al. 2003; Nguyen et al. 2004b; McDonald 2006) and has been extended to languages other than English (see Nguyen et al. 2004a). We opted for the decision-tree model instead of the also well-known noisy-channel model (Knight and Marcu 2002; Turner and Charniak 2005). Although both models yield comparable performance, Turner and Charniak (2005) show that the latter is not an appropriate compression model since it favours uncompressed sentences over compressed ones.[1]

Hori and Furui's (2004) model was originally developed for Japanese with spoken text in mind, it requires minimal supervision, and little linguistic knowledge. It therefor holds promise for languages and domains for which text processing tools (e.g., taggers, parsers) are not readily available. Furthermore, to our knowledge, its performance on written text has not been assessed.

SHIFT transfers the first word from the input list onto the stack.
REDUCE pops the syntactic trees located at the top of the stack, combines them into a new tree and then pushes the new tree onto the top of the stack.
DROP deletes from the input list subsequences of words that correspond to a syntactic constituent.
ASSIGNTYPE changes the label of the trees at the top of the stack (i.e., the POS tag of words).

Table 1: Stack rewriting operations

2.1 Decision-based Sentence Compression

In the decision-based model, sentence compression is treated as a deterministic rewriting process of converting a long parse tree, l, into a shorter parse tree s. The rewriting process is decomposed into a sequence of shift-reduce-drop actions that follow an extended shift-reduce parsing paradigm.

The compression process starts with an empty stack and an input list that is built from the original sentence's parse tree. Words in the input list are labelled with the name of all the syntactic constituents in the original sentence that start with it. Each stage of the rewriting process is an operation that aims to reconstruct the compressed tree. There are four types of operations that can be performed on the stack, they are illustrated in Table 1.

Learning cases are automatically generated from a parallel corpus. Each learning case is expressed by a set of features and represents one of the four possible operations for a given stack and input list. Using the C4.5 program (Quinlan 1993) a decision-tree model is automatically learnt. The model is applied to a parsed original sentence in a deterministic fashion. Features for the current state of the input list and stack are extracted and the classifier is queried for the next operation to perform. This is repeated until the input list is empty and the stack contains only one item (this corresponds to the parse for the compressed tree). The compressed sentence is recovered by traversing the leaves of the tree in order.

2.2 Word-based Sentence Compression

The decision-based method relies exclusively on parallel corpora; the caveat here is that appropriate training data may be scarce when porting this model to different text domains (where abstracts

[1] The noisy-channel model uses a source model trained on uncompressed sentences. This means that the most likely compressed sentence will be identical to the original sentence as the likelihood of a constituent deletion is typically far lower than that of leaving it in.

are not available for automatic corpus creation) or languages. To alleviate the problems inherent with using a parallel corpus, we have modified a weakly supervised algorithm originally proposed by Hori and Furui (2004). Their method is based on word deletion; given a prespecified compression length, a compression is formed by preserving the words which maximise a scoring function.

To make Hori and Furui's (2004) algorithm more comparable to the decision-based model, we have eliminated the compression length parameter. Instead, we search over all lengths to find the compression that gives the maximum score. This process yields more natural compressions with varying lengths. The original score measures the significance of each word (I) in the compression and the linguistic likelihood (L) of the resulting word combinations.[2] We add some linguistic knowledge to this formulation through a function (SOV) that captures information about subjects, objects and verbs. The compression score is given in Equation (1). The lambdas (λ_I, λ_{SOV}, λ_L) weight the contribution of the individual scores:

$$\begin{aligned} S(V) &= \sum_{i=1}^{M} \lambda_I I(v_i) + \lambda_{sov} SOV(v_i) \\ &\quad + \lambda_L L(v_i|v_{i-1}, v_{i-2}) \end{aligned} \tag{1}$$

The sentence $V = v_1, v_2, \ldots, v_m$ (of M words) that maximises the score $S(V)$ is the best compression for an original sentence consisting of N words ($M < N$). The best compression can be found using dynamic programming. The λ's in Equation (1) can be either optimised using a small amount of training data or set manually (e.g., if short compressions are preferred to longer ones, then the language model should be given a higher weight). Alternatively, weighting could be dispensed with by including a normalising factor in the language model. Here, we follow Hori and Furui's (2004) original formulation and leave the normalisation to future work. We next introduce each measure individually.

Word significance score The word significance score I measures the relative importance of a word in a document. It is similar to tf-idf, a term weighting score commonly used in information retrieval:

$$I(w_i) = f_i \log \frac{F_A}{F_i} \tag{2}$$

Where w_i is the topic word of interest (topic words are either nouns or verbs), f_i is the frequency of w_i in the document, F_i is the corpus frequency of w_i and F_A is the sum of all topic word occurrences in the corpus ($\sum_i F_i$).

Linguistic score The linguistic score's $L(v_i|v_{i-1}, v_{i-2})$ responsibility is to select some function words, thus ensuring that compressions remain grammatical. It also controls which topic words can be placed together. The score measures the n-gram probability of the compressed sentence.

SOV Score The *SOV* score is based on the intuition that subjects, objects and verbs should not be dropped while words in other syntactic roles can be considered for removal. This score is based solely on the contents of the sentence considered for compression without taking into account the distribution of subjects, objects or verbs, across documents. It is defined in (3) where f_i is the document frequency of a verb, or word bearing the subject/object role and $\lambda_{default}$ is a constant weight assigned to all other words.

$$SOV(w_i) = \begin{cases} f_i & \text{if } w_i \text{ in subject, object} \\ & \text{or verb role} \\ \lambda_{default} & \text{otherwise} \end{cases} \tag{3}$$

The *SOV* score is only applied to the head word of subjects and objects.

3 Corpora

Our intent was to assess the performance of the two models just described on written and spoken text. The appeal of written text is understandable since most summarisation work today focuses on this domain. Speech data not only provides a natural test-bed for compression applications (e.g., subtitle generation) but also poses additional challenges. Spoken utterances can be ungrammatical, incomplete, and often contain artefacts such as false starts, interjections, hesitations, and disfluencies. Rather than focusing on spontaneous speech which is abundant in these artefacts, we conduct our study on the less ambitious domain of broadcast news transcripts. This lies in-between the extremes of written text and spontaneous speech as it has been scripted beforehand and is usually read off an autocue.

One stumbling block to performing a comparative study between written data and speech data is that there are no naturally occurring parallel

[2] Hori and Furui (2004) also have a confidence score based upon how reliable the output of an automatic speech recognition system is. However, we need not consider this score when working with written text and manual transcripts.

speech corpora for studying compression. Automatic corpus creation is not a viable option either, speakers do not normally create summaries of their own utterances. We thus gathered our own corpus by asking humans to generate compressions for speech transcripts.

In what follows we describe how the manual compressions were performed. We also briefly present the written corpus we used for our experiments. The latter was automatically constructed and offers an interesting point of comparison with our manually created corpus.

Broadcast News Corpus Three annotators were asked to compress 50 broadcast news stories (1,370 sentences) taken from the HUB-4 1996 English Broadcast News corpus provided by the LDC. The HUB-4 corpus contains broadcast news from a variety of networks (CNN, ABC, CSPAN and NPR) which have been manually transcribed and split at the story and sentence level. Each document contains 27 sentences on average and the whole corpus consists of 26,151 tokens.[3] The Robust Accurate Statistical Parsing (RASP) toolkit (Briscoe and Carroll 2002) was used to automatically tokenise the corpus.

Each annotator was asked to perform sentence compression by removing tokens from the original transcript. Annotators were asked to remove words while: (a) preserving the most important information in the original sentence, and (b) ensuring the compressed sentence remained grammatical. If they wished they could leave a sentence uncompressed by marking it as inappropriate for compression. They were not allowed to delete whole sentences even if they believed they contained no information content with respect to the story as this would blur the task with abstracting.

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. The most popular compression corpus originates from the Ziff-Davis corpus — a collection of news articles on computer products. The corpus was created by matching sentences that occur in an article with sentences that occur in an abstract (Knight and Marcu 2002). The abstract sentences had to contain a subset of the original sentence's words and the word order had to remain the same.

	A1	A2	A3	Av.	Ziff-Davis
Comp%	88.0	79.0	87.0	84.4	97.0
CompR	73.1	79.0	70.0	73.0	47.0

Table 2: Compression Rates (Comp% measures the percentage of sentences compressed; CompR is the mean compression rate of all sentences)

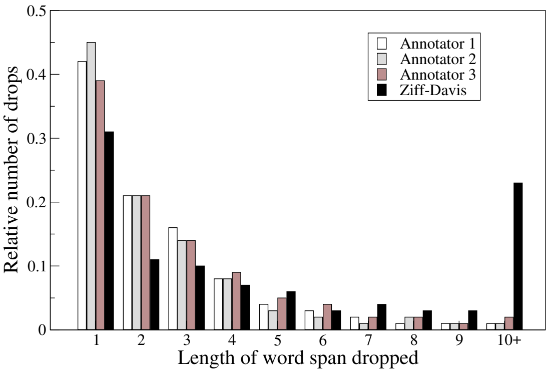


Figure 1: Distribution of span of words dropped

Comparisons Following the classification scheme adopted in the British National Corpus (Burnard 2000), we assume throughout this paper that Broadcast News and Ziff-Davis belong to different domains (spoken vs. written text) whereas they represent the same genre (i.e., news). Table 2 shows the percentage of sentences which were compressed (Comp%) and the mean compression rate (CompR) for the two corpora. The annotators compress the Broadcast News corpus to a similar degree. In contrast, the Ziff-Davis corpus is compressed much more aggressively with a compression rate of 47%, compared to 73% for Broadcast News. This suggests that the Ziff-Davis corpus may not be a true reflection of human compression performance and that humans tend to compress sentences more conservatively than the compressions found in abstracts.

We also examined whether the two corpora differ with regard to the length of word spans being removed. Figure 1 shows how frequently word spans of varying lengths are being dropped. As can be seen, a higher percentage of long spans (five or more words) are dropped in the Ziff-Davis corpus. This suggests that the annotators are removing words rather than syntactic constituents, which provides support for a model that can act on the word level. There is no statistically significant difference between the length of spans dropped between the annotators, whereas there is a significant difference ($p < 0.01$) between the annotators' spans and the Ziff-Davis' spans (using the

[3] The compression corpus is available at `http://homepages.inf.ed.ac.uk/s0460084/data/`.

Wilcoxon Test).

The compressions produced for the Broadcast News corpus may differ slightly to the Ziff-Davis corpus. Our annotators were asked to perform sentence compression explicitly as an isolated task rather than indirectly (and possibly subconsciously) as part of the broader task of abstracting, which we can assume is the case with the Ziff-Davis corpus.

4 Automatic Evaluation Measures

Previous studies relied almost exclusively on human judgements for assessing the well-formedness of automatically derived compressions. Although human evaluations of compression systems are not as large-scale as in other fields (e.g., machine translation), they are typically performed once, at the end of the development cycle. Automatic evaluation measures would allow more extensive parameter tuning and crucially experimentation with larger data sets. Most human studies to date are conducted on a small compression sample, the test portion of the Ziff-Davis corpus (32 sentences). Larger sample sizes would expectedly render human evaluations time consuming and generally more difficult to conduct frequently. Here, we review two automatic evaluation measures that hold promise for the compression task.

Simple String Accuracy (SSA, Bangalore et al. 2000) has been proposed as a baseline evaluation metric for natural language generation. It is based on the string edit distance between the generated output and a gold standard. It is a measure of the number of insertion (I), deletion (D) and substitution (S) errors between two strings. It is defined in (4) where R is the length of the gold standard string.

$$\text{Simple String Accuracy} = (1 - \frac{I+D+S}{R}) \quad (4)$$

The SSA score will assess whether appropriate words have been included in the compression.

Another stricter automatic evaluation method is to compare the grammatical relations found in the system compressions against those found in a gold standard. This allows us "to measure the semantic aspects of summarisation quality in terms of grammatical-functional information" (Riezler et al. 2003). The standard metrics of precision, recall and F-score can then be used to measure the quality of a system against a gold standard. Our implementation of the F-score measure used the grammatical relations annotations provided by RASP (Briscoe and Carroll 2002). This parser is particularly appropriate for the compression task since it provides parses for both full sentences and sentence fragments and is generally robust enough to analyse semi-grammatical compressions. We calculated F-score over all the relations provided by RASP (e.g., subject, direct/indirect object, modifier; 15 in total).

Correlation with human judgements is an important prerequisite for the wider use of automatic evaluation measures. In the following section we describe an evaluation study examining whether the measures just presented indeed correlate with human ratings of compression quality.

5 Experimental Set-up

In this section we present our experimental set-up for assessing the performance of the two algorithms discussed above. We explain how different model parameters were estimated. We also describe a judgement elicitation study on automatic and human-authored compressions.

Parameter Estimation We created two variants of the decision-tree model, one trained on the Ziff-Davis corpus and one on the Broadcast News corpus. We used 1,035 sentences from the Ziff-Davis corpus for training; the same sentences were previously used in related work (Knight and Marcu 2002). The second variant was trained on 1,237 sentences from the Broadcast News corpus. The training data for both models was parsed using Charniak's (2000) parser. Learning cases were automatically generated using a set of 90 features similar to Knight and Marcu (2002).

For the word-based method, we randomly selected 50 sentences from each training set to optimise the lambda weighting parameters[4]. Optimisation was performed using Powell's method (Press et al. 1992). Recall from Section 2.2 that the compression score has three main parameters: the significance, linguistic, and *SOV* scores. The significance score was calculated using 25 million tokens from the Broadcast News corpus (spoken variant) and 25 million tokens from the North American News Text Corpus (written variant). The linguistic score was estimated using a trigram language model. The language model was trained on the North Ameri-

[4] To treat both models on an equal footing, we attempted to train the decision-tree model solely on 50 sentences. However, it was unable to produce any reasonable compressions, presumably due to insufficient learning instances.

can corpus (25 million tokens) using the CMU-Cambridge Language Modeling Toolkit (Clarkson and Rosenfeld 1997) with a vocabulary size of 50,000 tokens and Good-Turing discounting. Subjects, objects, and verbs for the *SOV* score were obtained from RASP (Briscoe and Carroll 2002).

All our experiments were conducted on sentences for which we obtained syntactic analyses. RASP failed on 17 sentences from the Broadcast news corpus and 33 from the Ziff-Davis corpus; Charniak's (2000) parser successfully parsed the Broadcast News corpus but failed on three sentences from the Ziff-Davis corpus.

Evaluation Data We randomly selected 40 sentences for evaluation purposes, 20 from the testing portion of the Ziff-Davis corpus (32 sentences) and 20 sentences from the Broadcast News corpus (133 sentences were set aside for testing). This is comparable to previous studies which have used the 32 test sentences from the Ziff-Davis corpus. None of the 20 Broadcast News sentences were used for optimisation. We ran the decision-tree system and the word-based system on these 40 sentences. One annotator was randomly selected to act as the gold standard for the Broadcast News corpus; the gold standard for the Ziff-Davis corpus was the sentence that occurred in the abstract. For each original sentence we had three compressions; two generated automatically by our systems and a human authored gold standard. Thus, the total number of compressions was 120 (3x40).

Human Evaluation The 120 compressions were rated by human subjects. Their judgements were also used to examine whether the automatic evaluation measures discussed in Section 4 correlate reliably with behavioural data. Sixty unpaid volunteers participated in our elicitation study, all were self reported native English speakers. The study was conducted remotely over the Internet. Participants were presented with a set of instructions that explained the task and defined sentence compression with the aid of examples. They first read the original sentence with the compression hidden. Then the compression was revealed by pressing a button. Each participant saw 40 compressions. A Latin square design prevented subjects from seeing two different compressions of the same sentence. The order of the sentences was randomised. Participants were asked to rate each compression they saw on a five point scale taking into account the information retained by the compression and its grammaticality. They were told all compressions were automatically generated. Examples of the compressions our participants saw are given in Table 3.

o:	Apparently Fergie very much wants to have a career in television.
d:	A career in television.
w:	Fergie wants to have a career in television.
g:	Fergie wants a career in television.
o:	Many debugging features, including user-defined break points and variable-watching and message-watching windows, have been added.
d:	Many debugging features.
w:	Debugging features, and windows, have been added.
g:	Many debugging features have been added.
o:	As you said, the president has just left for a busy three days of speeches and fundraising in Nevada, California and New Mexico.
d:	As you said, the president has just left for a busy three days.
w:	You said, the president has left for three days of speeches and fundraising in Nevada, California and New Mexico.
g:	The president left for three days of speeches and fundraising in Nevada, California and New Mexico.

Table 3: Compression examples (o: original sentence, d: decision-tree compression, w: word-based compression, g: gold standard)

6 Results

Our experiments were designed to answer three questions: (1) Is there a significant difference between the compressions produced by supervised (constituent-based) and weakly unsupervised (word-based) approaches? (2) How well do the two models port across domains (written vs. spoken text) and corpora types (human vs. automatically created)? (3) Do automatic evaluation measures correlate with human judgements?

One of our first findings is that the the decision-tree model is rather sensitive to the style of training data. The model cannot capture and generalise single word drops as effectively as constituent drops. When the decision-tree is trained on the Broadcast News corpus, it is unable to create suitable compressions. On the evaluation data set, 75% of the compressions produced are the original sentence or the original sentence with one word removed. It is possible that the Broadcast News compression corpus contains more varied compressions than those of the Ziff-Davis and therefore a larger amount of training data would be required to learn a reliable decision-tree model. We thus used the Ziff-Davis trained decision-tree model to obtain compressions for both corpora.

Our results are summarised in Tables 4 and 5. Table 4 lists the average compression rates for

Broadcast News	CompR	SSA	F-score
Decision-tree	0.55	0.34	0.40
Word-based	0.72	0.51	0.54
gold standard	0.71	–	–

Ziff-Davis	CompR	SSA	F-score
Decision-tree	0.58	0.20	0.34
Word-based	0.60	0.19	0.39
gold standard	0.54	–	–

Table 4: Results using automatic evaluation measures

Compression	Broadcast News	Ziff-Davis
Decision-tree	2.04	2.34
Word-based	2.78	2.43
gold standard	3.87	3.53

Table 5: Mean ratings from human evaluation

each model as well as the models' performance according to the two automatic evaluation measures discussed in Section 4. The row 'gold standard' displays human-produced compression rates. Table 5 shows the results of our judgement elicitation study.

The compression rates (CompR, Table 4) indicate that the decision-tree model compresses more aggressively than the word-based model. This is due to the fact that it mostly removes entire constituents rather than individual words. The word-based model is closer to the human compression rate. According to our automatic evaluation measures, the decision-tree model is significantly worse than the word-based model (using the Student t test, SSA $p < 0.05$, F-score $p < 0.05$) on the Broadcast News corpus. Both models are significantly worse than humans (SSA $p < 0.05$, F-score $p < 0.01$). There is no significant difference between the two systems using the Ziff-Davis corpus on both simple string accuracy and relation F-score, whereas humans significantly outperform the two systems.

We have performed an Analysis of Variance (ANOVA) to examine whether similar results are obtained when using human judgements. Statistical tests were done using the mean of the ratings (see Table 5). The ANOVA revealed a reliable effect of compression type by subjects and by items ($p < 0.01$). Post-hoc Tukey tests confirmed that the word-based model outperforms the decision-tree model ($\alpha < 0.05$) on the Broadcast news corpus; however, the two models are not significantly different when using the Ziff-Davis corpus. Both systems perform significantly worse than the gold standard ($\alpha < 0.05$).

Measure	Ziff-Davis	Broadcast News
SSA	0.171	0.348*
F-score	0.575**	0.532**
$*p < 0.05$ $**p < 0.01$		

Table 6: Correlation (Pearson's r) between evaluation measures and human ratings. Stars indicate level of statistical significance.

We next examine the degree to which the automatic evaluation measures correlate with human ratings. Table 6 shows the results of correlating the simple string accuracy (SSA) and relation F-score against compression judgements. The SSA does not correlate on both corpora with human judgements; it thus seems to be an unreliable measure of compression performance. However, the F-score correlates significantly with human ratings, yielding a correlation coefficient of $r = 0.575$ on the Ziff-Davis corpus and $r = 0.532$ on the Broadcast news. To get a feeling for the difficulty of the task, we assessed how well our participants agreed in their ratings using leave-one-out resampling (Weiss and Kulikowski 1991). The technique correlates the ratings of each participant with the mean ratings of all the other participants. The average agreement is $r = 0.679$ on the Ziff-Davis corpus and $r = 0.746$ on the Broadcast News corpus. This result indicates that F-score's agreement with the human data is not far from the human upper bound.

7 Conclusions and Future Work

In this paper we have provided a comparison between a supervised (constituent-based) and a minimally supervised (word-based) approach to sentence compression. Our results demonstrate that the word-based model performs equally well on spoken and written text. Since it does not rely heavily on training data, it can be easily extended to languages or domains for which parallel compression corpora are scarce. When no parallel corpora are available the parameters can be manually tuned to produce compressions. In contrast, the supervised decision-tree model is not particularly robust on spoken text, it is sensitive to the nature of the training data, and did not produce adequate compressions when trained on the human-authored Broadcast News corpus. A comparison of the automatically gathered Ziff-Davis corpus with the Broadcast News corpus revealed important differences between the two corpora and thus suggests that automatically created corpora may not reflect human compression performance.

We have also assessed whether automatic evaluation measures can be used for the compression task. Our results show that grammatical relations-based F-score (Riezler et al. 2003) correlates reliably with human judgements and could thus be used to measure compression performance automatically. For example, it could be used to assess progress during system development or for comparing across different systems and system configurations with much larger test sets than currently employed.

In its current formulation, the only function driving compression in the word-based model is the language model. The word significance and *SOV* scores are designed to single out important words that the model should not drop. We have not yet considered any functions that encourage compression. Ideally these functions should be inspired from the underlying compression process. Finding such a mechanism is an avenue of future work. We would also like to enhance the word-based model with more linguistic knowledge; we plan to experiment with syntax-based language models and more richly annotated corpora.

Another important future direction lies in applying the unsupervised model presented here to languages with more flexible word order and richer morphology than English (e.g., German, Czech). We suspect that these languages will prove challenging for creating grammatically acceptable compressions. Finally, our automatic evaluation experiments motivate the use of relations-based F-score as a means of directly optimising compression quality, much in the same way MT systems optimise model parameters using BLEU as a measure of translation quality.

Acknowledgements

We are grateful to our annotators Vasilis Karaiskos, Beata Kouchnir, and Sarah Luger. Thanks to Jean Carletta, Frank Keller, Steve Renals, and Sebastian Riedel for helpful comments and suggestions. Lapata acknowledges the support of EPSRC (grant GR/T04540/01).

References

Bangalore, Srinivas, Owen Rambow, and Steve Whittaker. 2000. Evaluation metrics for generation. In *Proceedings of the 1st INLG*. Mitzpe Ramon, Israel, pages 1–8.

Briscoe, E. J. and J. Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd LREC*. Las Palmas, Spain, pages 1499–1504.

Burnard, Lou. 2000. *The Users Reference Guide for the British National Corpus (World Edition)*. British National Corpus Consortium, Oxford University Computing Service.

Charniak, Eugene. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st NAACL*. San Francisco, CA, pages 132–139.

Clarkson, Philip and Ronald Rosenfeld. 1997. Statistical language modeling using the CMU–cambridge toolkit. In *Proceedings of Eurospeech*. Rhodes, Greece, pages 2707–2710.

Corston-Oliver, Simon. 2001. Text Compaction for Display on Very Small Screens. In *Proceedings of the NAACL Workshop on Automatic Summarization*. Pittsburgh, PA, pages 89–98.

Grefenstette, Gregory. 1998. Producing Intelligent Telegraphic Text Reduction to Provide an Audio Scanning Service for the Blind. In *Proceedings of the AAAI Symposium on Intelligent Text Summarization*. Stanford, CA, pages 111–117.

Hori, Chiori and Sadaoki Furui. 2004. Speech summarization: an approach through word extraction and a method for evaluation. *IEICE Transactions on Information and Systems* E87-D(1):15–25.

Jing, Hongyan. 2000. Sentence Reduction for Automatic Text Summarization. In *Proceedings of the 6th ANLP*. Seattle,WA, pages 310–315.

Knight, Kevin and Daniel Marcu. 2002. Summarization beyond sentence extraction: a probabilistic approach to sentence compression. *Artificial Intelligence* 139(1):91–107.

McDonald, Ryan. 2006. Discriminative sentence compression with soft syntactic constraints. In *Proceedings of the 11th EACL*. Trento, Italy, pages 297–304.

Nguyen, Minh Le, Susumu Horiguchi, Akira Shimazu, and Bao Tu Ho. 2004a. Example-based sentence reduction using the hidden Markov model. *ACM TALIP* 3(2):146–158.

Nguyen, Minh Le, Akira Shimazu, Susumu Horiguchi, Tu Bao Ho, and Masaru Fukushi. 2004b. Probabilistic sentence reduction using support vector machines. In *Proceedings of the 20th COLING*. Geneva, Switzerland, pages 743–749.

Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA.

Quinlan, J. R. 1993. *C4.5 – Programs for Machine Learning*. The Morgan Kaufmann series in machine learning. Morgan Kaufman Publishers.

Riezler, Stefan, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of the HLT/NAACL*. Edmonton, Canada, pages 118–125.

Turner, Jenine and Eugene Charniak. 2005. Supervised and unsupervised learning for sentence compression. In *Proceedings of the 43rd ACL*. Ann Arbor, MI, pages 290–297.

Vandeghinste, Vincent and Yi Pan. 2004. Sentence compression for automated subtitling: A hybrid approach. In *Proceedings of the ACL Workshop on Text Summarization*. Barcelona, Spain, pages 89–95.

Weiss, Sholom M. and Casimir A. Kulikowski. 1991. *Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

A Bottom-up Approach to Sentence Ordering for Multi-document Summarization

Danushka Bollegala **Naoaki Okazaki** * **Mitsuru Ishizuka**
Graduate School of Information Science and Technology
The University of Tokyo
7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan
{danushka,okazaki}@mi.ci.i.u-tokyo.ac.jp
ishizuka@i.u-tokyo.ac.jp

Abstract

Ordering information is a difficult but important task for applications generating natural-language text. We present a bottom-up approach to arranging sentences extracted for multi-document summarization. To capture the association and order of two textual segments (eg, sentences), we define four criteria, *chronology*, *topical-closeness*, *precedence*, and *succession*. These criteria are integrated into a criterion by a supervised learning approach. We repeatedly concatenate two textual segments into one segment based on the criterion until we obtain the overall segment with all sentences arranged. Our experimental results show a significant improvement over existing sentence ordering strategies.

1 Introduction

Multi-document summarization (MDS) (Radev and McKeown, 1999) tackles the information overload problem by providing a condensed version of a set of documents. Among a number of sub-tasks involved in MDS, eg, sentence extraction, topic detection, sentence ordering, information extraction, sentence generation, etc., most MDS systems have been based on an extraction method, which identifies important textual segments (eg, sentences or paragraphs) in source documents. It is important for such MDS systems to determine a coherent arrangement of the textual segments extracted from multi-documents in order to reconstruct the text structure for summarization. Ordering information is also essential for other text-generation applications such as Question Answering.

A summary with improperly ordered sentences confuses the reader and degrades the quality/reliability of the summary itself. Barzilay (2002) has provided empirical evidence that proper order of extracted sentences improves their readability significantly. However, ordering a set of sentences into a coherent text is a nontrivial task. For example, identifying rhetorical relations (Mann and Thompson, 1988) in an ordered text has been a difficult task for computers, whereas our task is even more complicated: to reconstruct such relations from unordered sets of sentences. Source documents for a summary may have been written by different authors, by different writing styles, on different dates, and based on different background knowledge. We cannot expect that a set of extracted sentences from such diverse documents will be coherent on their own.

Several strategies to determine sentence ordering have been proposed as described in section 2. However, the appropriate way to combine these strategies to achieve more coherent summaries remains unsolved. In this paper, we propose four criteria to capture the association of sentences in the context of multi-document summarization for newspaper articles. These criteria are integrated into one criterion by a supervised learning approach. We also propose a bottom-up approach in arranging sentences, which repeatedly concatenates textual segments until the overall segment with all sentences arranged, is achieved.

2 Related Work

Existing methods for sentence ordering are divided into two approaches: making use of chronological information (McKeown et al., 1999; Lin

*Research Fellow of the Japan Society for the Promotion of Science (JSPS)

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 385–392,
Sydney, July 2006. ©2006 Association for Computational Linguistics

and Hovy, 2001; Barzilay et al., 2002; Okazaki et al., 2004); and learning the natural order of sentences from large corpora not necessarily based on chronological information (Lapata, 2003; Barzilay and Lee, 2004). A newspaper usually disseminates descriptions of novel events that have occurred since the last publication. For this reason, ordering sentences according to their publication date is an effective heuristic for multidocument summarization (Lin and Hovy, 2001; McKeown et al., 1999). Barzilay et al. (2002) have proposed an improved version of chronological ordering by first grouping sentences into sub-topics discussed in the source documents and then arranging the sentences in each group chronologically.

Okazaki et al. (2004) have proposed an algorithm to improve chronological ordering by resolving the presuppositional information of extracted sentences. They assume that each sentence in newspaper articles is written on the basis that presuppositional information should be transferred to the reader before the sentence is interpreted. The proposed algorithm first arranges sentences in a chronological order and then estimates the presuppositional information for each sentence by using the content of the sentences placed before each sentence in its original article. The evaluation results show that the proposed algorithm improves the chronological ordering significantly.

Lapata (2003) has suggested a probabilistic model of text structuring and its application to the sentence ordering. Her method calculates the transition probability from one sentence to the next from a corpus based on the Cartesian product between two sentences defined using the following features: verbs (precedent relationships of verbs in the corpus); nouns (entity-based coherence by keeping track of the nouns); and dependencies (structure of sentences). Although she has not compared her method with chronological ordering, it could be applied to generic domains, not relying on the chronological clue provided by newspaper articles.

Barzilay and Lee (2004) have proposed *content models* to deal with topic transition in domain specific text. The content models are formalized by Hidden Markov Models (HMMs) in which the hidden state corresponds to a topic in the domain of interest (eg, earthquake magnitude or previous earthquake occurrences), and the state transitions capture possible information-presentation orderings. The evaluation results showed that their method outperformed Lapata's approach by a wide margin. They did not compare their method with chronological ordering as an application of multi-document summarization.

As described above, several good strategies/heuristics to deal with the sentence ordering problem have been proposed. In order to integrate multiple strategies/heuristics, we have formalized them in a machine learning framework and have considered an algorithm to arrange sentences using the integrated strategy.

3 Method

We define notation $a \succ b$ to represent that sentence a precedes sentence b. We use the term *segment* to describe a sequence of ordered sentences. When segment A consists of sentences a_1, a_2, ..., a_m in this order, we denote as:

$$A = (a_1 \succ a_2 \succ ... \succ a_m). \qquad (1)$$

The two segments A and B can be ordered either B after A or A after B. We define the notation $A \succ B$ to show that segment A precedes segment B.

Let us consider a bottom-up approach in arranging sentences. Starting with a set of segments initialized with a sentence for each, we concatenate two segments, with the strongest association (discussed later) of all possible segment pairs, into one segment. Repeating the concatenating will eventually yield a segment with all sentences arranged. The algorithm is considered as a variation of agglomerative hierarchical clustering with the ordering information retained at each concatenating process.

The underlying idea of the algorithm, a bottom-up approach to text planning, was proposed by Marcu (1997). Assuming that the semantic units (sentences) and their rhetorical relations (eg, sentence *a* is an *elaboration* of sentence *d*) are given, he transcribed a text structuring task into the problem of finding the best discourse tree that satisfied the set of rhetorical relations. He stated that global coherence could be achieved by satisfying local coherence constraints in ordering and clustering, thereby ensuring that the resultant discourse tree was well-formed.

Unfortunately, identifying the rhetorical relation between two sentences has been a difficult

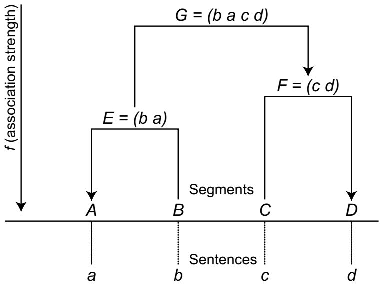


Figure 1: Arranging four sentences A, B, C, and D with a bottom-up approach.

task for computers. However, the bottom-up algorithm for arranging sentences can still be applied only if the direction and strength of the association of the two segments (sentences) are defined. Hence, we introduce a function $f(A \succ B)$ to represent the direction and strength of the association of two segments A and B,

$$f(A \succ B) = \begin{cases} p & \text{(if } A \text{ precedes } B\text{)} \\ 0 & \text{(if } B \text{ precedes } A\text{)} \end{cases}, \qquad (2)$$

where p $(0 \leq p \leq 1)$ denotes the association strength of the segments A and B. The association strengths of the two segments with different directions, eg, $f(A \succ B)$ and $f(B \succ A)$, are not always identical in our definition,

$$f(A \succ B) \neq f(B \succ A). \qquad (3)$$

Figure 1 shows the process of arranging four sentences a, b, c, and d. Firstly, we initialize four segments with a sentence for each,

$$A = (a), B = (b), C = (c), D = (d). \qquad (4)$$

Suppose that $f(B \succ A)$ has the highest value of all possible pairs, eg, $f(A \succ B)$, $f(C \succ D)$, etc, we concatenate B and A to obtain a new segment,

$$E = (b \succ a). \qquad (5)$$

Then we search for the segment pair with the strongest association. Supposing that $f(C \succ D)$ has the highest value, we concatenate C and D to obtain a new segment,

$$F = (c \succ d). \qquad (6)$$

Finally, comparing $f(E \succ F)$ and $f(F \succ E)$, we obtain the global sentence ordering,

$$G = (b \succ a \succ c \succ d). \qquad (7)$$

In the above description, we have not defined the association of the two segments. The previous work described in Section 2 has addressed the association of textual segments (sentences) to obtain coherent orderings. We define four criteria to capture the association of two segments: *chronology*; *topical-closeness*; *precedence*; and *succession*. These criteria are integrated into a function $f(A \succ B)$ by using a machine learning approach. The rest of this section explains the four criteria and an integration method with a Support Vector Machine (SVM) (Vapnik, 1998) classifier.

3.1 Chronology criterion

Chronology criterion reflects the chronological ordering (Lin and Hovy, 2001; McKeown et al., 1999), which arranges sentences in a chronological order of the publication date. We define the association strength of arranging segments B after A measured by a chronology criterion $f_{\text{chro}}(A \succ B)$ in the following formula,

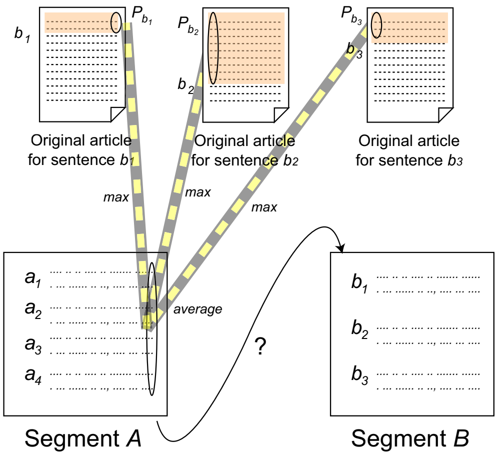
$$\begin{aligned} & f_{\text{chro}}(A \succ B) \\ = & \begin{cases} 1 & \mathrm{T}(a_m) < \mathrm{T}(b_1) \\ 1 & [\mathrm{D}(a_m) = \mathrm{D}(b_1)] \wedge [\mathrm{N}(a_m) < \mathrm{N}(b_1)] \\ 0.5 & [\mathrm{T}(a_m) = \mathrm{T}(b_1)] \wedge [\mathrm{D}(a_m) \neq \mathrm{D}(b_1)] \\ 0 & \text{otherwise} \end{cases}. \end{aligned} \qquad (8)$$

Here, a_m represents the last sentence in segment A; b_1 represents the first sentence in segment B; $T(s)$ is the publication date of the sentence s; $D(s)$ is the unique identifier of the document to which sentence s belongs: and $N(s)$ denotes the line number of sentence s in the original document. The chronological order of arranging segment B after A is determined by the comparison between the last sentence in the segment A and the first sentence in the segment B.

The chronology criterion assesses the appropriateness of arranging segment B after A if: sentence a_m is published earlier than b_1; or sentence a_m appears before b_1 in the same article. If sentence a_m and b_1 are published on the same day but appear in different articles, the criterion assumes the order to be undefined. If none of the above conditions are satisfied, the criterion estimates that segment B will precede A.

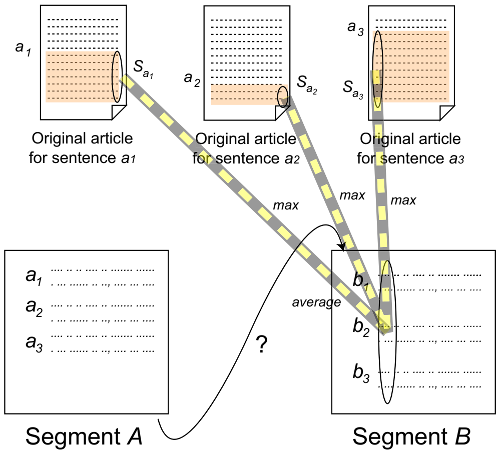
3.2 Topical-closeness criterion

The topical-closeness criterion deals with the association, based on the topical similarity, of two



b1
Pb1
Pb2
Pb3
b2
b3
Original article for sentence b1
Original article for sentence b2
Original article for sentence b3
max
max
max
average
a1
a2
a3
a4
?
b1
b2
b3
Segment A
Segment B

Figure 2: Precedence criterion



a1
Sa1
a2
Sa2
a3
Sa3
Original article for sentence a1
Original article for sentence a2
Original article for sentence a3
max
max
max
average
a1
a2
a3
?
b1
b2
b3
Segment A
Segment B

Figure 3: Succession criterion

segments. The criterion reflects the ordering strategy proposed by Barzilay et al (2002), which groups sentences referring to the same topic. To measure the topical closeness of two sentences, we represent each sentence with a vector whose elements correspond to the occurrence[1] of the nouns and verbs in the sentence. We define the topical closeness of two segments A and B as follows,

$$f_{\text{topic}}(A \succ B) = \frac{1}{|B|} \sum_{b \in B} \max_{a \in A} \text{sim}(a, b). \quad (9)$$

Here, $\text{sim}(a, b)$ denotes the similarity of sentences a and b, which is calculated by the cosine similarity of two vectors corresponding to the sentences. For sentence $b \in B$, $\max_{a \in A} \text{sim}(a, b)$ chooses the sentence $a \in A$ most similar to sentence b and yields the similarity. The topical-closeness criterion $f_{\text{topic}}(A \succ B)$ assigns a higher value when the topic referred by segment B is the same as segment A.

3.3 Precedence criterion

Let us think of the case where we arrange segment A before B. Each sentence in segment B has the presuppositional information that should be conveyed to a reader in advance. Given sentence $b \in B$, such presuppositional information may be presented by the sentences appearing before the sentence b in the original article. However, we cannot guarantee whether a sentence-extraction method for multi-document summarization chooses any sentences before b for a summary because the extraction method usually determines a set of sentences, within the constraint of summary length, that maximizes information coverage and excludes redundant information. *Precedence criterion* measures the substitutability of the presuppositional information of segment B (eg, the sentences appearing before sentence b) as segment A. This criterion is a formalization of the sentence-ordering algorithm proposed by Okazaki et al, (2004).

We define the precedence criterion in the following formula,

$$f_{\text{pre}}(A \succ B) = \frac{1}{|B|} \sum_{b \in B} \max_{a \in A, p \in P_b} \text{sim}(a, p). \quad (10)$$

Here, P_b is a set of sentences appearing before sentence b in the original article; and $\text{sim}(a, b)$ denotes the cosine similarity of sentences a and b (defined as in the topical-closeness criterion). Figure 2 shows an example of calculating the precedence criterion for arranging segment B after A. We approximate the presuppositional information for sentence b by sentences P_b, ie, sentences appearing before the sentence b in the original article. Calculating the similarity among sentences in P_b and A by the maximum similarity of the possible sentence combinations, Formula 10 is interpreted as the average similarity of the precedent sentences $\forall P_b (b \in B)$ to the segment A.

3.4 Succession criterion

The idea of *succession criterion* is the exact opposite of the precedence criterion. The succession criterion assesses the coverage of the succedent information for segment A by arranging segment B

[1] The vector values are represented by boolean values, i.e., 1 if the sentence contains a word, otherwise 0.

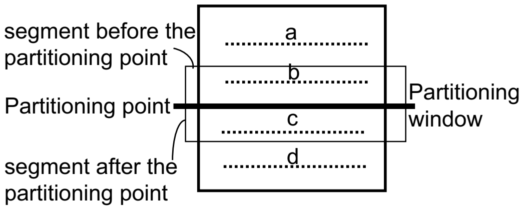

segment before the partitioning point
a
b
Partitioning point
c
d
segment after the partitioning point
Partitioning window

Figure 4: Partitioning a human-ordered extract into pairs of segments

after A:

$$f_{\rm succ}(A \succ B) = \frac{1}{|A|}\sum_{a\in A}\max_{s\in S_a, b\in B} {\rm sim}(s,b). \tag{11}$$

Here, S_a is a set of sentences appearing after sentence a in the original article; and ${\rm sim}(a,b)$ denotes the cosine similarity of sentences a and b (defined as in the topical-closeness criterion). Figure 3 shows an example of calculating the succession criterion to arrange segments B after A. The succession criterion measures the substitutability of the succedent information (eg, the sentences appearing after the sentence $a \in A$) as segment B.

3.5 SVM classifier to assess the integrated criterion

We integrate the four criteria described above to define the function $f(A \succ B)$ to represent the association direction and strength of the two segments A and B (Formula 2). More specifically, given the two segments A and B, function $f(A \succ B)$ is defined to yield the integrated association strength from four values, $f_{\rm chro}(A \succ B)$, $f_{\rm topic}(A \succ B)$, $f_{\rm pre}(A \succ B)$, and $f_{\rm succ}(A \succ B)$. We formalize the integration task as a binary classification problem and employ a Support Vector Machine (SVM) as the classifier. We conducted a supervised learning as follows.

We partition a human-ordered extract into pairs each of which consists of two non-overlapping segments. Let us explain the partitioning process taking four human-ordered sentences, $a \succ b \succ c \succ d$ shown in Figure 4. Firstly, we place the partitioning point just after the first sentence a. Focusing on sentence a arranged just before the partition point and sentence b arranged just after we identify the pair $\{(a),(b)\}$ of two segments (a) and (b). Enumerating all possible pairs of two segments facing just before/after the partitioning point, we obtain the following pairs, $\{(a),(b \succ c)\}$ and $\{(a),(b \succ c \succ d)\}$. Similarly, segment pairs, $\{(b),(c)\}$, $\{(a \succ b),(c)\}$, $\{(b),(c \succ d)\}$, $\{(a \succ b),(c \succ d)\}$, are obtained from the partitioning point between sentence b and c. Collecting the segment pairs from the partitioning point between sentences c and d (i.e., $\{(c),(d)\}$, $\{(b \succ c),(d)\}$ and $\{(a \succ b \succ c),(d)\}$), we identify ten pairs in total form the four ordered sentences. In general, this process yields $n(n^2-1)/6$ pairs from ordered n sentences. From each pair of segments, we generate one positive and one negative training instance as follows.

$$\begin{array}{l} +1 : [f_{\rm chro}(A \succ B), f_{\rm topic}(A \succ B), f_{\rm pre}(A \succ B), f_{\rm succ}(A \succ B)] \\ -1 : [f_{\rm chro}(B \succ A), f_{\rm topic}(B \succ A), f_{\rm pre}(B \succ A), f_{\rm succ}(B \succ A)] \end{array}$$

Figure 5: Two vectors in a training data generated from two ordered segments $A \succ B$

Given a pair of two segments A and B arranged in an order $A \succ B$, we calculate four values, $f_{\rm chro}(A \succ B)$, $f_{\rm topic}(A \succ B)$, $f_{\rm pre}(A \succ B)$, and $f_{\rm succ}(A \succ B)$ to obtain the instance with the four-dimensional vector (Figure 5). We label the instance (corresponding to $A \succ B$) as a positive class (ie, $+1$). Simultaneously, we obtain another instance with a four-dimensional vector corresponding to $B \succ A$. We label it as a negative class (ie, -1). Accumulating these instances as training data, we obtain a binary classifier by using a Support Vector Machine with a quadratic kernel. The SVM classifier yields the association direction of two segments (eg, $A \succ B$ or $B \succ A$) with the class information (ie, $+1$ or -1). We assign the association strength of two segments by using the class probability estimate that the instance belongs to a positive ($+1$) class. When an instance is classified into a negative (-1) class, we set the association strength as zero (see the definition of Formula 2).

4 Evaluation

We evaluated the proposed method by using the 3rd Text Summarization Challenge (TSC-3) corpus[2]. The TSC-3 corpus contains 30 sets of extracts, each of which consists of unordered sentences[3] extracted from Japanese newspaper articles relevant to a topic (query). We arrange the extracts by using different algorithms and evaluate

[2] http://lr-www.pi.titech.ac.jp/tsc/tsc3-en.html

[3] Each extract consists of ca. 15 sentences on average.

Table 1: Correlation between two sets of human-ordered extracts

Metric	Mean	Std. Dev	Min	Max
Spearman	0.739	0.304	-0.2	1
Kendall	0.694	0.290	0	1
Average Continuity	0.401	0.404	0.001	1

the readability of the ordered extracts by a subjective grading and several metrics.

In order to construct training data applicable to the proposed method, we asked two human subjects to arrange the extracts and obtained $30(\text{topics}) \times 2(\text{humans}) = 60$ sets of ordered extracts. Table 1 shows the agreement of the ordered extracts between the two subjects. The correlation is measured by three metrics, Spearman's rank correlation, Kendall's rank correlation, and average continuity (described later). The mean correlation values (0.74 for Spearman's rank correlation and 0.69 for Kendall's rank correlation) indicate a certain level of agreement in sentence orderings made by the two subjects. 8 out of 30 extracts were actually identical.

We applied the leave-one-out method to the proposed method to produce a set of sentence orderings. In this experiment, the leave-out-out method arranges an extract by using an SVM model trained from the rest of the 29 extracts. Repeating this process 30 times with a different topic for each iteration, we generated a set of 30 extracts for evaluation. In addition to the proposed method, we prepared six sets of sentence orderings produced by different algorithms for comparison. We describe briefly the seven algorithms (including the proposed method):

Agglomerative ordering (AGL) is an ordering arranged by the proposed method;

Random ordering (RND) is the lowest anchor, in which sentences are arranged randomly;

Human-made ordering (HUM) is the highest anchor, in which sentences are arranged by a human subject;

Chronological ordering (CHR) arranges sentences with the chronology criterion defined in Formula 8. Sentences are arranged in chronological order of their publication date;

Topical-closeness ordering (TOP) arranges sentences with the topical-closeness criterion defined in Formula 9;

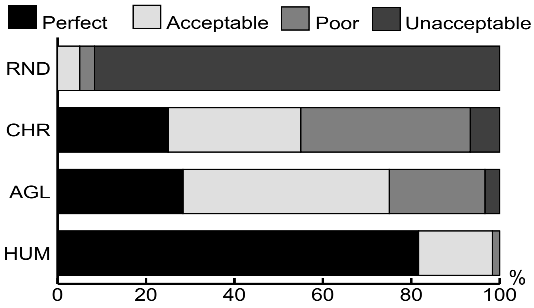


Figure 6: Subjective grading

Precedence ordering (PRE) arranges sentences with the precedence criterion defined in Formula 10;

Suceedence ordering (SUC) arranges sentences with the succession criterion defined in Formula 11.

The last four algorithms (CHR, TOP, PRE, and SUC) arrange sentences by the corresponding criterion alone, each of which uses the association strength directly to arrange sentences without the integration of other criteria. These orderings are expected to show the performance of each expert independently and their contribution to solving the sentence ordering problem.

4.1 Subjective grading

Evaluating a sentence ordering is a challenging task. Intrinsic evaluation that involves human judges to rank a set of sentence orderings is a necessary approach to this task (Barzilay et al., 2002; Okazaki et al., 2004). We asked two human judges to rate sentence orderings according to the following criteria. A *perfect* summary is a text that we cannot improve any further by re-ordering. An *acceptable* summary is one that makes sense and is unnecessary to revise even though there is some room for improvement in terms of readability. A *poor* summary is one that loses a thread of the story at some places and requires minor amendment to bring it up to an acceptable level. An *unacceptable* summary is one that leaves much to be improved and requires overall restructuring rather than partial revision. To avoid any disturbance in rating, we inform the judges that the summaries were made from a same set of extracted sentences and only the ordering of sentences is different.

Figure 6 shows the distribution of the subjective grading made by two judges to four sets of orderings, RND, CHR, AGL and HUM. Each set of or-

$$\begin{aligned} T_{eval} &= (e \succ a \succ b \succ c \succ d) \\ T_{ref} &= (a \succ b \succ c \succ d \succ e) \end{aligned}$$

Figure 7: An example of an ordering under evaluation T_{eval} and its reference T_{ref}.

derings has 30(topics) $\times$ 2(judges) $= 60$ ratings. Most RND orderings are rated as *unacceptable*. Although CHR and AGL orderings have roughly the same number of *perfect* orderings (ca. 25%), the AGL algorithm gained more *acceptable* orderings (47%) than the CHR alghrotihm (30%). This fact shows that integration of CHR experts with other experts worked well by pushing poor ordering to an acceptable level. However, a huge gap between *AGL* and *HUM* orderings was also found. The judges rated 28% AGL orderings as *perfect* while the figure rose as high as 82% for HUM orderings. Kendall's coefficient of concordance (Kendall's W), which asses the inter-judge agreement of overall ratings, reported a higher agreement between the two judges ($W = 0.939$).

4.2 Metrics for semi-automatic evaluation

We also evaluated sentence orderings by reusing two sets of gold-standard orderings made for the training data. In general, subjective grading consumes much time and effort, even though we cannot reproduce the evaluation afterwards. The previous studies (Barzilay et al., 2002; Lapata, 2003) employ rank correlation coefficients such as Spearman's rank correlation and Kendall's rank correlation, assuming a sentence ordering to be a rank. Okazaki et al. (2004) propose a metric that assess continuity of pairwise sentences compared with the gold standard. In addition to Spearman's and Kendall's rank correlation coefficients, we propose an *average continuity* metric, which extends the idea of the continuity metric to continuous k sentences.

A text with sentences arranged in proper order does not interrupt a human's reading while moving from one sentence to the next. Hence, the quality of a sentence ordering can be estimated by the number of continuous sentences that are also reproduced in the reference sentence ordering. This is equivalent to measuring a precision of continuous sentences in an ordering against the reference ordering. We define P_n to measure the precision of n continuous sentences in an ordering to be evaluated as,

$$P_n = \frac{m}{N - n + 1}. \qquad (12)$$

Here, N is the number of sentences in the reference ordering; n is the length of continuous sentences on which we are evaluating; m is the number of continuous sentences that appear in both the evaluation and reference orderings. In Figure 7, the precision of 3 continuous sentences P_3 is calculated as:

$$P_3 = \frac{2}{5 - 3 + 1} = 0.67. \qquad (13)$$

The Average Continuity (AC) is defined as the logarithmic average of P_n over 2 to k:

$$\text{AC} = \exp\left(\frac{1}{k-1}\sum_{n=2}^{k}\log(P_n + \alpha)\right). \qquad (14)$$

Here, k is a parameter to control the range of the logarithmic average; and α is a small value in case if P_n is zero. We set $k = 4$ (ie, more than five continuous sentences are not included for evaluation) and $\alpha = 0.01$. Average Continuity becomes 0 when evaluation and reference orderings share no continuous sentences and 1 when the two orderings are identical. In Figure 7, Average Continuity is calculated as 0.63. The underlying idea of Formula 14 was proposed by Papineni et al. (2002) as the BLEU metric for the semi-automatic evaluation of machine-translation systems. The original definition of the BLEU metric is to compare a machine-translated text with its reference translation by using the word n-grams.

Table 2: Comparison with human-made ordering

Method	Spearman coefficient	Kendall coefficient	Average Continuity
RND	-0.127	-0.069	0.011
TOP	0.414	0.400	0.197
PRE	0.415	0.428	0.293
SUC	0.473	0.476	0.291
CHR	0.583	0.587	0.356
AGL	0.603	0.612	0.459

4.3 Results of semi-automatic evaluation

Table 2 reports the resemblance of orderings produced by six algorithms to the human-made ones with three metrics, Spearman's rank correlation, Kendall's rank correlation, and Average Continuity. The proposed method (AGL) outperforms the

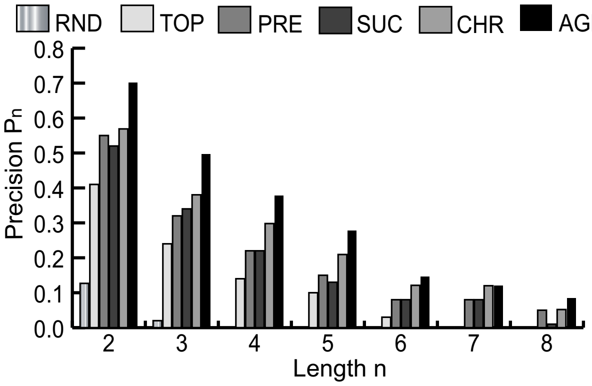


Figure 8: Precision vs unit of measuring continuity.

rest in all evaluation metrics, although the chronological ordering (CHR) appeared to play the major role. The one-way analysis of variance (ANOVA) verified the effects of different algorithms for sentence orderings with all metrics ($p < 0.01$). We performed Tukey Honest Significant Differences (HSD) test to compare differences among these algorithms. The Tukey test revealed that AGL was significantly better than the rest. Even though we could not compare our experiment with the probabilistic approach (Lapata, 2003) directly due to the difference of the text corpora, the Kendall coefficient reported higher agreement than Lapata's experiment (Kendall=0.48 with lemmatized nouns and Kendall=0.56 with verb-noun dependencies).

Figure 8 shows precision P_n with different length values of continuous sentence n for the six methods compared in Table 2. The number of continuous sentences becomes sparse for a higher value of length n. Therefore, the precision values decrease as the length n increases. Although RND ordering reported some continuous sentences for lower n values, no continuous sentences could be observed for the higher n values. Four criteria described in Section 3 (ie, CHR, TOP, PRE, SUC) produce segments of continuous sentences at all values of n.

5 Conclusion

We present a bottom-up approach to arrange sentences extracted for multi-document summarization. Our experimental results showed a significant improvement over existing sentence ordering strategies. However, the results also implied that chronological ordering played the major role in arranging sentences. A future direction of this study would be to explore the application of the proposed framework to more generic texts, such as documents without chronological information.

Acknowledgment

We used Mainichi Shinbun and Yomiuri Shinbun newspaper articles, and the TSC-3 test collection.

References

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *HLT-NAACL 2004: Proceedings of the Main Conference*, pages 113–120.

Regina Barzilay, Noemie Elhadad, and Kathleen McKeown. 2002. Inferring strategies for sentence ordering in multidocument news summarization. *Journal of Artificial Intelligence Research*, 17:35–55.

Mirella Lapata. 2003. Probabilistic text structuring: Experiments with sentence ordering. *Proceedings of the annual meeting of ACL, 2003.*, pages 545–552.

C.Y. Lin and E. Hovy. 2001. Neats:a multidocument summarizer. *Proceedings of the Document Understanding Workshop(DUC).*

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

Daniel Marcu. 1997. From local to global coherence: A bottom-up approach to text planning. In *Proceedings of the 14th National Conference on Artificial Intelligence*, pages 629–635, Providence, Rhode Island.

Kathleen McKeown, Judith Klavans, Vasileios Hatzivassiloglou, Regina Barzilay, and Eleazar Eskin. 1999. Towards multidocument summarization by reformulation: Progress and prospects. *AAAI/IAAI*, pages 453–460.

Naoaki Okazaki, Yutaka Matsuo, and Mitsuru Ishizuka. 2004. Improving chronological sentence ordering by precedence relation. In *Proceedings of 20th International Conference on Computational Linguistics (COLING 04)*, pages 750–756.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu:a method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.

Dragomir R. Radev and Kathy McKeown. 1999. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24:469–500.

V. Vapnik. 1998. *Statistical Learning Theory*. Wiley, Chichester, GB.

Learning Event Durations from Event Descriptions

Feng Pan, Rutu Mulkar, and Jerry R. Hobbs
Information Sciences Institute (ISI), University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292, USA
{pan, rutu, hobbs}@isi.edu

Abstract

We have constructed a corpus of news articles in which events are annotated for estimated bounds on their duration. Here we describe a method for measuring inter-annotator agreement for these event duration distributions. We then show that machine learning techniques applied to this data yield *coarse-grained* event duration information, considerably outperforming a baseline and approaching human performance.

1 Introduction

Consider the sentence from a news article:

George W. Bush met with Vladimir Putin in Moscow.

How long was the meeting? Our first reaction to this question might be that we have no idea. But in fact we do have an idea. We know the meeting was longer than 10 seconds and less than a year. How much tighter can we get the bounds to be? Most people would say the meeting lasted between an hour and three days.

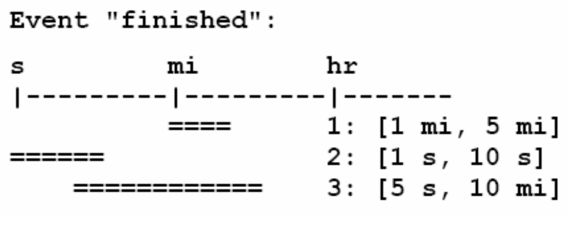
There is much temporal information in text that has hitherto been largely unexploited, encoded in the descriptions of events and relying on our knowledge of the range of usual durations of types of events. This paper describes one part of an exploration into how this information can be captured automatically. Specifically, we have developed annotation guidelines to minimize discrepant judgments and annotated 58 articles, comprising 2288 events; we have developed a method for measuring inter-annotator agreement when the judgments are intervals on a scale; and we have shown that machine learning techniques applied to the annotated data considerably outperform a baseline and approach human performance.

This research is potentially very important in applications in which the time course of events is to be extracted from news. For example, whether two events overlap or are in sequence often depends very much on their durations. If a war started yesterday, we can be pretty sure it is still going on today. If a hurricane started last year, we can be sure it is over by now.

The corpus that we have annotated currently contains all the 48 non-Wall-Street-Journal (non-WSJ) news articles (a total of 2132 event instances), as well as 10 WSJ articles (156 event instances), from the TimeBank corpus annotated in TimeML (Pustejovky et al., 2003). The non-WSJ articles (mainly political and disaster news) include both print and broadcast news that are from a variety of news sources, such as ABC, AP, and VOA.

In the corpus, every event to be annotated was already identified in TimeBank. Annotators were instructed to provide lower and upper bounds on the duration of the event, encompassing 80% of the possibilities, excluding anomalous cases, and taking the entire context of the article into account. For example, here is the graphical output of the annotations (3 annotators) for the "finished" event (underlined) in the sentence

After the victim, Linda Sanders, 35, had finished her cleaning and was waiting for her clothes to dry,...



```
Event "finished":

s         mi        hr
|---------|---------|-------
          ====      1: [1 mi, 5 mi]
======              2: [1 s, 10 s]
    ============    3: [5 s, 10 mi]
```

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 393–400,
Sydney, July 2006. ©2006 Association for Computational Linguistics

This graph shows that the first annotator believes that the event lasts for minutes whereas the second annotator believes it could only last for several seconds. The third annotates the event to range from a few seconds to a few minutes. A logarithmic scale is used for the output because of the intuition that the difference between 1 second and 20 seconds is significant, while the difference between 1 year 1 second and 1 year 20 seconds is negligible.

A preliminary exercise in annotation revealed about a dozen classes of systematic discrepancies among annotators' judgments. We thus developed guidelines to make annotators aware of these cases and to guide them in making the judgments. For example, many occurrences of verbs and other event descriptors refer to multiple events, especially but not exclusively if the subject or object of the verb is plural. In "*Iraq has destroyed its long-range missiles*", there is the time it takes to destroy one missile and the duration of the interval in which all the individual events are situated – the time it takes to destroy all its missiles. Initially, there were wide discrepancies because some annotators would annotate one value, others the other. Annotators are now instructed to make judgments on both values in this case. The use of the annotation guidelines resulted in about 10% improvement in inter-annotator agreement (Pan et al., 2006), measured as described in Section 2.

There is a residual of gross discrepancies in annotators' judgments that result from differences of opinion, for example, about how long a government policy is typically in effect. But the number of these discrepancies was surprisingly small.

The method and guidelines for annotation are described in much greater detail in (Pan et al., 2006). In the current paper, we focus on how inter-annotator agreement is measured, in Section 2, and in Sections 3-5 on the machine learning experiments. Because the annotated corpus is still fairly small, we cannot hope to learn to make *fine-grained* judgments of event durations that are currently annotated in the corpus, but as we demonstrate, it is possible to learn useful *coarse-grained* judgments.

Although there has been much work on temporal anchoring and event ordering in text (Hitzeman et al., 1995; Mani and Wilson, 2000; Filatova and Hovy, 2001; Boguraev and Ando, 2005), to our knowledge, there has been no serious published empirical effort to model and learn vague and implicit duration information in natural language, such as the typical durations of events, and to perform reasoning over this information. (Cyc apparently has some fuzzy duration information, although it is not generally available; Rieger (1974) discusses the issue for less than a page; there has been work in fuzzy logic on representing and reasoning with imprecise durations (Godo and Vila, 1995; Fortemps, 1997), but these make no attempt to collect human judgments on such durations or learn to extract them automatically from texts.)

2 Inter-Annotator Agreement

Although the graphical output of the annotations enables us to visualize quickly the level of agreement among different annotators for each event, a quantitative measurement of the agreement is needed.

The kappa statistic (Krippendorff, 1980; Carletta, 1996) has become the de facto standard to assess inter-annotator agreement. It is computed as:

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

$P(A)$ is the observed agreement among the annotators, and $P(E)$ is the expected agreement, which is the probability that the annotators agree by chance.

In order to compute the kappa statistic for our task, we have to compute $P(A)$ and $P(E)$, but those computations are not straightforward.

$P(A)$: What should count as agreement among annotators for our task?

$P(E)$: What is the probability that the annotators agree by chance for our task?

2.1 What Should Count as Agreement?

Determining what should count as agreement is not only important for assessing inter-annotator agreement, but is also crucial for later evaluation of machine learning experiments. For example, for a given event with a known gold standard duration range from 1 hour to 4 hours, if a machine learning program outputs a duration of 3 hours to 5 hours, how should we evaluate this result?

In the literature on the kappa statistic, most authors address only category data; some can handle more general data, such as data in interval scales or ratio scales. However, none of the techniques directly apply to our data, which are ranges of durations from a lower bound to an upper bound.

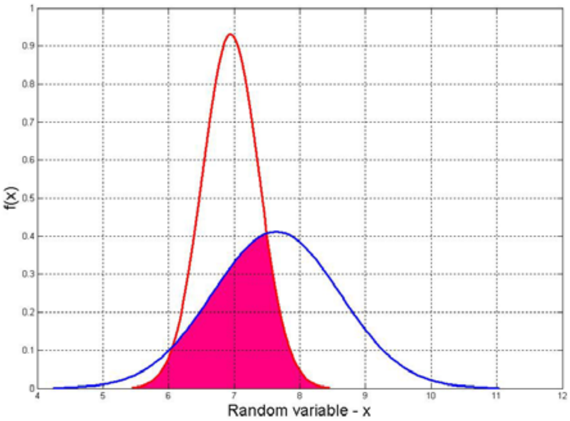


Figure 1: Overlap of Judgments of [10 minutes, 30 minutes] and [10 minutes, 2 hours].

In fact, what coders were instructed to annotate for a given event is not just a range, but a *duration distribution* for the event, where the area between the lower bound and the upper bound covers about 80% of the entire distribution area. Since it's natural to assume the most likely duration for such distribution is its mean (average) duration, and the distribution flattens out toward the upper and lower bounds, we use the normal or Gaussian distribution to model our duration distributions. If the area between lower and upper bounds covers 80% of the entire distribution area, the bounds are each 1.28 standard deviations from the mean.

Figure 1 shows the overlap in distributions for judgments of [10 minutes, 30 minutes] and [10 minutes, 2 hours], and the overlap or agreement is 0.508706.

2.2 Expected Agreement

What is the probability that the annotators agree by chance for our task? The first quick response to this question may be 0, if we consider all the possible durations from 1 second to 1000 years or even positive infinity.

However, not all the durations are equally possible. As in (Krippendorff, 1980), we assume there exists one global distribution for our task (i.e., the duration ranges for all the events), and "chance" annotations would be consistent with this distribution. Thus, the baseline will be an annotator who knows the global distribution and annotates in accordance with it, but does not read the specific article being annotated. Therefore, we must compute the global distribution of the durations, in particular, of their means and their widths. This will be of interest not only in determining expected agreement, but also in terms of what it says about the genre of news articles and about fuzzy judgments in general.

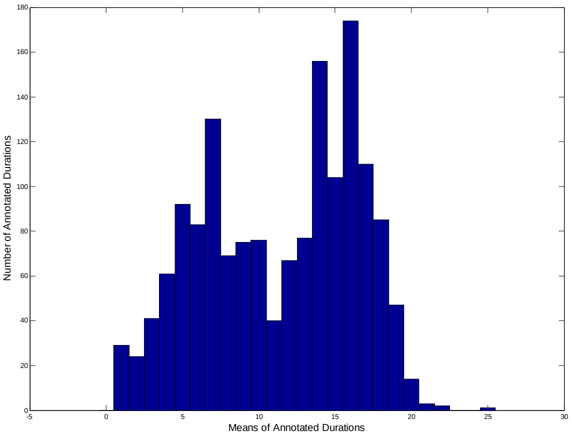


Figure 2: Distribution of Means of Annotated Durations.

We first compute the distribution of the means of all the annotated durations. Its histogram is shown in Figure 2, where the horizontal axis represents the mean values in the natural logarithmic scale and the vertical axis represents the number of annotated durations with that mean.

There are two peaks in this distribution. One is from 5 to 7 in the natural logarithmic scale, which corresponds to about 1.5 minutes to 30 minutes. The other is from 14 to 17 in the natural logarithmic scale, which corresponds to about 8 days to 6 months. One could speculate that this bimodal distribution is because daily newspapers report short events that happened the day before and place them in the context of larger trends.

We also compute the distribution of the widths (i.e., $X_{upper} – X_{lower}$) of all the annotated durations, and its histogram is shown in Figure 3, where the horizontal axis represents the width in the natural logarithmic scale and the vertical axis represents the number of annotated durations with that width. Note that it peaks at about a half order of magnitude (Hobbs and Kreinovich, 2001).

Since the global distribution is determined by the above mean and width distributions, we can then compute the expected agreement, i.e., the probability that the annotators agree by chance, where the chance is actually based on this global distribution.

Two different methods were used to compute the expected agreement (baseline), both yielding nearly equal results. These are described in detail in (Pan et al., 2006). For both, P(E) is about 0.15.

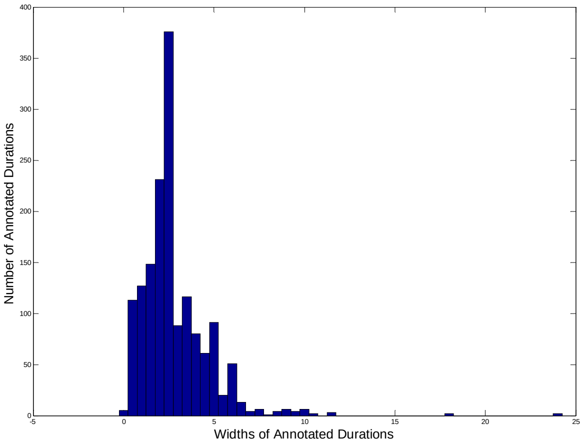


Figure 3: Distribution of Widths of Annotated Durations.

3 Features

In this section, we describe the lexical, syntactic, and semantic features that we considered in learning event durations.

3.1 Local Context

For a given event, the local context features include a window of *n* tokens to its left and *n* tokens to its right, as well as the event itself, for *n* = {0, 1, 2, 3}. The best *n* determined via cross validation turned out to be 0, i.e., the event itself with no local context. But we also present results for *n* = 2 in Section 4.3 to evaluate the utility of local context.

A token can be a word or a punctuation mark. Punctuation marks are not removed, because they can be indicative features for learning event durations. For example, the quotation mark is a good indication of quoted reporting events, and the duration of such events most likely lasts for seconds or minutes, depending on the length of the quoted content. However, there are also cases where quotation marks are used for other purposes, such as emphasis of quoted words and titles of artistic works.

For each token in the local context, including the event itself, three features are included: the original form of the token, its lemma (or root form), and its part-of-speech (POS) tag. The lemma of the token is extracted from parse trees generated by the CONTEX parser (Hermjakob and Mooney, 1997) which includes rich context information in parse trees, and the Brill tagger (Brill, 1992) is used for POS tagging.

The context window doesn't cross the boundaries of sentences. When there are not enough tokens on either side of the event within the window, "NULL" is used for the feature values.

Features	Original	Lemma	POS
Event	signed	sign	VBD
1token-after	the	the	DT
2token-after	plan	plan	NN
1token-before	Friday	Friday	NNP
2token-before	on	on	IN

Table 1: Local context features for the "signed" event in sentence (1) with n = 2.

The local context features extracted for the "signed" event in sentence (1) is shown in Table 1 (with a window size *n* = 2). The feature vector is [signed, sign, VBD, the, the, DT, plan, plan, NN, Friday, Friday, NNP, on, on, IN].

(1) *The two presidents on Friday <u>signed</u> the plan.*

3.2 Syntactic Relations

The information in the event's syntactic environment is very important in deciding the durations of events. For example, there is a difference in the durations of the "watch" events in the phrases "*<u>watch</u> a movie*" and "*<u>watch</u> a bird fly*".

For a given event, both the head of its subject and the head of its object are extracted from the parse trees generated by the CONTEX parser. Similarly to the local context features, for both the subject head and the object head, their original form, lemma, and POS tags are extracted as features. When there is no subject or object for an event, "NULL" is used for the feature values.

For the "signed" event in sentence (1), the head of its subject is "presidents" and the head of its object is "plan". The extracted syntactic relation features are shown in Table 2, and the feature vector is [presidents, president, NNS, plan, plan, NN].

3.3 WordNet Hypernyms

Events with the same hypernyms may have similar durations. For example, events "ask" and "talk" both have a direct WordNet (Miller, 1990) hypernym of "communicate", and most of the time they do have very similar durations in the corpus.

However, closely related events don't always have the same direct hypernyms. For example, "see" has a direct hypernym of "perceive", whereas "observe" needs two steps up through the hypernym hierarchy before reaching "perceive". Such correlation between events may be lost if only the direct hypernyms of the words are extracted.

Features	Original	Lemma	POS
Subject	presidents	president	NNS
Object	plan	plan	NN

Table 2: Syntactic relation features for the "signed" event in sentence (1).

Feature	1-hyper	2-hyper	3-hyper
Event	write	communicate	interact
Subject	corporate executive	executive	adminis-trator
Object	idea	content	cognition

Table 3: WordNet hypernym features for the event ("signed"), its subject ("presidents"), and its object ("plan") in sentence (1).

It is useful to extract the hypernyms not only for the event itself, but also for the subject and object of the event. For example, events related to a group of people or an organization usually last longer than those involving individuals, and the hypernyms can help distinguish such concepts. For example, "society" has a "group" hypernym (2 steps up in the hierarchy), and "school" has an "organization" hypernym (3 steps up). The direct hypernyms of nouns are always not general enough for such purpose, but a hypernym at too high a level can be too general to be useful. For our learning experiments, we extract the first 3 levels of hypernyms from WordNet.

Hypernyms are only extracted for the events and their subjects and objects, not for the local context words. For each level of hypernyms in the hierarchy, it's possible to have more than one hypernym, for example, "see" has two direct hypernyms, "perceive" and "comprehend". For a given word, it may also have more than one sense in WordNet. In such cases, as in (Gildea and Jurafsky, 2002), we only take the first sense of the word and the first hypernym listed for each level of the hierarchy. A word disambiguation module might improve the learning performance. But since the features we need are the hypernyms, not the word sense itself, even if the first word sense is not the correct one, its hypernyms can still be good enough in many cases. For example, in one news article, the word "controller" refers to an air traffic controller, which corresponds to the second sense in WordNet, but its first sense (business controller) has the same hypernym of "person" (3 levels up) as the second sense (direct hypernym). Since we take the first 3 levels of hypernyms, the correct hypernym is still extracted.

P(A)	P(E)	Kappa
0.877	0.528	0.740
	0.500	0.755

Table 4: Inter-Annotator Agreement for Binary Event Durations.

When there are less than 3 levels of hypernyms for a given word, its hypernym on the previous level is used. When there is no hypernym for a given word (e.g., "go"), the word itself will be used as its hypernyms. Since WordNet only provides hypernyms for nouns and verbs, "NULL" is used for the feature values for a word that is not a noun or a verb.

For the "signed" event in sentence (1), the extracted WordNet hypernym features for the event ("signed"), its subject ("presidents"), and its object ("plan") are shown in Table 3, and the feature vector is [write, communicate, interact, corporate_executive, executive, administrator, idea, content, cognition].

4 Experiments

The distribution of the means of the annotated durations in Figure 2 is bimodal, dividing the events into those that take less than a day and those that take more than a day. Thus, in our first machine learning experiment, we have tried to learn this *coarse-grained* event duration information as a binary classification task.

4.1 Inter-Annotator Agreement, Baseline, and Upper Bound

Before evaluating the performance of different learning algorithms, the inter-annotator agreement, the baseline and the upper bound for the learning task are assessed first.

Table 4 shows the inter-annotator agreement results among 3 annotators for binary event durations. The experiments were conducted on the same data sets as in (Pan et al., 2006). Two kappa values are reported with different ways of measuring expected agreement (P(E)), i.e., whether or not the annotators have prior knowledge of the global distribution of the task.

The human agreement before reading the guidelines (0.877) is a good estimate of the *upper bound* performance for this binary classification task. The *baseline* for the learning task is always taking the most probable class. Since 59.0% of the total data is "long" events, the baseline performance is 59.0%**.**

Class	Algor.	Prec.	Recall	F-Score
Short	**SVM**	0.707	0.606	**0.653**
	NB	0.567	0.768	0.652
	C4.5	0.571	0.600	0.585
Long	**SVM**	0.793	0.857	**0.823**
	NB	0.834	0.665	0.740
	C4.5	0.765	0.743	0.754

Table 5: Test Performance of Three Algorithms.

Algorithm	Precision
Baseline	59.0%
C4.5	69.1%
NB	70.3%
SVM	**76.6%**
Human Agreement	87.7%

Table 6: Overall Test Precision on non-WSJ Data.

4.2 Data

The original annotated data can be straightforwardly transformed for this binary classification task. For each event annotation, the most likely (mean) duration is calculated first by averaging (the logs of) its lower and upper bound durations. If its most likely (mean) duration is less than a day (about 11.4 in the natural logarithmic scale), it is assigned to the "short" event class, otherwise it is assigned to the "long" event class. (Note that these labels are strictly a convenience and not an analysis of the meanings of "short" and "long".)

We divide the total annotated non-WSJ data (2132 event instances) into two data sets: a training data set with 1705 event instances (about 80% of the total non-WSJ data) and a held-out test data set with 427 event instances (about 20% of the total non-WSJ data). The WSJ data (156 event instances) is kept for further test purposes (see Section 4.4).

4.3 Experimental Results (non-WSJ)

Learning Algorithms. Three supervised learning algorithms were evaluated for our binary classification task, namely, Support Vector Machines (SVM) (Vapnik, 1995), Naïve Bayes (NB) (Duda and Hart, 1973), and Decision Trees C4.5 (Quinlan, 1993). The Weka (Witten and Frank, 2005) machine learning package was used for the implementation of these learning algorithms. Linear kernel is used for SVM in our experiments.

Each event instance has a total of 18 feature values, as described in Section 3, for the event only condition, and 30 feature values for the local context condition when *n* = 2. For SVM and C4.5, all features are converted into binary features (6665 and 12502 features).

Results. 10-fold cross validation was used to train the learning models, which were then tested on the unseen held-out test set, and the performance (including the precision, recall, and F-score[1] for each class) of the three learning algorithms is shown in Table 5. The significant measure is overall precision, and this is shown for the three algorithms in Table 6, together with human agreement (the upper bound of the learning task) and the baseline.

We can see that among all three learning algorithms, SVM achieves the best F-score for each class and also the best overall precision (76.6%). Compared with the baseline (59.0%) and human agreement (87.7%), this level of performance is very encouraging, especially as the learning is from such limited training data.

Feature Evaluation. The best performing learning algorithm, SVM, was then used to examine the utility of combinations of four different feature sets (i.e., event, local context, syntactic, and WordNet hypernym features). The detailed comparison is shown in Table 7.

We can see that most of the performance comes from event word or phrase itself. A significant improvement above that is due to the addition of information about the subject and object. Local context does not help and in fact may hurt, and hypernym information also does not seem to help[2]. It is of interest that the most important information is that from the predicate and arguments describing the event, as our linguistic intuitions would lead us to expect.

4.4 Test on WSJ Data

Section 4.3 shows the experimental results with the learned model trained and tested on the data with the same genre, i.e., non-WSJ articles.

In order to evaluate whether the learned model can perform well on data from different news genres, we tested it on the unseen WSJ data (156 event instances). The performance (including the precision, recall, and F-score for each class) is shown in Table 8. The precision (75.0%) is very close to the test performance on the non-WSJ

[1] F-score is computed as the harmonic mean of the precision and recall: F = (2*Prec*Rec)/(Prec+Rec).

[2] In the "Syn+Hyper" cases, the learning algorithm with and without local context gives identical results, probably because the other features dominate.

Class	**Event Only (*n* = 0)**			**Event Only + Syntactic**			**Event + Syn + Hyper**		
	Prec.	Rec.	F	Prec.	Rec.	F	Prec.	Rec.	F
Short	0.742	0.465	0.571	0.758	0.587	0.662	0.707	0.606	0.653
Long	0.748	0.908	0.821	0.792	0.893	0.839	0.793	0.857	0.823
Overall Prec.	**74.7%**			**78.2%**			**76.6%**		
	Local Context (*n* = 2)			**Context + Syntactic**			**Context + Syn + Hyper**		
Short	0.672	0.568	0.615	0.710	0.600	0.650	0.707	0.606	0.653
Long	0.774	0.842	0.806	0.791	0.860	0.824	0.793	0.857	0.823
Overall Prec.	**74.2%**			**76.6%**			**76.6%**		

Table 7: Feature Evaluation with Different Feature Sets using SVM.

Class	Prec.	Rec.	F
Short	0.692	0.610	0.649
Long	0.779	0.835	0.806
Overall Prec.	**75.0%**		

Table 8: Test Performance on WSJ data.

P(A)	P(E)	Kappa
0.798	0.151	0.762
	0.143	0.764

Table 9: Inter-Annotator Agreement for Most Likely Temporal Unit.

data, and indicates the significant generalization capacity of the learned model.

5 Learning the Most Likely Temporal Unit

These encouraging results have prompted us to try to learn more fine-grained event duration information, viz., the most likely temporal units of event durations (cf. (Rieger 1974)'s ORDERHOURS, ORDERDAYS).

For each original event annotation, we can obtain the most likely (mean) duration by averaging its lower and upper bound durations, and assigning it to one of seven classes (i.e., second, minute, hour, day, week, month, and year) based on the temporal unit of its most likely duration.

However, human agreement on this more fine-grained task is low (44.4%). Based on this observation, instead of evaluating the *exact* agreement between annotators, an "*approximate* agreement" is computed for the most likely temporal unit of events. In "approximate agreement", temporal units are considered to match if they are the same temporal unit or an adjacent one. For example, "second" and "minute" match, but "minute" and "day" do not.

Some preliminary experiments have been conducted for learning this multi-classification task. The same data sets as in the binary classification task were used. The only difference is that the class for each instance is now labeled with one of the seven temporal unit classes.

Algorithm	Precision
Baseline	51.5%
C4.5	56.4%
NB	65.8%
SVM	**67.9%**
Human Agreement	79.8%

Table 10: Overall Test Precisions.

The baseline for this multi-classification task is always taking the temporal unit which with its two neighbors spans the greatest amount of data. Since the "week", "month", and "year" classes together take up largest portion (51.5%) of the data, the baseline is always taking the "month" class, where both "week" and "year" are also considered a match. Table 9 shows the inter-annotator agreement results for most likely temporal unit when using "approximate agreement". Human agreement (the upper bound) for this learning task increases from 44.4% to 79.8%.

10-fold cross validation was also used to train the learning models, which were then tested on the unseen held-out test set. The performance of the three algorithms is shown in Table 10. The best performing learning algorithm is again SVM with 67.9% test precision. Compared with the baseline (51.5%) and human agreement (79.8%), this again is a very promising result, especially for a multi-classification task with such limited training data. It is reasonable to expect that when more annotated data becomes available, the learning algorithm will achieve higher performance when learning this and more fine-grained event duration information.

Although the coarse-grained duration information may look too coarse to be useful, computers have no idea at all whether a meeting event takes seconds or centuries, so even coarse-grained estimates would give it a useful rough sense of how long each event may take. More fine-grained duration information is definitely more desirable for temporal reasoning tasks. But coarse-grained durations to a level of temporal units can already be very useful.

6 Conclusion

In the research described in this paper, we have addressed a problem -- extracting information about event durations encoded in event descriptions -- that has heretofore received very little attention in the field. It is information that can have a substantial impact on applications where the temporal placement of events is important. Moreover, it is representative of a set of problems – making use of the vague information in text – that has largely eluded empirical approaches in the past. In (Pan et al., 2006), we explicate the linguistic categories of the phenomena that give rise to grossly discrepant judgments among annotators, and give guidelines on resolving these discrepancies. In the present paper, we describe a method for measuring inter-annotator agreement when the judgments are intervals on a scale; this should extend from time to other scalar judgments. Inter-annotator agreement is too low on fine-grained judgments. However, for the coarse-grained judgments of more than or less than a day, and of approximate agreement on temporal unit, human agreement is acceptably high. For these cases, we have shown that machine-learning techniques achieve impressive results.

Acknowledgments

This work was supported by the Advanced Research and Development Activity (ARDA), now the Disruptive Technology Office (DTO), under DOD/DOI/ARDA Contract No. NBCHC040027. The authors have profited from discussions with Hoa Trang Dang, Donghui Feng, Kevin Knight, Daniel Marcu, James Pustejovsky, Deepak Ravichandran, and Nathan Sobo.

References

B. Boguraev and R. K. Ando. 2005. TimeML-Compliant Text Analysis for Temporal Reasoning. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI).*

E. Brill. 1992. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing.*

J. Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Computational Lingustics*, 22(2):249–254.

R. O. Duda and P. E. Hart. 1973. *Pattern Classification and Scene Analysis*. Wiley, New York.

E. Filatova and E. Hovy. 2001. Assigning Time-Stamps to Event-Clauses. *Proceedings of ACL Workshop on Temporal and Spatial Reasoning.*

P. Fortemps. 1997. Jobshop Scheduling with Imprecise Durations: A Fuzzy Approach. *IEEE Transactions on Fuzzy Systems* Vol. 5 No. 4.

D. Gildea and D. Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245-288.

L. Godo and L. Vila. 1995. Possibilistic Temporal Reasoning based on Fuzzy Temporal Constraints. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI).*

U. Hermjakob and R. J. Mooney. 1997. Learning Parse and Translation Decisions from Examples with Rich Context. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL).*

J. Hitzeman, M. Moens, and C. Grover. 1995. Algorithms for Analyzing the Temporal Structure of Discourse. In *Proceedings of EACL*. Dublin, Ireland.

J. R. Hobbs and V. Kreinovich. 2001. Optimal Choice of Granularity in Commonsense Estimation: Why Half Orders of Magnitude, In *Proceedings of Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, Vacouver, British Columbia.

K. Krippendorf. 1980. *Content Analysis: An introduction to its methodology*. Sage Publications.

I. Mani and G. Wilson. 2000. Robust Temporal Processing of News. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL).*

G. A. Miller. 1990. WordNet: an On-line Lexical Database. *International Journal of Lexicography* 3(4).

F. Pan, R. Mulkar, and J. R. Hobbs. 2006. An Annotated Corpus of Typical Durations of Events. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC)*, Genoa, Italy.

J. Pustejovsky, P. Hanks, R. Saurí, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro and M. Lazo. 2003. The timebank corpus. In *Corpus Linguistics*, Lancaster, U.K.

J. R. Quinlan. 1993. C4.5: *Programs for Machine Learning*. Morgan Kaufmann, San Francisco.

C. J. Rieger. 1974. Conceptual memory: A theory and computer program for processing and meaning content of natural language utterances. *Stanford AIM-233*.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York.

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

Automatic learning of textual entailments with cross-pair similarities

Fabio Massimo Zanzotto
DISCo
University of Milano-Bicocca
Milan, Italy
zanzotto@disco.unimib.it

Alessandro Moschitti
Department of Computer Science
University of Rome “Tor Vergata”
Rome, Italy
moschitti@info.uniroma2.it

Abstract

In this paper we define a novel similarity measure between examples of textual entailments and we use it as a kernel function in Support Vector Machines (SVMs). This allows us to automatically learn the *rewrite rules* that describe a non trivial set of entailment cases. The experiments with the data sets of the RTE 2005 challenge show an improvement of 4.4% over the state-of-the-art methods.

1 Introduction

Recently, textual entailment recognition has been receiving a lot of attention. The main reason is that the understanding of the basic entailment processes will allow us to model more accurate semantic theories of natural languages (Chierchia and McConnell-Ginet, 2001) and design important applications (Dagan and Glickman, 2004), e.g., Question Answering and Information Extraction.

However, previous work (e.g., (Zaenen et al., 2005)) suggests that determining whether or not a text T entails a hypothesis H is quite complex even when all the needed information is explicitly asserted. For example, the sentence T_1: “*At the end of the year, all solid companies pay dividends.*” entails the hypothesis H_1: “*At the end of the year, all solid insurance companies pay dividends.*” but it does not entail the hypothesis H_2: “*At the end of the year, all solid companies pay cash dividends.*”

Although these implications are uncontroversial, their automatic recognition is complex if we rely on models based on lexical distance (or similarity) between hypothesis and text, e.g., (Corley and Mihalcea, 2005). Indeed, according to such approaches, the hypotheses H_1 and H_2 are very similar and seem to be similarly related to T_1. This suggests that we should study the properties and differences of such two examples (negative and positive) to derive more accurate entailment models. For example, if we consider the following entailment:

$T_3 \Rightarrow H_3$?	
T_3	*“All wild animals eat plants that have scientifically proven medicinal properties.”*
H_3	*“All wild mountain animals eat plants that have scientifically proven medicinal properties.”*

we note that T_3 is structurally (and somehow lexically similar) to T_1 and H_3 is more similar to H_1 than to H_2. Thus, from $T_1 \Rightarrow H_1$ we may extract rules to derive that $T_3 \Rightarrow H_3$.

The above example suggests that we should rely not only on a *intra-pair* similarity between T and H but also on a *cross-pair* similarity between two pairs (T', H') and (T'', H''). The latter similarity measure along with a set of annotated examples allows a learning algorithm to automatically derive syntactic and lexical rules that can solve complex entailment cases.

In this paper, we define a new cross-pair similarity measure based on text and hypothesis syntactic trees and we use such similarity with traditional *intra-pair* similarities to define a novel semantic kernel function. We experimented with such kernel using Support Vector Machines (Vapnik, 1995) on the test tests of the Recognizing Textual Entailment (RTE) challenges (Dagan et al., 2005; Bar Haim et al., 2006). The comparative results show that (a) we have designed an effective way to automatically learn entailment rules from examples and (b) our approach is highly accurate and exceeds the accuracy of the current state-of-the-art

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 401–408,
Sydney, July 2006. ©2006 Association for Computational Linguistics

models (Glickman et al., 2005; Bayer et al., 2005) by about 4.4% (i.e. 63% vs. 58.6%) on the RTE 1 test set (Dagan et al., 2005).

In the remainder of this paper, Sec. 2 illustrates the related work, Sec. 3 introduces the complexity of learning entailments from examples, Sec. 4 describes our models, Sec. 6 shows the experimental results and finally Sec. 7 derives the conclusions.

2 Related work

Although the textual entailment recognition problem is not new, most of the automatic approaches have been proposed only recently. This has been mainly due to the RTE challenge events (Dagan et al., 2005; Bar Haim et al., 2006). In the following we report some of such researches.

A first class of methods defines measures of the distance or similarity between T and H either assuming the independence between words (Corley and Mihalcea, 2005; Glickman et al., 2005) in a bag-of-word fashion or exploiting syntactic interpretations (Kouylekov and Magnini, 2005). A pair (T, H) is then in entailment when $sim(T, H) > \alpha$. These approaches can hardly determine whether the entailment holds in the examples of the previous section. From the point of view of bag-of-word methods, the pairs (T_1, H_1) and (T_1, H_2) have both the same intra-pair similarity since the sentences of T_1 and H_1 as well as those of T_1 and H_2 differ by a noun, *insurance* and *cash*, respectively. At syntactic level, also, we cannot capture the required information as such nouns are both noun modifiers: *insurance* modifies *companies* and *cash* modifies *dividends*.

A second class of methods can give a solution to the previous problem. These methods generally combine a similarity measure with a set of possible transformations $\mathcal{T}$ applied over syntactic and semantic interpretations. The entailment between T and H is detected when there is a transformation $r \in \mathcal{T}$ so that $sim(r(T), H) > \alpha$. These transformations are logical rules in (Bos and Markert, 2005) or sequences of allowed *rewrite rules* in (de Salvo Braz et al., 2005). The disadvantage is that such rules have to be manually designed. Moreover, they generally model better positive implications than negative ones and they do not consider errors in syntactic parsing and semantic analysis.

3 Challenges in learning from examples

In the introductory section, we have shown that, to carry out automatic learning from examples, we need to define a cross-pair similarity measure. Its definition is not straightforward as it should detect whether two pairs (T', H') and (T'', H'') realize the same *rewrite rules*. This measure should consider pairs similar when: (1) T' and H' are structurally similar to T'' and H'', respectively and (2) the lexical relations within the pair (T', H') are compatible with those in (T'', H''). Typically, T and H show a certain degree of overlapping, thus, lexical relations (e.g., between the same words) determine *word movements* from T to H (or vice versa). This is important to model the syntactic/lexical similarity between example pairs. Indeed, if we encode such movements in the syntactic parse trees of texts and hypotheses, we can use interesting similarity measures defined for syntactic parsing, e.g., the tree kernel devised in (Collins and Duffy, 2002).

To consider structural and lexical relation similarity, we augment syntactic trees with *placeholders* which identify linked words. More in detail:

- We detect links between words w_t in T that are equal, similar, or semantically dependent on words w_h in H. We call *anchors* the pairs (w_t, w_h) and we associate them with *placeholders*. For example, in Fig. 1, the placeholder $\boxed{2''}$ indicates the *(companies,companies)* anchor between T_1 and H_1. This allows us to derive the word movements between text and hypothesis.
- We align the trees of the two texts T' and T'' as well as the tree of the two hypotheses H' and H'' by considering the *word movements*. We find a correct mapping between placeholders of the two hypothesis H' and H'' and apply it to the tree of H'' to substitute its placeholders. The same mapping is used to substitute the placeholders in T''. This mapping should maximize the structural *similarity* between the four trees by considering that placeholders augment the node labels. Hence, the cross-pair similarity computation is reduced to the tree similarity computation.

The above steps define an effective cross-pair similarity that can be applied to the example in Fig. 1: T_1 and T_3 share the subtree in bold starting with **S** → **NP VP**. The lexicals in T_3 and H_3 are quite different from those T_1 and H_1, but we can rely on the structural properties expressed by their bold subtrees. These are more similar to the subtrees of T_1 and H_1 than those of T_1 and H_2, respectively. Indeed, H_1 and H_3 share the production **NP** → **DT JJ NN NNS** while H_2 and H_3 do

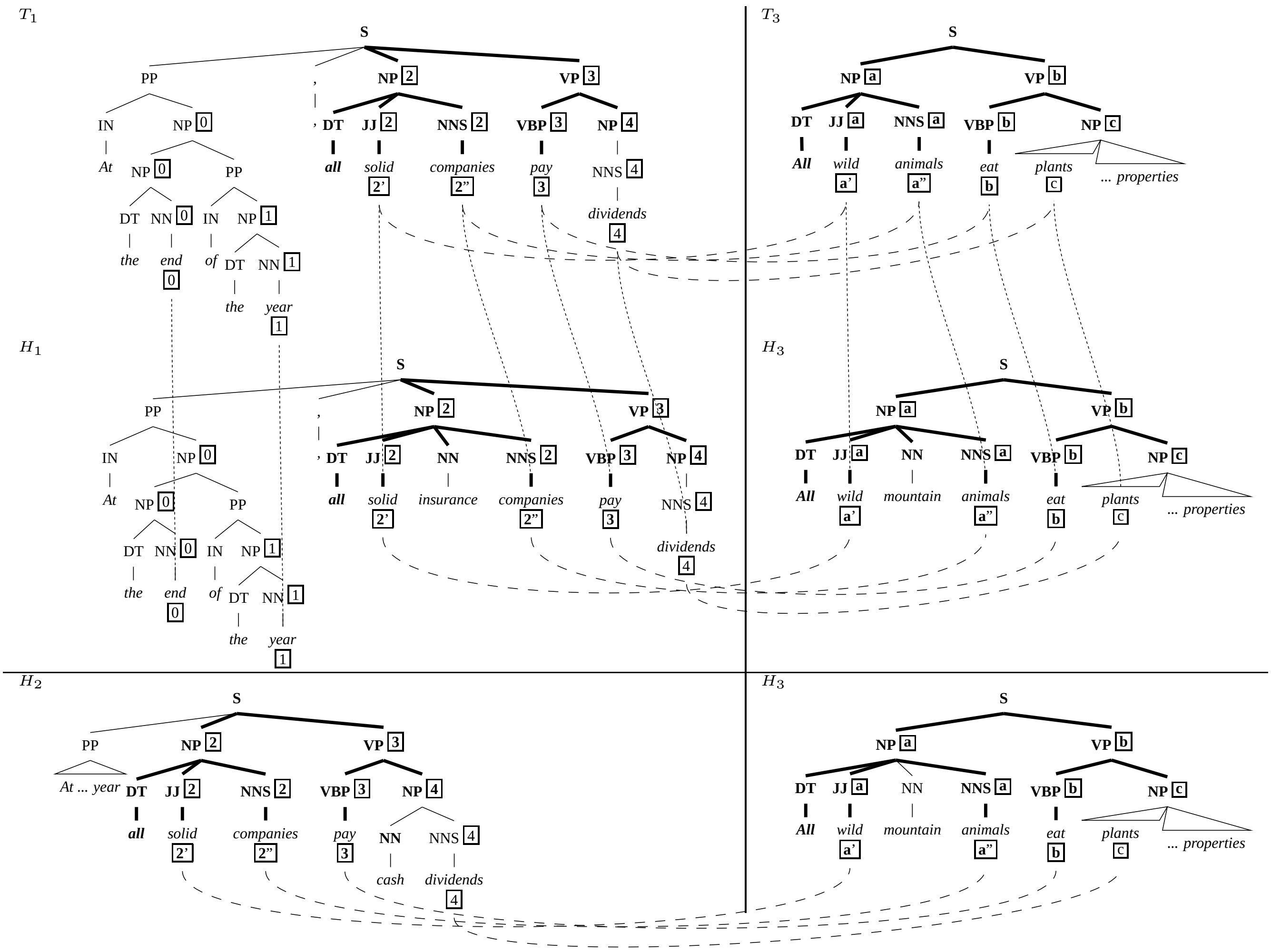


Figure 1: Relations between (T_1, H_1), (T_1, H_2), and (T_3, H_3).

not. Consequently, to decide if (T_3,H_3) is a valid entailment, we should rely on the decision made for (T_1, H_1). Note also that the dashed lines connecting placeholders of two texts (hypotheses) indicate structurally equivalent nodes. For instance, the dashed line between $\boxed{3}$ and $\boxed{b}$ links the main verbs both in the texts T_1 and T_3 and in the hypotheses H_1 and H_3. After substituting $\boxed{3}$ with $\boxed{b}$ and $\boxed{2}$ with $\boxed{a}$, we can detect if T_1 and T_3 share the bold subtree $\mathbf{S} \rightarrow \mathbf{NP}\boxed{2}\ \mathbf{VP}\boxed{3}$. As such subtree is shared also by H_1 and H_3, the words within the pair (T_1, H_1) are correlated similarly to the words in (T_3, H_3).

The above example emphasizes that we need to derive the *best* mapping between placeholder sets. It can be obtained as follows: let A' and A'' be the placeholders of (T', H') and (T'', H''), respectively, without loss of generality, we consider $|A'| \geq |A''|$ and we align a subset of A' to A''. The best alignment is the one that maximizes the syntactic and lexical overlapping of the two subtrees induced by the aligned set of anchors.

More precisely, let C be the set of all bijective mappings from $a' \subseteq A' : |a'| = |A''|$ to A'', an element $c \in C$ is a substitution function. We define as the best alignment the one determined by

$$c_{max} = argmax_{c \in C}(K_T(t(H', c), t(H'', i)) + K_T(t(T', c), t(T'', i)) \qquad (1)$$

where (a) $t(S, c)$ returns the syntactic tree of the hypothesis (text) S with placeholders replaced by means of the substitution c, (b) i is the identity substitution and (c) $K_T(t_1, t_2)$ is a function that measures the similarity between the two trees t_1 and t_2 (for more details see Sec. 4.2). For example, the c_{max} between (T_1, H_1) and (T_3, H_3) is $\{(\boxed{2'}, \boxed{a'}), (\boxed{2''}, \boxed{a''}), (\boxed{3}, \boxed{b}), (\boxed{4}, \boxed{c})\}$.

4 Similarity Models

In this section we describe how anchors are found at the level of a single pair (T, H) (Sec. 4.1). The anchoring process gives the direct possibility of implementing an inter-pair similarity that can be used as a baseline approach or in combination with the cross-pair similarity. This latter will be implemented with tree kernel functions over syntactic structures (Sec. 4.2).

4.1 Anchoring and Lexical Similarity

The algorithm that we design to find the anchors is based on similarity functions between words or more complex expressions. Our approach is in line with many other researches (e.g., (Corley and Mihalcea, 2005; Glickman et al., 2005)).

Given the set of content words (verbs, nouns, adjectives, and adverbs) W_T and W_H of the two sentences T and H, respectively, the set of anchors $A \subset W_T \times W_H$ is built using a similarity measure between two words $sim_w(w_t, w_h)$. Each element $w_h \in W_H$ will be part of a pair $(w_t, w_h) \in A$ if:
1) $sim_w(w_t, w_h) \neq 0$
2) $sim_w(w_t, w_h) = \max_{w'_t \in W_T} sim_w(w'_t, w_h)$
According to these properties, elements in W_H can participate in more than one anchor and conversely more than one element in W_H can be linked to a single element $w \in W_T$.

The similarity $sim_w(w_t, w_h)$ can be defined using different indicators and resources. First of all, two words are maximally similar if these have the same surface form $w_t = w_h$. Second, we can use one of the WordNet (Miller, 1995) similarities indicated with $d(l_w, l_{w'})$ (in line with what was done in (Corley and Mihalcea, 2005)) and different relation between words such as the lexical entailment between verbs (Ent) and derivationally relation between words (Der). Finally, we use the edit distance measure $lev(w_t, w_h)$ to capture the similarity between words that are missed by the previous analysis for misspelling errors or for the lack of derivationally forms not coded in WordNet.

As result, given the syntactic category $c_w \in \{noun, verb, adjective, adverb\}$ and the lemmatized form l_w of a word w, the similarity measure between two words w and w' is defined as follows:

$$sim_w(w, w') = \begin{cases} 1 & \textbf{if } w = w' \vee \\ & l_w = l_{w'} \wedge c_w = c_{w'} \vee \\ & ((l_w, c_w), (l_{w'}, c_{w'})) \in Ent \vee \\ & ((l_w, c_w), (l_{w'}, c_{w'})) \in Der \vee \\ & lev(w, w') = 1 \\ d(l_w, l_{w'}) & \textbf{if } c_w = c_{w'} \wedge d(l_w, l_{w'}) > 0.2 \\ 0 & \textbf{otherwise} \end{cases} \quad (2)$$

It is worth noticing that, the above measure is not a *pure* similarity measure as it includes the entailment relation that does not represent synonymy or similarity between verbs. To emphasize the contribution of each used resource, in the experimental section, we will compare Eq. 2 with some versions that exclude some word relations.

The above word similarity measure can be used to compute the similarity between T and H. In line with (Corley and Mihalcea, 2005), we define it as:

$$s_1(T, H) = \frac{\displaystyle\sum_{(w_t, w_h) \in A} sim_w(w_t, w_h) \times idf(w_h)}{\displaystyle\sum_{w_h \in W_H} idf(w_h)} \quad (3)$$

where $idf(w)$ is the inverse document frequency of the word w. For sake of comparison, we consider also the corresponding more classical version that does not apply the inverse document frequency

$$s_2(T, H) = \sum_{(w_t, w_h) \in A} sim_w(w_t, w_h) / |W_H| \quad (4)$$

¿From the above intra-pair similarities, s_1 and s_2, we can obtain the baseline *cross-pair* similarities based on only lexical information:

$$K_i((T', H'), (T'', H'')) = s_i(T', H') \times s_i(T'', H''), \quad (5)$$

where $i \in \{1, 2\}$. In the next section we define a novel cross-pair similarity that takes into account syntactic evidence by means of tree kernel functions.

4.2 Cross-pair syntactic kernels

Section 3 has shown that to measure the syntactic similarity between two pairs, (T', H') and (T'', H''), we should capture the number of common subtrees between texts and hypotheses that share the same anchoring scheme. The best alignment between anchor sets, i.e. the best substitution c_{max}, can be found with Eq. 1. As the corresponding maximum quantifies the *alignment degree*, we could define a cross-pair similarity as follows:

$$K_s((T', H'), (T'', H'')) = \max_{c \in C} \Big(K_T(t(H', c), t(H'', i)) + K_T(t(T', c), t(T'', i)) \Big), \quad (6)$$

where as $K_T(t_1, t_2)$ we use the tree kernel function defined in (Collins and Duffy, 2002). This evaluates the number of subtrees shared by t_1 and t_2, thus defining an implicit substructure space.

Formally, given a subtree space $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$, the indicator function $I_i(n)$ is equal to 1 if the target f_i is rooted at node n and equal to 0 otherwise. A tree-kernel function over t_1 and t_2 is $K_T(t_1, t_2) = \sum_{n_1 \in N_{t_1}} \sum_{n_2 \in N_{t_2}} \Delta(n_1, n_2)$, where N_{t_1} and N_{t_2} are the sets of the t_1's and t_2's nodes, respectively. In turn $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \lambda^{l(f_i)} I_i(n_1) I_i(n_2)$, where $0 \leq \lambda \leq 1$ and $l(f_i)$ is the number of levels of the subtree f_i. Thus $\lambda^{l(f_i)}$ assigns a lower weight to larger fragments. When $\lambda = 1$, Δ is equal to the number of common fragments rooted at nodes n_1 and n_2. As described in (Collins and Duffy, 2002), Δ can be computed in $O(|N_{t_1}| \times |N_{t_2}|)$.

The K_T function has been proven to be a valid kernel, i.e. its associated *Gram* matrix is positive-semidefinite. Some basic operations on kernel functions, e.g. the sum, are closed with respect to the set of valid kernels. Thus, if the maximum held such property, Eq. 6 would be a valid kernel and we could use it in kernel based machines like SVMs. Unfortunately, a counterexample illustrated in (Boughorbel et al., 2004) shows that the *max* function does not produce valid kernels in general.

However, we observe that: (1) $K_s((T', H'), (T'', H''))$ is a symmetric function since the set of transformation C are always computed with respect to the pair that has the largest anchor set; (2) in (Haasdonk, 2005), it is shown that when kernel functions are not positive semidefinite, SVMs still solve a data separation problem in pseudo Euclidean spaces. The drawback is that the solution may be only a local optimum. Therefore, we can experiment Eq. 6 with SVMs and observe if the empirical results are satisfactory. Section 6 shows that the solutions found by Eq. 6 produce accuracy higher than those evaluated on previous automatic textual entailment recognition approaches.

5 Refining cross-pair syntactic similarity

In the previous section we have defined the intra and the cross pair similarity. The former does not show relevant implementation issues whereas the latter should be optimized to favor its applicability with SVMs. The Eq. 6 improvement depends on three factors: (1) its computation complexity; (2) a correct marking of tree nodes with placeholders; and, (3) the pruning of irrelevant information in large syntactic trees.

5.1 Controlling the computational cost

The computational cost of cross-pair similarity between two tree pairs (Eq. 6) depends on the size of C. This is combinatorial in the size of A' and A'', i.e. $|C| = (|A'| - |A''|)!|A''|!$ if $|A'| \geq |A''|$. Thus we should keep the sizes of A' and A'' reasonably small.

To reduce the number of placeholders, we consider the notion of *chunk* defined in (Abney, 1996), i.e., *not recursive kernels* of noun, verb, adjective, and adverb phrases. When placeholders are in a single chunk both in the text and hypothesis we assign them the same name. For example, Fig. 1 shows the placeholders $\boxed{2'}$ and $\boxed{2''}$ that are substituted by the placeholder $\boxed{2}$. The placeholder reduction procedure also gives the possibility of resolving the ambiguity still present in the anchor set A (see Sec. 4.1). A way to eliminate the ambiguous anchors is to select the ones that reduce the final number of placeholders.

5.2 Augmenting tree nodes with placeholders

Anchors are mainly used to extract relevant syntactic subtrees between pairs of text and hypothesis. We also use them to characterize the syntactic information expressed by such subtrees. Indeed, Eq. 6 depends on the number of common subtrees between two pairs. Such subtrees are matched when they have the same node labels. Thus, to keep track of the argument movements, we augment the node labels with placeholders. The larger number of placeholders two hypotheses (texts) match the larger the number of their common substructures is (i.e. higher similarity). Thus, it is really important where placeholders are inserted.

For example, the sentences in the pair (T_1, H_1) have related subjects $\boxed{2}$ and related main verbs $\boxed{3}$. The same occurs in the sentences of the pair (T_3, H_3), respectively $\boxed{a}$ and $\boxed{b}$. To obtain such node marking, the placeholders are propagated in the syntactic tree, from the leaves[1] to the target nodes according to the head of constituents. The example of Fig. 1 shows that the placeholder $\boxed{0}$ climbs up to the node governing all the NPs.

5.3 Pruning irrelevant information in large *text* trees

Often only a portion of the parse trees is relevant to detect entailments. For instance, let us consider the following pair from the RTE 2005 corpus:

[1] To increase the generalization capacity of the tree kernel function we choose not to assign any placeholder to the leaves.

$T \Rightarrow H$	(id: 929)
T	“***Ron Gainsford, chief executive of the TSI,*** *said: ”It is a major concern to us that parents could be unwittingly exposing their children to the risk of sun damage, thinking they are better protected than they actually are.”*
H	*“Ron Gainsford is the chief executive of the TSI.”*

Only the bold part of T supports the implication; the rest is useless and also misleading: if we used it to compute the similarity it would reduce the importance of the relevant part. Moreover, as we normalize the syntactic tree kernel (K_T) with respect to the size of the two trees, we need to focus only on the part relevant to the implication.

The anchored leaves are good indicators of relevant parts but also some other parts may be very relevant. For example, the function word *not* plays an important role. Another example is given by the word *insurance* in H_1 and *mountain* in H_3 (see Fig. 1). They support the implication $T_1 \Rightarrow H_1$ and $T_1 \Rightarrow H_3$ as well as *cash* supports $T_1 \nRightarrow H_2$. By removing these words and the related structures, we cannot determine the correct implications of the first two and the incorrect implication of the second one. Thus, we keep all the words that are immediately related to relevant constituents.

The reduction procedure can be formally expressed as follows: given a syntactic tree t, the set of its nodes $N(t)$, and a set of anchors, we build a tree t' with all the nodes N' that are anchors or ancestors of any anchor. Moreover, we add to t' the leaf nodes of the original tree t that are direct children of the nodes in N'. We apply such procedure only to the syntactic trees of texts before the computation of the kernel function.

6 Experimental investigation

The aim of the experiments is twofold: we show that (a) entailment recognition rules can be learned from examples and (b) our kernel functions over syntactic structures are effective to derive syntactic properties. The above goals can be achieved by comparing the different intra and cross pair similarity measures.

6.1 Experimental settings

For the experiments, we used the Recognizing Textual Entailment Challenge data sets, which we name as follows:

- $D1$, $T1$ and $D2$, $T2$, are the development and the test sets of the first (Dagan et al., 2005) and second (Bar Haim et al., 2006) challenges, respectively. $D1$ contains 567 examples whereas $T1$, $D2$ and $T2$ have all the same size, i.e. 800 training/testing instances. The positive examples constitute the 50% of the data.
- ALL is the union of $D1$, $D2$, and $T1$, which we also split in 70%-30%. This set is useful to test if we can learn entailments from the data prepared in the two different challenges.
- $D2(50\%)'$ and $D2(50\%)''$ is a random split of $D2$. It is possible that the data sets of the two competitions are quite different thus we created this *homogeneous* split.

We also used the following resources:

- The Charniak parser (Charniak, 2000) and the `morpha` lemmatiser (Minnen et al., 2001) to carry out the syntactic and morphological analysis.
- WordNet 2.0 (Miller, 1995) to extract both the verbs in entailment, Ent set, and the derivationally related words, Der set.
- The `wn::similarity` package (Pedersen et al., 2004) to compute the Jiang&Conrath (J&C) distance (Jiang and Conrath, 1997) as in (Corley and Mihalcea, 2005). This is one of the best figure method which provides a similarity score in the $[0, 1]$ interval. We used it to implement the $d(l_w, l_{w'})$ function.
- A selected portion of the British National Corpus[2] to compute the inverse document frequency (idf). We assigned the maximum idf to words not found in the BNC.
- SVM-light-TK[3] (Moschitti, 2006) which encodes the basic tree kernel function, K_T, in SVM-light (Joachims, 1999). We used such software to implement K_s (Eq. 6), K_1, K_2 (Eq. 5) and $K_s + K_i$ kernels. The latter combines our new kernel with traditional approaches ($i \in \{1, 2\}$).

6.2 Results and analysis

Table 1 reports the results of different similarity kernels on the different training and test splits described in the previous section. The table is organized as follows:

The first 5 rows (*Experiment settings*) report the intra-pair similarity measures defined in Section 4.1, the 6th row refers to only the idf similarity metric whereas the following two rows report the cross-pair similarity carried out with Eq. 6 with (*Synt Trees with placeholders*) and without (*Only Synt Trees*) augmenting the trees with placeholders, respectively. Each column in the *Experiment*

[2] http://www.natcorp.ox.ac.uk/

[3] SVM-light-TK is available at `http://ai-nlp.info.uniroma2.it/moschitti/`

Experiment Settings								
$w = w' \vee l_w = l_{w'} \wedge c_w = c_{w'}$	√	√	√	√	√	√	√	√
$c_w = c_{w'} \wedge d(l_w, l_{w'}) > 0.2$			√	√	√	√	√	√
$((l_w, c_w), (l_{w'}, c_{w'})) \in Der$					√	√	√	√
$((l_w, c_w), (l_{w'}, c_{w'})) \in Ent$					√	√	√	√
$lev(w, w') = 1$						√	√	√
idf		√		√	√	√	√	√
Only Synt Trees							√	
Synt Trees with placeholders								√
Datasets								
“Train:$D1$-Test:$T1$”	0.5388	0.5813	0.5500	0.5788	0.5900	0.5888	0.6213	**0.6300**
“Train:$T1$-Test:$D1$”	0.5714	0.5538	0.5767	0.5450	0.5591	0.5644	0.5732	**0.5838**
“Train:$D2(50\%)'$-Test:$D2(50\%)''$”	0.6034	0.5961	0.6083	0.6010	0.6083	0.6083	0.6156	**0.6350**
“Train:$D2(50\%)''$-Test:$D2(50\%)'$”	0.6452	0.6375	0.6427	0.6350	0.6324	0.6272	0.5861	**0.6607**
“Train:$D2$-Test:$T2$”	0.6000	0.5950	0.6025	0.6050	0.6050	0.6038	0.6238	**0.6388**
Mean	0.5918 (± 0.0396)	0.5927 (± 0.0303)	0.5960 (± 0.0349)	0.5930 (± 0.0335)	0.5990 (± 0.0270)	0.5985 (± 0.0235)	0.6040 (± 0.0229)	**0.6297** (± 0.0282)
“Train:$ALL(70\%)$-Test:$ALL(30\%)$”	0.5902	0.6024	0.6009	-	0.6131	0.6193	0.6086	**0.6376**
“Train:ALL-Test:$T2$”	0.5863	0.5975	0.5975	0.6038	-	-	0.6213	**0.6250**

Table 1: Experimental results of the different methods over different test settings

settings indicates a different intra-pair similarity measure built by means of a combination of basic similarity approaches. These are specified with the check sign √. For example, Column 5 refers to a model using: the surface word form similarity, the $d(l_w, l_{w'})$ similarity and the idf.

The next 5 rows show the accuracy on the data sets and splits used for the experiments and the next row reports the average and Std. Dev. over the previous 5 results. Finally, the last two rows report the accuracy on ALL dataset split in 70/30% and on the whole ALL dataset used for training and T2 for testing.

¿From the table we note the following aspects:
- First, the lexical-based distance kernels K_1 and K_2 (Eq. 5) show accuracy significantly higher than the random baseline, i.e. 50%. In all the datasets (except for the first one), the $sim_w(T, H)$ similarity based on the lexical overlap (first column) provides an accuracy essentially similar to the best lexical-based distance method.
- Second, the dataset “Train:$D1$-Test:$T1$” allows us to compare our models with the ones of the first RTE challenge (Dagan et al., 2005). The accuracy reported for the best systems, i.e. 58.6% (Glickman et al., 2005; Bayer et al., 2005), is not significantly different from the result obtained with K_1 that uses the idf.
- Third, the dramatic improvement observed in (Corley and Mihalcea, 2005) on the dataset “Train:$D1$-Test:$T1$” is given by the idf rather than the use of the J&C similarity (second vs. third columns). The use of J&C with the idf decreases the accuracy of the idf alone.
- Next, our approach (last column) is significantly better than all the other methods as it provides the best result for each combination of training and test sets. On the “Train:$D1$-Test:$T1$” test set, it exceeds the accuracy of the current state-of-the-art models (Glickman et al., 2005; Bayer et al., 2005) by about 4.4 absolute percent points (63% vs. 58.6%) and 4% over our best lexical similarity measure. By comparing the average on all datasets, our system improves on all the methods by at least 3 absolute percent points.
- Finally, the accuracy produced by *Synt Trees with placeholders* is higher than the one obtained with *Only Synt Trees*. Thus, the use of placeholders is fundamental to automatically learn entailments from examples.

6.2.1 Qualitative analysis

Hereafter we show some instances selected from the first experiment “Train:$T1$-Test:$D1$”. They were correctly classified by our overall model (last column) and miss-classified by the models in the seventh and in the eighth columns. The first is an example in entailment:

$T \Rightarrow H$	(id: 35)
T	*“Saudi Arabia, the biggest oil producer in the world, was once a supporter of Osama bin Laden and his associates who led attacks against the United States.”*
H	*“Saudi Arabia is the world’s biggest oil exporter.”*

It was correctly classified by exploiting examples like these two:

$T \Rightarrow H$	(id: 929)
T	*“Ron Gainsford, chief executive of the TSI, said: ...”*
H	*“Ron Gainsford is the chief executive of the TSI.”*

$T \Rightarrow H$	(id: 976)
T	*“Harvey Weinstein, the co-chairman of Miramax, who was instrumental in popularizing both independent and foreign films with broad audiences, agrees.”*
H	*“Harvey Weinstein is the co-chairman of Miramax.”*

The rewrite rule is: *"X, Y, ..."* implies *"X is Y"*. This rule is also described in (Hearst, 1992).

A more interesting rule relates the following two sentences which are not in entailment:

$T \nRightarrow H$	(id: 2045)
T	*"Mrs. Lane, who has been a Director since 1989, is Special Assistant to the Board of Trustees and to the President of Stanford University."*
H	*"Mrs. Lane is the president of Stanford University."*

It was correctly classified using instances like the following:

$T \nRightarrow H$	(id: 2044)
T	*"Jacqueline B. Wender is Assistant to the President of Stanford University."*
H	*"Jacqueline B. Wender is the President of Stanford University."*

$T \nRightarrow H$	(id: 2069)
T	*"Grieving father Christopher Yavelow hopes to deliver one million letters to the queen of Holland to bring his children home."*
H	*"Christopher Yavelow is the queen of Holland."*

Here, the implicit rule is: *"X (VP (V ...) (NP (to Y) ...)"* does not imply *"X is Y"*.

7 Conclusions

We have presented a model for the automatic learning of rewrite rules for textual entailments from examples. For this purpose, we devised a novel powerful kernel based on cross-pair similarities. We experimented with such kernel using Support Vector Machines on the RTE test sets. The results show that (1) learning entailments from positive and negative examples is a viable approach and (2) our model based on kernel methods is highly accurate and improves on the current state-of-the-art entailment systems.

In the future, we would like to study approaches to improve the computational complexity of our kernel function and to design approximated versions that are valid Mercer's kernels.

References

Steven Abney. 1996. Part-of-speech tagging and partial parsing. In G.Bloothooft K.Church, S.Young, editor, *Corpus-based methods in language and speech*. Kluwer academic publishers, Dordrecht.

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The II PASCAL RTE challenge. In *RTE Workshop*, Venice, Italy.

Samuel Bayer, John Burger, Lisa Ferro, John Henderson, and Alexander Yeh. 2005. MITRE's submissions to the eu PASCAL RTE challenge. In *Proceedings of the 1st RTE Workshop*, Southampton, UK.

Johan Bos and Katja Markert. 2005. Recognising textual entailment with logical inference. In *Proc. of HLT-EMNLP Conference*, Canada.

S. Boughorbel, J-P. Tarel, and F. Fleuret. 2004. Non-mercer kernel for svm object recognition. In *Proceedings of BMVC 2004*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. of the 1st NAACL*,Seattle, Washington.

Gennaro Chierchia and Sally McConnell-Ginet. 2001. *Meaning and Grammar: An introduction to Semantics*. MIT press, Cambridge, MA.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL02*.

Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, Ann Arbor, Michigan.

Ido Dagan and Oren Glickman. 2004. Probabilistic textual entailment: Generic applied modeling of language variability. In *Proceedings of the Workshop on Learning Methods for Text Understanding and Mining*, Grenoble, France.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The PASCAL RTE challenge. In *RTE Workshop*, Southampton, U.K.

Rodrigo de Salvo Braz, Roxana Girju, Vasin Punyakanok, Dan Roth, and Mark Sammons. 2005. An inference model for semantic entailment in natural language. In *Proc. of the RTE Workshop*, Southampton, U.K.

Oren Glickman, Ido Dagan, and Moshe Koppel. 2005. Web based probabilistic textual entailment. In *Proceedings of the 1st RTE Workshop*, Southampton, UK.

Bernard Haasdonk. 2005. Feature space interpretation of SVMs with indefinite kernels. *IEEE Trans Pattern Anal Mach Intell*.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proc. of the 15th CoLing*, Nantes, France.

Jay J. Jiang and David W. Conrath. 1997. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of the 10th ROCLING*, Tapei, Taiwan.

Thorsten Joachims. 1999. Making large-scale svm learning practical. In *Advances in Kernel Methods-Support Vector Learning*. MIT Press.

Milen Kouylekov and Bernardo Magnini. 2005. Tree edit distance for textual entailment. In *Proc. of the RANLP-2005*, Borovets, Bulgaria.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, November.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language learning. In *Proceedings of EACL'06*, Trento, Italy.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. Wordnet::similarity - measuring the relatedness of concepts. In *Proc. of 5th NAACL*, Boston, MA.

Vladimir Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

Annie Zaenen, Lauri Karttunen, and Richard Crouch. 2005. Local textual inference: Can it be defined or circumscribed? In *Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, Ann Arbor, Michigan.

An Improved Redundancy Elimination Algorithm for Underspecified Representations

Alexander Koller and **Stefan Thater**
Dept. of Computational Linguistics
Universität des Saarlandes, Saarbrücken, Germany
{koller,stth}@coli.uni-sb.de

Abstract

We present an efficient algorithm for the *redundancy elimination* problem: Given an underspecified semantic representation (USR) of a scope ambiguity, compute an USR with fewer mutually equivalent readings. The algorithm operates on underspecified chart representations which are derived from dominance graphs; it can be applied to the USRs computed by large-scale grammars. We evaluate the algorithm on a corpus, and show that it reduces the degree of ambiguity significantly while taking negligible runtime.

1 Introduction

Underspecification is nowadays the standard approach to dealing with scope ambiguities in computational semantics (van Deemter and Peters, 1996; Copestake et al., 2004; Egg et al., 2001; Blackburn and Bos, 2005). The basic idea behind it is to not enumerate all possible semantic representations for each syntactic analysis, but to derive a single compact *underspecified representation (USR)*. This simplifies semantics construction, and current algorithms support the efficient enumeration of the individual semantic representations from an USR (Koller and Thater, 2005b).

A major promise of underspecification is that it makes it possible, in principle, to rule out entire subsets of readings that we are not interested in wholesale, without even enumerating them. For instance, real-world sentences with scope ambiguities often have many readings that are semantically equivalent. Subsequent modules (e.g. for doing inference) will typically only be interested in one reading from each equivalence class, and all others could be deleted. This situation is illustrated by the following two (out of many) sentences from the Rondane treebank, which is distributed with the English Resource Grammar (ERG; Flickinger (2002)), a large-scale HPSG grammar of English.

(1) For travellers going to Finnmark there is a bus service from Oslo to Alta through Sweden. (Rondane 1262)

(2) We quickly put up the tents in the lee of a small hillside and cook for the first time in the open. (Rondane 892)

For the annotated syntactic analysis of (1), the ERG derives an USR with eight scope bearing operators, which results in a total of 3960 readings. These readings are all semantically equivalent to each other. On the other hand, the USR for (2) has 480 readings, which fall into two classes of mutually equivalent readings, characterised by the relative scope of “the lee of” and “a small hillside.”

In this paper, we present an algorithm for the *redundancy elimination* problem: Given an USR, compute an USR which has fewer readings, but still describes at least one representative of each equivalence class – without enumerating any readings. This algorithm makes it possible to compute the one or two representatives of the semantic equivalence classes in the examples, so subsequent modules don’t have to deal with all the other equivalent readings. It also closes the gap between the large number of readings predicted by the grammar and the intuitively perceived much lower degree of ambiguity of these sentences. Finally, it can be helpful for a grammar designer because it is much more feasible to check whether two readings are linguistically reasonable than 480. Our algorithm is applicable to arbitrary USRs (not just those computed by the ERG). While its effect is particularly significant on the ERG, which uniformly treats all kinds of noun phrases, including proper names and pronouns, as generalised quantifiers, it will generally help deal with spurious ambiguities (such as scope ambiguities between indef-

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 409–416,
Sydney, July 2006. ©2006 Association for Computational Linguistics

inites), which have been a ubiquitous problem in most theories of scope since Montague Grammar.

We model equivalence in terms of rewrite rules that permute quantifiers without changing the semantics of the readings. The particular USRs we work with are underspecified chart representations, which can be computed from dominance graphs (or USRs in some other underspecification formalisms) efficiently (Koller and Thater, 2005b). We evaluate the performance of the algorithm on the Rondane treebank and show that it reduces the median number of readings from 56 to 4, by up to a factor of 666.240 for individual USRs, while running in negligible time.

To our knowledge, our algorithm and its less powerful predecessor (Koller and Thater, 2006) are the first redundancy elimination algorithms in the literature that operate on the level of USRs. There has been previous research on *enumerating* only some representatives of each equivalence class (Vestre, 1991; Chaves, 2003), but these approaches don't maintain underspecification: After running their algorithms, they are left with a set of readings rather than an underspecified representation, i.e. we could no longer run other algorithms on an USR.

The paper is structured as follows. We will first define dominance graphs and review the necessary background theory in Section 2. We will then introduce our notion of equivalence in Section 3, and present the redundancy elimination algorithm in Section 4. In Section 5, we describe the evaluation of the algorithm on the Rondane corpus. Finally, Section 6 concludes and points to further work.

2 Dominance graphs

The basic underspecification formalism we assume here is that of *(labelled) dominance graphs* (Althaus et al., 2003). Dominance graphs are equivalent to leaf-labelled normal dominance constraints (Egg et al., 2001), which have been discussed extensively in previous literature.

Definition 1. A *(compact) dominance graph* is a directed graph $(V, E \uplus D)$ with two kinds of edges, *tree edges E* and *dominance edges D*, such that:

1. The graph (V,E) defines a collection of node disjoint trees of height 0 or 1. We call the trees in (V,E) the *fragments* of the graph.
2. If (v,v') is a dominance edge in D, then v is a hole and v' is a root. A node v is a *root* if v does not have incoming tree edges; otherwise, v is a *hole*.

A *labelled dominance graph* over a ranked signature Σ is a triple $G = (V, E \uplus D, L)$ such that $(V, E \uplus D)$ is a dominance graph and $L : V \rightsquigarrow \Sigma$ is a partial *labelling function* which assigns a node v a label with arity n iff v is a root with n outgoing tree edges. Nodes without labels (i.e. holes) must have outgoing dominance edges.

We will write $R(F)$ for the root of the fragment F, and we will typically just say "graph" instead of "labelled dominance graph".

An example of a labelled dominance graph is shown to the left of Fig. 1. Tree edges are drawn as solid lines, and dominance edges as dotted lines, directed from top to bottom. This graph can serve as an USR for the sentence "a representative of a company saw a sample" if we demand that the holes are "plugged" by roots while realising the dominance edges as dominance, as in the two *configurations* (of five) shown to the right. These configurations are trees that encode semantic representations of the sentence. We will freely read configurations as ground terms over the signature Σ.

2.1 Hypernormally connected graphs

Throughout this paper, we will only consider *hypernormally connected (hnc)* dominance graphs. Hnc graphs are equivalent to *chain-connected* dominance constraints (Koller et al., 2003), and are closely related to *dominance nets* (Niehren and Thater, 2003). Fuchss et al. (2004) have presented a corpus study that strongly suggests that all dominance graphs that are generated by current large-scale grammars are (or should be) hnc.

Technically, a graph G is hypernormally connected iff each pair of nodes is connected by a simple *hypernormal path* in G. A hypernormal path (Althaus et al., 2003) in G is a path in the undirected version G_u of G that does not use two dominance edges that are incident to the same hole.

Hnc graphs have a number of very useful structural properties on which this paper rests. One which is particularly relevant here is that we can predict in which way different fragments can dominate each other.

Definition 2. Let G be a hnc dominance graph. A fragment F_1 in G is called a *possible dominator* of another fragment F_2 in G iff it has exactly one hole h which is connected to $R(F_2)$ by a simple hy-

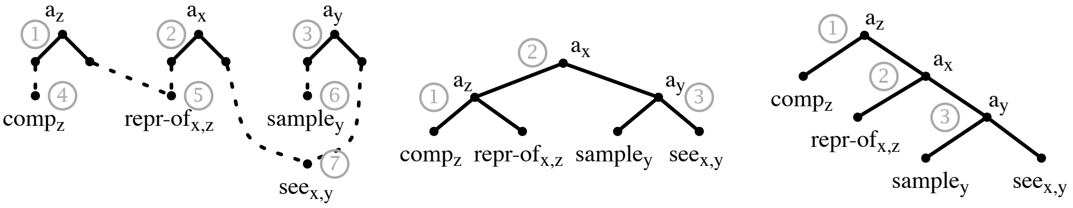


Figure 1: A dominance graph that represents the five readings of the sentence “a representative of a company saw a sample” (left) and two of its five configurations.

$$\begin{aligned}
\{1,2,3,4,5,6,7\} :& \langle 1, h_1 \mapsto \{4\}, h_2 \mapsto \{2,3,5,6,7\}\rangle \\
& \langle 2, h_3 \mapsto \{1,4,5\}, h_4 \mapsto \{3,6,7\}\rangle \\
& \langle 3, h_5 \mapsto \{5\}, h_6 \mapsto \{1,2,4,5,7\}\rangle \\
\{2,3,5,6,7\} :& \langle 2, h_3 \mapsto \{5\}, h_4 \mapsto \{3,6,7\}\rangle \\
& \langle 3, h_5 \mapsto \{6\}, h_6 \mapsto \{2,5,7\}\rangle \\
\{3,6,7\} :& \langle 3, h_5 \mapsto \{6\}, h_6 \mapsto \{7\}\rangle \\
\{2,5,7\} :& \langle 2, h_3 \mapsto \{5\}, h_4 \mapsto \{7\}\rangle \\
\{1,4,5\} :& \langle 1, h_1 \mapsto \{4\}, h_2 \mapsto \{5\}\rangle \\
\{1,2,4,5,7\} :& \langle 1, h_1 \mapsto \{4\}, h_2 \mapsto \{2,5,7\}\rangle \\
& \langle 2, h_3 \mapsto \{1,4,5\}, h_4 \mapsto \{7\}\rangle
\end{aligned}$$

Figure 2: The chart for the graph in Fig. 1.

pernormal path which doesn't use $R(F_1)$. We write $\mathsf{ch}(F_1, F_2)$ for this unique h.

Lemma 1 (Koller and Thater (2006))**.** Let F_1, F_2 be fragments in a hnc dominance graph G. If there is a configuration C of G in which $R(F_1)$ dominates $R(F_2)$, then F_1 is a possible dominator of F_2, and in particular $\mathsf{ch}(F_1, F_2)$ dominates $R(F_2)$ in C.

By applying this rather abstract result, we can derive a number of interesting facts about the example graph in Fig. 1. The fragments 1, 2, and 3 are possible dominators of all other fragments (and of each other), while the fragments 4 through 7 aren't possible dominators of anything (they have no holes); so 4 through 7 must be leaves in any configuration of the graph. In addition, if fragment 2 dominates fragment 3 in any configuration, then in particular the *right* hole of 2 will dominate the root of 3; and so on.

2.2 Dominance charts

Below we will not work with dominance graphs directly. Rather, we will use *dominance charts* (Koller and Thater, 2005b) as our USRs: they are more explicit USRs, which support a more fine-grained deletion of reading sets than graphs.

A dominance chart for the graph G is a mapping of weakly connected subgraphs of G to sets of *splits* (see Fig. 2), which describe possible ways of constructing configurations of the subgraph. A subgraph G' is assigned one split for each fragment F in G' which can be at the root of a configuration of G'. If the graph is hnc, removing F from the graph splits G' into a set of weakly connected components (wccs), each of which is connected to exactly one hole of F. We also record the wccs, and the hole to which each wcc belongs, in the split. In order to compute all configurations represented by a split, we can first compute recursively the configurations of each component; then we plug each combination of these sub-configurations into the appropriate holes of the root fragment. We define the configurations associated with a subgraph as the union over its splits, and those of the entire chart as the configurations associated with the complete graph.

Fig. 2 shows the dominance chart corresponding to the graph in Fig. 1. The chart represents exactly the configuration set of the graph, and is minimal in the sense that every subgraph and every split in the chart can be used in constructing some configuration. Such charts can be computed efficiently (Koller and Thater, 2005b) from a dominance graph, and can also be used to compute the configurations of a graph efficiently.

The example chart expresses that three fragments can be at the root of a configuration of the complete graph: 1, 2, and 3. The entry for the split with root fragment 2 tells us that removing 2 splits the graph into the subgraphs $\{1,4,5\}$ and $\{3,6,7\}$ (see Fig. 3). If we configure these two subgraphs recursively, we obtain the configurations shown in the third column of Fig. 3; we can then plug these sub-configurations into the appropriate holes of 2 and obtain a configuration for the entire graph.

Notice that charts can be exponentially larger than the original graph, but they are still exponentially smaller than the entire set of readings because common subgraphs (such as the graph $\{2,5,7\}$ in the example) are represented only once,

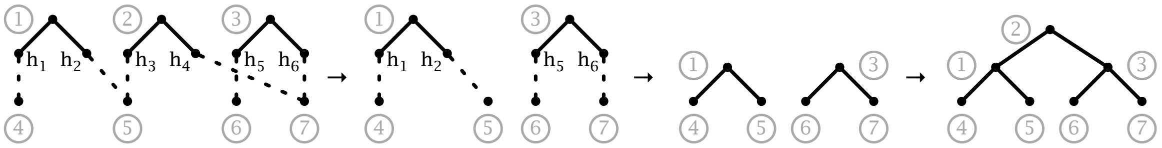


Figure 3: Extracting a configuration from a chart.

and are small in practice (see (Koller and Thater, 2005b) for an analysis). Thus the chart can still serve as an underspecified representation.

3 Equivalence

Now let's define equivalence of readings more precisely. Equivalence of semantic representations is traditionally defined as the relation between formulas (say, of first-order logic) which have the same interpretation. However, even first-order equivalence is an undecidable problem, and broad-coverage semantic representations such as those computed by the ERG usually have no well-defined model-theoretic semantics and therefore no concept of semantic equivalence.

On the other hand, we do not need to solve the full semantic equivalence problem, as we only want to compare formulas that are readings of the same sentence, i.e. different configurations of the same USR. Such formulas only differ in the way that the fragments are combined. We can therefore approximate equivalence by using a *rewrite system* that permutes fragments and defining equivalence of configurations as mutual rewritability as usual.

By way of example, consider again the two configurations shown in Fig. 1. We can obtain the second configuration from the (semantically equivalent) first one by applying the following rewrite rule, which rotates the fragments 1 and 2:

$$\mathsf{a}_x(\mathsf{a}_z(P,Q),R) \to \mathsf{a}_z(P,\mathsf{a}_x(Q,R)) \qquad (3)$$

Thus we take these two configurations to be equivalent with respect to the rewrite rule. (We could also have argued that the second configuration can be rewritten into the first by using the inverted rule.)

We formalise this rewriting-based notion of equivalence as follows. The definition uses the abbreviation $\overline{x_{[1,k)}}$ for the sequence $x_1,\ldots,x_{k-1}$, and $\overline{x_{(k,n]}}$ for $x_{k+1},\ldots,x_n$.

Definition 3. A *permutation system R* is a system of rewrite rules over the signature Σ of the following form:

$$f_1(\overline{x_{[1,i)}}, f_2(\overline{y_{[1,k)}}, z, \overline{y_{(k,m]}}), \overline{x_{(i,n]}}) \to f_2(\overline{y_{[1,k)}}, f_1(\overline{x_{[1,i)}}, z, \overline{x_{(i,n]}}), \overline{y_{(k,m]}})$$

The *permutability relation $P(R)$* is the binary relation $P(R) \subseteq (\Sigma \times \mathbb{N})^2$ which contains exactly the tuples $((f_1,i),(f_2,k))$ and $((f_2,k),(f_1,i))$ for each such rewrite rule. Two terms are *equivalent* with respect to R, $s \approx_R t$, iff there is a sequence of rewrite steps and inverse rewrite steps that rewrite s into t.

If G is a graph over Σ and R a permutation system, then we write $SC_R(G)$ for the set of equivalence classes $\mathrm{Conf}(G)/{\approx_R}$, where $\mathrm{Conf}(G)$ is the set of configurations of G.

The rewrite rule (3) above is an instance of this schema, as are the other three permutations of existential quantifiers. These rules approximate classical semantic equivalence of first-order logic, as they rewrite formulas into classically equivalent ones. Indeed, all five configurations of the graph in Fig. 1 are rewriting-equivalent to each other.

In the case of the semantic representations generated by the ERG, we don't have access to an underlying interpretation. But we can capture linguistic intuitions about the equivalence of readings in permutation rules. For instance, proper names and pronouns (which the ERG analyses as scope-bearers, although they can be reduced to constants without scope) can be permuted with anything. Indefinites and definites permute with each other if they occur in each other's *scope*, but not if they occur in each other's *restriction*; and so on.

4 Redundancy elimination

Given a permutation system, we can now try to get rid of readings that are equivalent to other readings. One way to formalise this is to enumerate exactly one representative of each equivalence class. However, after such a step we would be left with a collection of semantic representations rather than an USR, and could not use the USR for ruling out further readings. Besides, a naive algorithm which first enumerates all configurations would be prohibitively slow.

We will instead tackle the following *underspecified redundancy elimination* problem: Given an USR G, compute an USR G' with $\mathrm{Conf}(G') \subseteq \mathrm{Conf}(G)$ and $SC_R(G) = SC_R(G')$. We want $\mathrm{Conf}(G')$ to be as small as possible. Ideally, it would contain no two equivalent readings, but in practice we won't always achieve this kind of completeness. Our redundancy elimination algorithm will operate on a dominance chart and successively delete splits and subgraphs from the chart.

4.1 Permutable fragments

Because the algorithm must operate on USRs rather than configurations, it needs a way to predict from the USR alone which fragments can be permuted in configurations. This is not generally possible in unrestricted graphs, but for hnc graphs it is captured by the following criterion.

Definition 4. Let R be a permutation system. Two fragments F_1 and F_2 with root labels f_1 and f_2 in a hnc graph G are called *R-permutable* iff they are possible dominators of each other and $((f_1, \mathsf{ch}(F_1,F_2)),(f_2,\mathsf{ch}(F_2,F_1))) \in P(R)$.

For example, in Fig. 1, the fragments 1 and 2 are permutable, and indeed they can be permuted in any configuration in which one is the parent of the other. This is true more generally:

Lemma 2 (Koller and Thater (2006))**.** Let G be a hnc graph, F_1 and F_2 be R-permutable fragments with root labels f_1 and f_2, and C_1 any configuration of G of the form $C(f_1(\dots,f_2(\dots),\dots))$ (where C is the context of the subterm). Then C_1 can be R-rewritten into a tree C_2 of the form $C(f_2(\dots,f_1(\dots),\dots))$ which is also a configuration of G.

The proof uses the hn connectedness of G in two ways: in order to ensure that C_2 is still a configuration of G, and to make sure that F_2 is plugged into the correct hole of F_1 for a rule application (cf. Lemma 1). Note that $C_2 \approx_R C_1$ by definition.

4.2 The redundancy elimination algorithm

Now we can use permutability of fragments to define *eliminable splits*. Intuitively, a split of a subgraph G is eliminable if each of its configurations is equivalent to a configuration of some other split of G. Removing such a split from the chart will rule out some configurations; but it does not change the set of equivalence classes.

Definition 5. Let R be a permutation system. A split $S = (F,\dots,h_i \mapsto G_i,\dots)$ of a graph G is called *eliminable* in a chart Ch if some G_i contains a fragment F' such that (a) Ch contains a split S' of G with root fragment F', and (b) F' is R-permutable with F and all possible dominators of F' in G_i.

In Fig. 1, each of the three splits is eliminable. For example, the split with root fragment 1 is eliminable because the fragment 3 permutes both with 2 (which is the only possible dominator of 3 in the same wcc) and with 1 itself.

Proposition 3. Let Ch be a dominance chart, and let S be an eliminable split of a hnc subgraph. Then $SC(Ch) = SC(Ch - S)$.

Proof. Let C be an arbitrary configuration of $S = (F, h_1 \mapsto G_1, \dots, h_n \mapsto G_n)$, and let $F' \in G_i$ be the root fragment of the assumed second split S'.

Let $F_1,\dots,F_n$ be those fragments in C that are properly dominated by F and properly dominate F'. All of these fragments must be possible dominators of F', and all of them must be in G_i as well, so F' is permutable with each of them. F' must also be permutable with F. This means that we can apply Lemma 2 repeatedly to move F' to the root of the configuration, obtaining a configuration of S' which is equivalent to C. □

Notice that we didn't require that Ch must be the complete chart of a dominance graph. This means we can remove eliminable splits from a chart repeatedly, i.e. we can apply the following redundancy elimination algorithm:

```
REDUNDANCY-ELIMINATION($Ch$, $R$)
  for each split $S$ in $Ch$
      do if $S$ is eliminable with respect to $R$
            then remove $S$ from $Ch$
```

Prop. 3 shows that the algorithm is a correct algorithm for the underspecified redundancy elimination problem. The particular order in which eliminable splits are removed doesn't affect the correctness of the algorithm, but it may change the number of remaining configurations. The algorithm generalises an earlier elimination algorithm (Koller and Thater, 2006) in that the earlier algorithm required the existence of a *single* split which could be used to establish eliminability of *all* other splits of the same subgraph.

We can further optimise this algorithm by keeping track of how often each subgraph is referenced

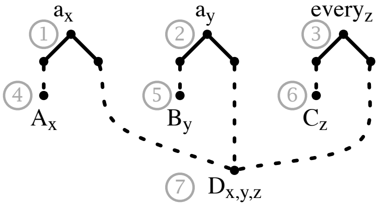


Figure 4: A graph for which the algorithm is not complete.

by the splits in the chart. Once a reference count drops to zero, we can remove the entry for this subgraph and all of its splits from the chart. This doesn't change the set of configurations of the chart, but may further reduce the chart size. The overall runtime for the algorithm is $O(n^2S)$, where S is the number of splits in Ch and n is the number of nodes in the graph. This is asymptotically not much slower than the runtime $O((n+m)S)$ it takes to compute the chart in the first place (where m is the number of edges in the graph).

4.3 Examples and discussion

Let's look at a run of the algorithm on the chart in Fig. 2. The algorithm can first delete the eliminable split with root 1 for the entire graph G. After this deletion, the splits for G with root fragments 2 and 3 are still eliminable; so we can e.g. delete the split for 3. At this point, only one split is left for G. The last split for a subgraph can never be eliminable, so we are finished with the splits for G. This reduces the reference count of some subgraphs (e.g. $\{2,3,5,6,7\}$) to 0, so we can remove these subgraphs too. The output of the algorithm is the chart shown below, which represents a single configuration (the one shown in Fig. 3).

$$\begin{aligned}\{1,2,3,4,5,6,7\} &: \langle 2, h_2 \mapsto \{1,4\}, h_4 \mapsto \{3,6,7\}\rangle\\ \{1,4\} &: \langle 1, h_1 \mapsto \{4\}\rangle\\ \{3,6,7\} &: \langle 3, h_5 \mapsto \{6\}, h_6 \mapsto \{7\}\rangle\end{aligned}$$

In this case, the algorithm achieves *complete reduction*, in the sense that the final chart has no two equivalent configurations. It remains complete for all variations of the graph in Fig. 1 in which some or all existential quantifiers are replaces by universal quantifiers. This is an improvement over our earlier algorithm (Koller and Thater, 2006), which computed a chart with four configurations for the graph in which 1 and 2 are existential and 3 is universal, as opposed to the three equivalence classes of this graph's configurations.

However, the present algorithm still doesn't achieve complete reduction for all USRs. One example is shown in Fig. 4. This graph has six configurations in four equivalence classes, but no split of the whole graph is eliminable. The algorithm will delete a split for the subgraph $\{1,2,4,5,7\}$, but the final chart will still have five, rather than four, configurations. A complete algorithm would have to recognise that $\{1,3,4,6,7\}$ and $\{2,3,5,6,7\}$ have splits (for 1 and 2, respectively) that lead to equivalent configurations and delete one of them. But it is far from obvious how such a non-local decision could be made efficiently, and we leave this for future work.

5 Evaluation

In this final section, we evaluate the the effectiveness and efficiency of the elimination algorithm: We run it on USRs from a treebank and measure how many readings are redundant, to what extent the algorithm eliminates this redundancy, and how much time it takes to do this.

Resources. The experiments are based on the Rondane corpus, a Redwoods (Oepen et al., 2002) style corpus which is distributed with the English Resource Grammar (Flickinger, 2002). The corpus contains analyses for 1076 sentences from the tourism domain, which are associated with USRs based upon Minimal Recursion Semantics (MRS). The MRS representations are translated into dominance graphs using the open-source `utool` tool (Koller and Thater, 2005a), which is restricted to MRS representations whose translations are hnc. By restricting ourselves to such MRSs, we end up with a data set of 999 dominance graphs. The average number of scope bearing operators in the data set is 6.5, and the median number of readings is 56.

We then defined a (rather conservative) rewrite system R_{ERG} for capturing the permutability relation of the quantifiers in the ERG. This amounted to 34 rule schemata, which are automatically expanded to 494 rewrite rules.

Experiment: Reduction. We first analysed the extent to which our algorithm eliminated the redundancy of the USRs in the corpus. We computed dominance charts for all USRs, ran the algorithm on them, and counted the number of configurations of the reduced charts. We then compared these numbers against a baseline and an upper bound. The upper bound is the true number of

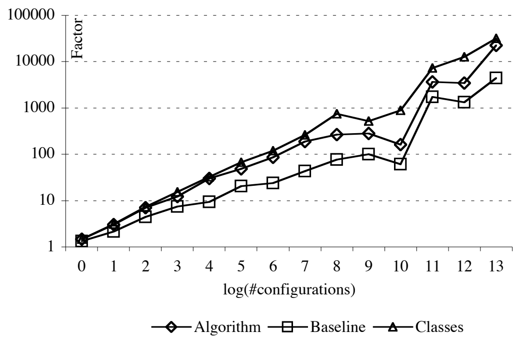


Figure 5: Mean reduction factor on Rondane.

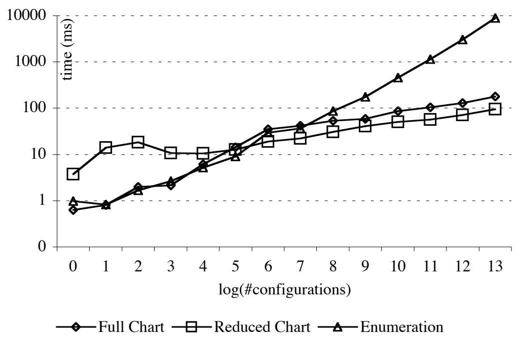


Figure 7: Mean runtimes.

equivalence classes with respect to R_{ERG}; for efficiency reasons we could only compute this number for USRs with up to 500.000 configurations (95 % of the data set). The baseline is given by the number of readings that remain if we replace proper names and pronouns by constants and variables, respectively. This simple heuristic is easy to compute, and still achieves nontrivial redundancy elimination because proper names and pronouns are quite frequent (28% of the noun phrase occurrences in the data set). It also shows the degree of non-trivial scope ambiguity in the corpus.

For each measurement, we sorted the USRs according to the number N of configurations, and grouped USRs according to the natural logarithm of N (rounded down) to obtain a logarithmic scale.

First, we measured the mean reduction factor for each $\log(N)$ class, i.e. the ratio of the number of all configurations to the number of remaining configurations after redundancy elimination (Fig. 5). The upper-bound line in the figure shows that there is a great deal of redundancy in the USRs in the data set. The average performance of our algorithm is close to the upper bound and much better than the baseline. For USRs with fewer than $e^8 = 2980$ configurations (83 % of the data set), the mean reduction factor of our algorithm is above 86 % of the upper bound. The median number of configurations for the USRs in the whole data set is 56, and the median number of equivalence classes is 3; again, the median number of configurations of the reduced charts is very close to the upper bound, at 4 (baseline: 8). The highest reduction factor for an individual USR is 666.240.

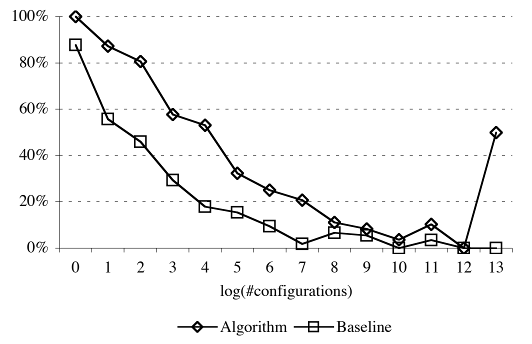


Figure 6: Percentage of USRs for which the algorithm and the baseline achieve complete reduction.

We also measured the ratio of USRs for which the algorithm achieves complete reduction (Fig. 6): The algorithm is complete for 56 % of the USRs in the data set. It is complete for 78 % of the USRs with fewer than $e^5 = 148$ configurations (64 % of the data set), and still complete for 66 % of the USRs with fewer than e^8 configurations.

Experiment: Efficiency. Finally, we measured the runtime of the elimination algorithm. The runtime of the elimination algorithm is generally comparable to the runtime for computing the chart in the first place. However, in our experiments we used an optimised version of the elimination algorithm, which computes the reduced chart directly from a dominance graph by checking each split for eliminability *before* it is added to the chart. We compare the performance of this algorithm to the baseline of computing the complete chart. For comparison, we have also added the time it takes to enumerate all configurations of the graph, as a lower bound for any algorithm that computes the equivalence classes based on the full set of configurations. Fig. 7 shows the mean runtimes for each $\log(N)$ class, on the USRs with less than one million configurations (958 USRs).

As the figure shows, the asymptotic runtimes for computing the complete chart and the reduced chart are about the same, whereas the time for enumerating all configurations grows much faster. (Note that the runtime is reported on a logarithmic scale.) For USRs with many configurations, computing the reduced chart actually takes *less* time on average than computing the complete chart because the chart-filling algorithm is called on fewer subgraphs. While the reduced-chart algorithm seems to be slower than the complete-chart one for USRs with less than e^5 configurations, these runtimes remain below 20 milliseconds on average, and the measurements are thus quite unreliable. In summary, we can say that there is no overhead for redundancy elimination in practice.

6 Conclusion

We presented an algorithm for redundancy elimination on underspecified chart representations. This algorithm successively deletes *eliminable splits* from the chart, which reduces the set of described readings while making sure that at least one representative of each original equivalence class remains. Equivalence is defined with respect to a certain class of rewriting systems; this definition approximates semantic equivalence of the described formulas and fits well with the underspecification setting. The algorithm runs in polynomial time in the size of the chart.

We then evaluated the algorithm on the Rondane corpus and showed that it is useful in practice: the median number of readings drops from 56 to 4, and the maximum individual reduction factor is 666.240. The algorithm achieves complete reduction for 56% of all sentences. It does this in negligible runtime; even the most difficult sentences in the corpus are reduced in a matter of seconds, whereas the enumeration of all readings would take about a year. This is the first corpus evaluation of a redundancy elimination in the literature.

The algorithm improves upon previous work (Koller and Thater, 2006) in that it eliminates more splits from the chart. It is an improvement over earlier algorithms for enumerating irredundant readings (Vestre, 1991; Chaves, 2003) in that it maintains underspecifiedness; note that these earlier papers never made any claims with respect to, or evaluated, completeness.

There are a number of directions in which the present algorithm could be improved. We are currently pursuing some ideas on how to improve the completeness of the algorithm further. It would also be worthwhile to explore heuristics for the order in which splits of the same subgraph are eliminated. The present work could be extended to allow equivalence with respect to arbitrary rewrite systems. Most generally, we hope that the methods developed here will be useful for defining other elimination algorithms, which take e.g. full world knowledge into account.

References

E. Althaus, D. Duchier, A. Koller, K. Mehlhorn, J. Niehren, and S. Thiel. 2003. An efficient graph algorithm for dominance constraints. *Journal of Algorithms*, 48:194–219.

P. Blackburn and J. Bos. 2005. *Representation and Inference for Natural Language. A First Course in Computational Semantics*. CSLI Publications.

R. P. Chaves. 2003. Non-redundant scope disambiguation in underspecified semantics. In *Proc. 8th ESSLLI Student Session*.

A. Copestake, D. Flickinger, C. Pollard, and I. Sag. 2004. Minimal recursion semantics: An introduction. *Journal of Language and Computation*. To appear.

M. Egg, A. Koller, and J. Niehren. 2001. The Constraint Language for Lambda Structures. *Logic, Language, and Information*, 10.

D. Flickinger. 2002. On building a more efficient grammar by exploiting types. In J. Tsujii S. Oepen, D. Flickinger and H. Uszkoreit, editors, *Collaborative Language Engineering*. CSLI Publications, Stanford.

R. Fuchss, A. Koller, J. Niehren, and S. Thater. 2004. Minimal recursion semantics as dominance constraints: Translation, evaluation, and analysis. In *Proc. of the 42nd ACL*.

A. Koller and S. Thater. 2005a. Efficient solving and exploration of scope ambiguities. In *ACL-05 Demonstration Notes*, Ann Arbor.

A. Koller and S. Thater. 2005b. The evolution of dominance constraint solvers. In *Proceedings of the ACL-05 Workshop on Software*, Ann Arbor.

A. Koller and S. Thater. 2006. Towards a redundancy elimination algorithm for underspecified descriptions. In *Proc. 5th Intl. Workshop on Inference in Computational Semantics (ICoS-5)*.

A. Koller, J. Niehren, and S. Thater. 2003. Bridging the gap between underspecification formalisms: Hole semantics as dominance constraints. In *Proc. 10th EACL*.

J. Niehren and S. Thater. 2003. Bridging the gap between underspecification formalisms: Minimal recursion semantics as dominance constraints. In *Proc. of the 41st ACL*.

S. Oepen, K. Toutanova, S. Shieber, C. Manning, D. Flickinger, and T. Brants. 2002. The LinGO Redwoods treebank: Motivation and preliminary applications. In *Proceedings of COLING'02*.

K. van Deemter and S. Peters. 1996. *Semantic Ambiguity and Underspecification*. CSLI, Stanford.

E. Vestre. 1991. An algorithm for generating non-redundant quantifier scopings. In *Proc. of the Fifth EACL*, Berlin.

Integrating Syntactic Priming into an Incremental Probabilistic Parser, with an Application to Psycholinguistic Modeling

Amit Dubey and **Frank Keller** and **Patrick Sturt**
Human Communication Research Centre, University of Edinburgh
2 Buccleuch Place, Edinburgh EH8 9LW, UK
{amit.dubey,patrick.sturt,frank.keller}@ed.ac.uk

Abstract

The psycholinguistic literature provides evidence for syntactic priming, i.e., the tendency to repeat structures. This paper describes a method for incorporating priming into an incremental probabilistic parser. Three models are compared, which involve priming of rules between sentences, within sentences, and within coordinate structures. These models simulate the reading time advantage for parallel structures found in human data, and also yield a small increase in overall parsing accuracy.

1 Introduction

Over the last two decades, the psycholinguistic literature has provided a wealth of experimental evidence for *syntactic priming*, i.e., the tendency to repeat syntactic structures (e.g., Bock, 1986). Most work on syntactic priming has been concerned with sentence production; however, recent studies also demonstrate a preference for structural repetition in human parsing. This includes the so-called *parallelism effect* demonstrated by Frazier et al. (2000): speakers processes coordinated structures more quickly when the second conjunct repeats the syntactic structure of the first conjunct.

Two alternative accounts of the parallelism effect have been proposed. Dubey et al. (2005) argue that the effect is simply an instance of a pervasive syntactic priming mechanism in human parsing. They provide evidence from a series of corpus studies which show that parallelism is not limited to co-ordination, but occurs in a wide range of syntactic structures, both within and between sentences, as predicted if a general priming mechanism is assumed. (They also show this effect is stronger in coordinate structures, which could explain Frazier et al.'s (2000) results.)

Frazier and Clifton (2001) propose an alternative account of the parallelism effect in terms of a *copying mechanism*. Unlike priming, this mechanism is highly specialized and only applies to coordinate structures: if the second conjunct is encountered, then instead of building new structure, the language processor simply copies the structure of the first conjunct; this explains why a speed-up is observed if the two conjuncts are parallel. If the copying account is correct, then we would expect parallelism effects to be restricted to coordinate structures and not to apply in other contexts.

This paper presents a parsing model which implements both the priming mechanism and the copying mechanism, making it possible to compare their predictions on human reading time data. Our model also simulates other important aspects of human parsing: (i) it is broad-coverage, i.e., it yields accurate parses for unrestricted input, and (ii) it processes sentences incrementally, i.e., on a word-by-word basis. This general modeling framework builds on probabilistic accounts of human parsing as proposed by Jurafsky (1996) and Crocker and Brants (2000).

A priming-based parser is also interesting from an engineering point of view. To avoid sparse data problems, probabilistic parsing models make strong independence assumptions; in particular, they generally assume that sentences are independent of each other, in spite of corpus evidence for structural repetition between sentences. We therefore expect a parsing model that includes structural repetition to provide a better fit with real corpus data, resulting in better parsing performance. A simple and principled approach to handling structure re-use would be to use adaptation probabilities for probabilistic grammar rules (Church, 2000), analogous to cache probabilities used in caching language models (Kuhn and de Mori, 1990). This is the approach we will pursue in this paper.

Dubey et al. (2005) present a corpus study that demonstrates the existence of parallelism in corpus data. This is an important precondition for understanding the parallelism effect; however, they

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 417–424,
Sydney, July 2006. ©2006 Association for Computational Linguistics

do not develop a parsing model that accounts for the effect, which means they are unable to evaluate their claims against experimental data. The present paper overcomes this limitation. In Section 2, we present a formalization of the priming and copying models of parallelism and integrate them into an incremental probabilistic parser. In Section 3, we evaluate this parser against reading time data taken from Frazier et al.'s (2000) parallelism experiments. In Section 4, we test the engineering aspects of our model by demonstrating that a small increase in parsing accuracy can be obtained with a parallelism-based model. Section 5 provides an analysis of the performance of our model, focusing on the role of the distance between prime and target.

2 Priming Models

We propose three models designed to capture the different theories of structural repetition discussed above. To keep our model as simple as possible, each formulation is based on an unlexicalized probabilistic context free grammar (PCFG). In this section, we introduce the models and discuss the novel techniques used to model structural similarity. We also discuss the design of the probabilistic parser used to evaluate the models.

2.1 Baseline Model

The unmodified PCFG model serves as the Baseline. A PCFG assigns trees probabilities by treating each rule expansion as conditionally independent given the parent node. The probability of a rule $LHS \rightarrow RHS$ is estimated as:

$$P(RHS|LHS) = \frac{c(LHS \rightarrow RHS)}{c(LHS)}$$

2.2 Copy Model

The first model we introduce is a probabilistic variant of Frazier and Clifton's (2001) copying mechanism: it models parallelism in coordination and nothing else. This is achieved by assuming that the default operation upon observing a coordinator (assumed to be anything with a *CC* tag, e.g., 'and') is to copy the full subtree of the preceding coordinate sister. Copying impacts on how the parser works (see Section 2.5), and in a probabilistic setting, it also changes the probability of trees with parallel coordinated structures. If coordination is present, the structure of the second item is either identical to the first, or it is not.[1] Let us call the probability of having a copied tree as p_{ident}. This value may be estimated directly from a corpus using the formula

$$\hat{p}_{ident} = \frac{c_{ident}}{c_{total}}$$

Here, c_{ident} is the number of coordinate structures in which the two conjuncts have the same internal structure and c_{total} is the total number of coordinate structures. Note we assume there is only one parameter p_{ident} applicable everywhere (i.e., it has the same value for all rules).

How is this used in a PCFG parser? Let t_1 and t_2 represent, respectively, the first and second coordinate sisters and let $P_{PCFG}(t)$ be the PCFG probability of an arbitrary subtree t.

Because of the independence assumptions of the PCFG, we know that $p_{ident} \gg P_{PCFG}(t)$. One way to proceed would be to assign a probability of p_{ident} when structures match, and $(1 - p_{ident}) \cdot P_{PCFG}(t_2)$ when structures do not match. However, some probability mass is lost this way: there is a nonzero PCFG probability (namely, $P_{PCFG}(t_1)$) that the structures match.

In other words, we may have identical subtrees in two different ways: either due to a copy operation, or due to a PCFG derivation. If p_{copy} is the probability of a copy operation, we can write this fact more formally as: $p_{ident} = P_{PCFG}(t_1) + p_{copy}$.

Thus, if the structures do match, we assign the second sister a probability of:

$$p_{copy} + P_{PCFG}(t_1)$$

If they do not match, we assign the second conjunct the following probability:

$$\frac{1 - P_{PCFG}(t_1) - p_{copy}}{1 - P_{PCFG}(t_1)} \cdot P_{PCFG}(t_2)$$

This accounts for both a copy mismatch and a PCFG derivation mismatch, and assures the probabilities still sum to one. These probabilities for parallel and non-parallel coordinate sisters, therefore, gives us the basis of the Copy model.

This leaves us with the problem of finding an estimate for p_{copy}. This value is approximated as:

$$\hat{p}_{copy} = \hat{p}_{ident} - \frac{1}{|T_2|} \sum_{t \in T_2} P_{PCFG}(t)$$

In this equation, T_2 is the set of all second conjuncts.

2.3 Between Model

While the Copy model limits itself to parallelism in coordination, the next two models simulate structural priming in general. Both are similar in design, and are based on a simple insight: we may

[1] The model only considers two-item coordination or the last two sisters of multiple-item coordination.

condition a PCFG rule expansion on whether the rule occurred in some previous context. If *Prime* is a binary-valued random variable denoting if a rule occurred in the context, then we define:

$$P(RHS|LHS,Prime) = \frac{c(LHS \rightarrow RHS,Prime)}{c(LHS,Prime)}$$

This is essentially an instantiation of Church's (2000) adaptation probability, albeit with PCFG rules instead of words. For our first model, this context is the previous sentence. Thus, the model can be said to capture the degree to which rule use is primed between sentences. We henceforth refer to this as the Between model. Following the convention in the psycholinguistic literature, we refer to a rule use in the previous sentence as a 'prime', and a rule use in the current sentence as the 'target'. Each rule acts once as a target (i.e., the event of interest) and once as a prime. We may classify such adapted probabilities into 'positive adaptation', i.e., the probability of a rule given the rule occurred in the preceding sentence, and 'negative adaptation', i.e., the probability of a rule given that the rule did not occur in the preceding sentence.

2.4 Within Model

Just as the Between model conditions on rules from the previous sentence, the Within sentence model conditions on rules from earlier in the current sentence. Each rule acts once as a target, and possibly several times as a prime (for each subsequent rule in the sentence). A rule is considered 'used' once the parser passes the word on the leftmost corner of the rule. Because the Within model is finer grained than the Between model, it can be used to capture the parallelism effect in coordination. In other words, this model could explain parallelism in coordination as an instance of a more general priming effect.

2.5 Parser

As our main purpose is to build a psycholinguistic model of structure repetition, the most important feature of the parsing model is to build structures incrementally.[2]

Reading time experiments, including the parallelism studies of Frazier et al. (2000), make word-by-word measurements of the time taken to read sentences. Slower reading times are known to be correlated with processing difficulty, and faster reading times (as is the case with parallel structures) are correlated with processing ease. A probabilistic parser may be considered to be a sentence processing model via a 'linking hypothesis', which links the parser's word-by-word behavior to human reading behavior. We discuss this topic in more detail in Section 3. At this point, it suffices to say that we require a parser which has the prefix property, i.e., which parses incrementally, from left to right.

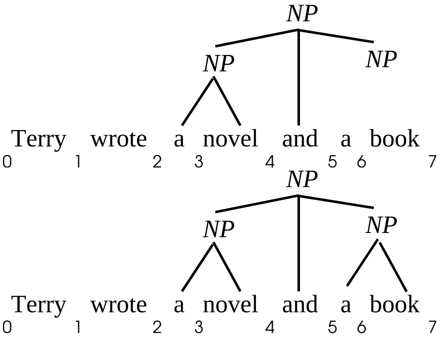

NP
NP
NP
Terry wrote a novel and a book
0 1 2 3 4 5 6 7
NP
NP
NP
Terry wrote a novel and a book
0 1 2 3 4 5 6 7

Figure 1: Upon encountering a coordinator, the copy model copies the most likely first conjunct.

Therefore, we use an Earley-style probabilistic parser, which outputs Viterbi parses (Stolcke, 1995). We have two versions of the parser: one which parses exhaustively, and a second which uses a variable width beam, pruning any edges whose merit is $\frac{1}{2000}$ of the best edge. The merit of an edge is its inside probability times a prior $P(LHS)$ times a lookahead probability (Roark and Johnson, 1999). To speed up parsing time, we right binarize the grammar,[3] remove empty nodes, coindexation and grammatical functions. As our goal is to create the simplest possible model which can nonetheless model experimental data, we do not make any tree modification designed to improve accuracy (as, e.g., Klein and Manning 2003).

The approach used to implement the Copy model is to have the parser copy the subtree of the first conjunct whenever it comes across a *CC* tag. Before copying, though, the parser looks ahead to check if the part-of-speech tags after the *CC* are equivalent to those inside the first conjunct. The copying model is visualized in Figure 1: the top panel depicts a partially completed edge upon seeing a *CC* tag, and the second panel shows the completed copying operation. It should be clear that

[2] In addition to incremental parsing, a characteristic some of psycholinguistic models of sentence comprehension is to parse deterministically. While we can compute the best incremental analysis at any point, ours models do not parse deterministically. However, following the principles of rational analysis (Anderson, 1991), our goal is not to mimic the human parsing *mechanism*, but rather to create a model of human parsing *behavior*.

[3] We found that using an unbinarized grammar did not alter the results, at least in the exhaustive parsing case.

the copy operation gives the most probable subtree in a given span. To illustrate this, consider Figure 1. If the most likely *NP* between spans 2 and 7 does not involve copying (i.e. only standard PCFG rule derivations), the parser will find it using normal rule derivations. If it does involve copying, for this particular rule, it must involve the most likely *NP* subtree from spans 2 to 3. As we parse incrementally, we are guaranteed to have found this edge, and can use it to construct the copied conjunct over spans 5 to 7 and therefore the whole co-ordinated *NP* from spans 2 to 7.

To simplify the implementation of the copying operation, we turn off right binarization so that the constituent before and after a coordinator are part of the same rule, and therefore accessible from the same edge. This makes it simple to calculate the new probability: construct the copied subtree, and decide where to place the resulting edge on the chart.

The Between and Within models require a cache of recently used rules. This raises two dilemmas. First, in the Within model, keeping track of full contextual history is incompatible with chart parsing. Second, whenever a parsing error occurs, the accuracy of the contextual history is compromised. As we are using a simple unlexicalized parser, such parsing errors are probably quite frequent.

We handle the first problem by using one single parse as an approximation of the history. The more realistic choice for this single parse is the best parse so far according to the parser. Indeed, this is the approach we use for our main results in Section 3. However, because of the second problem noted above, in Section 4, we simulated the context by filling the cache with rules from the correct tree. In the Between model, these are the rules of the correct parse of the previous tree; in the Within model, these are the rules used in the correct parse at points up to (but not including) the current word.

3 Human Reading Time Experiment

In this section, we test our models by applying them to experimental reading time data. Frazier et al. (2000) reported a series of experiments that examined the parallelism preference in reading. In one of their experiments, they monitored subjects' eye-movements while they read sentences like (1):

(1) a. Hilda noticed a strange man and a tall woman when she entered the house.
    b. Hilda noticed a man and a tall woman when she entered the house.

They found that total reading times were faster on the phrase *tall woman* in (1a), where the coordinated noun phrases are parallel in structure, compared with in (1b), where they are not.

There are various approaches to modeling processing difficulty using a probabilistic approach. One possibility is to use an incremental parser with a beam search or an *n*-best approach. Processing difficulty is predicted at points in the input string where the current best parse is replaced by an alternative derivation (Jurafsky, 1996; Crocker and Brants, 2000). An alternative is to keep track of all derivations, and predict difficulty at points where there is a large change in the shape of the probability distribution across adjacent parsing states (Hale, 2001). A third approach is to calculate the forward probability (Stolcke, 1995) of the sentence using a PCFG. Low probabilities are then predicted to correspond to high processing difficulty. A variant of this third approach is to assume that processing difficulty is correlated with the (log) probability of the best parse (Keller, 2003). This final formulation is the one used for the experiments presented in this paper.

3.1 Method

The item set was adapted from that of Frazier et al. (2000). The original two relevant conditions of their experiment (1a,b) differ in terms of length. This results in a confound in the PCFG framework, because longer sentences tend to result in lower probabilities (as the parses tend to involve more rules). To control for such length differences, we adapted the materials by adding two extra conditions in which the relation between syntactic parallelism and length was reversed. This resulted in the following four conditions:

(2) a. DT JJ NN and DT JJ NN (parallel)
    Hilda noticed a tall man and a strange woman when she entered the house.
    b. DT NN and DT JJ NN (non-parallel)
    Hilda noticed a man and a strange woman when she entered the house.
    c. DT JJ NN and DT NN (non-parallel)
    Hilda noticed a tall man and a woman when she entered the house.
    d. DT NN and DT NN (parallel)
    Hilda noticed a man and a woman when she entered the house.

In order to account for Frazier et al.'s parallelism effect a probabilistic model should predict a greater difference in probability between (2a) and (2b) than between (2c) and (2d) (i.e., (2a)−(2b) > (2c)−(2d)). This effect will not be confounded with length, because the relation between length and parallelism is reversed between (2a,b) and (2c,d). We added 8 items to the original Frazier et al. materials, resulting in a new set of 24 items similar to (2).

We tested three of our PCFG-based models on all 24 sets of 4 conditions. The models were the Baseline, the Within and the Copy models, trained exactly as described above. The Between model was not tested as the experimental stimuli were presented without context. Each experimental sentence was input as a sequence of correct POS tags, and the log probability estimate of the best parse was recorded.

3.2 Results and Discussion

Table 1 shows the mean log probabilities estimated by the models for the four conditions, along with the relevant differences between parallel and non-parallel conditions.

Both the Within and the Copy models show a parallelism advantage, with this effect being much more pronounced for the Copy model than the Within model. To evaluate statistical significance, the two differences for each item were compared using a Wilcoxon signed ranks test. Significant results were obtained both for the Within model ($N = 24$, $Z = 1.67$, $p < .05$, one-tailed) and for the Copy model ($N = 24$, $Z = 4.27$, $p < .001$, one-tailed). However, the effect was much larger for the Copy model, a conclusion which is confirmed by comparing the differences of differences between the two models ($N = 24$, $Z = 4.27$, $p < .001$, one-tailed). The Baseline model was not evaluated statistically, because by definition it predicts a constant value for (2a)−(2b) and (2c)−(2d) across all items. This is simply a consequence of the PCFG independence assumption, coupled with the fact that the four conditions of each experimental item differ only in the occurrences of two NP rules.

The results show that the approach taken here can be successfully applied to the modeling of experimental data. In particular, both the Within and the Copy models show statistically reliable parallelism effects. It is not surprising that the copy model shows a large parallelism effect for the Frazier et al. (2000) items, as it was explicitly designed to prefer structurally parallel conjuncts. The more interesting result is the parallelism effect found for the Within model, which shows that such an effect can arise from a more general probabilistic priming mechanism.

4 Parsing Experiment

In the previous section, we were able to show that the Copy and Within models are able to account for human reading-time performance for parallel coordinate structures. While this result alone is sufficient to claim success as a psycholinguistic model, it has been argued that more realistic psycholinguistic models ought to also exhibit high accuracy and broad-coverage, both crucial properties of the human parsing mechanism (e.g., Crocker and Brants, 2000).

This should not be difficult: our starting point was a PCFG, which already has broad coverage behavior (albeit with only moderate accuracy). However, in this section we explore what effects our modifications have to overall coverage, and, perhaps more interestingly, to parsing accuracy.

4.1 Method

The models used here were the ones introduced in Section 2 (which also contains a detailed description of the parser that we used to apply the models). The corpus used for both training and evaluation is the Wall Street Journal part of the Penn Treebank. We use sections 1–22 for training, section 0 for development and section 23 for testing. Because the Copy model posits coordinated structures whenever POS tags match, parsing efficiency decreases if POS tags are not predetermined. Therefore, we assume POS tags as input, using the gold-standard tags from the treebank (following, e.g., Roark and Johnson 1999).

4.2 Results and Discussion

Table 2 lists the results in terms of F-score on the test set.[4] Using exhaustive search, the baseline model achieves an F-score of 73.3, which is comparable to results reported for unlexicalized incremental parsers in the literature (e.g. the RB1 model of Roark and Johnson, 1999). All models exhibit a small decline in performance when beam search is used. For the Within model we observe a slight improvement in performance over the baseline, both for the exhaustive search and the beam

[4] Based on a χ^2 test on precision and recall, all results are statistically different from each other. The Copy model actually performs slightly better than the Baseline in the exhaustive case.

Model	para: (2a)	non-para: (2b)	non-para: (2c)	para: (2d)	(2a)−(2b)	(2c)−(2d)
Baseline	−33.47	−32.37	−32.37	−31.27	−1.10	−1.10
Within	−33.28	−31.67	−31.70	−29.92	−1.61	−1.78
Copy	−16.18	−27.22	−26.91	−15.87	11.04	−11.04

Table 1: Mean log probability estimates for Frazier et al (2000) items

	Exhaustive Search		Beam Search		Beam + Coord		Fixed Coverage	
Model	F-score	Coverage	F-score	Coverage	F-score	Coverage	F-score	Coverage
Baseline	73.3	100	73.0	98.0	73.1	98.1	73.0	97.5
Within	73.6	100	73.4	98.4	73.0	98.5	73.4	97.5
Between	71.6	100	71.7	98.7	71.5	99.0	71.8	97.5
Copy	73.3	100	–	–	73.0	98.1	73.1	97.5

Table 2: Parsing results for the Within, Between, and Copy model compared to a PCFG baseline.

search conditions. The Between model, however, resulted in a decrease in performance.

We also find that the Copy model performs at the baseline level. Recall that in order to simplify the implementation of the copying, we had to disable binarization for coordinate constituents. This means that quaternary rules were used for coordination ($X \rightarrow X_1\ CC\ X_2\ X'$), while normal binary rules ($X \rightarrow Y\ X'$) were used everywhere else. It is conceivable that this difference in binarization explains the difference in performance between the Between and Within models and the Copy model when beam search was used. We therefore also state the performance for Between and Within models with binarization limited to non-coordinate structures in the column labeled 'Beam + Coord' in Table 2. The pattern of results, however, remains the same.

The fact that coverage differs between models poses a problem in that it makes it difficult to compare the F-scores directly. We therefore compute separate F-scores for just those sentences that were covered by all four models. The results are reported in the 'Fixed Coverage' column of Table 2. Again, we observe that the copy model performs at baseline level, while the Within model slightly outperforms the baseline, and the Between model performs worse than the baseline. In Section 5 below we will present an error analysis that tries to investigate why the adaptation models do not perform as well as expected.

Overall, we find that the modifications we introduced to model the parallelism effect in humans have a positive, but small, effect on parsing accuracy. Nonetheless, the results also indicate the success of both the Copy and Within approaches to parallelism as psycholinguistic models: a modification primarily useful for modeling human behavior has no negative effects on computational measures of coverage or accuracy.

5 Distance Between Rule Uses

Although both the Within and Copy models succeed at the main task of modeling the parallelism effect, the parsing experiments in Section 4 showed mixed results with respect to F-scores: a slight increase in F-score was observed for the Within model, but the Between model performed below the baseline. We therefore turn to an error analysis, focusing on these two models.

Recall that the Within and Between models estimate two probabilities for a rule, which we have been calling the positive adaptation (the probability of a rule when the rule is also in the history), and the negative adaptation (the probability of a rule when the rule is *not* in the history). While the effect is not always strong, we expect positive adaptation to be higher than negative adaptation (Dubey et al., 2005). However, this is not always the case.

In the Within model, for example, the rule $NP \rightarrow DT\ JJ\ NN$ has a higher negative than positive adaptation (we will refer to such rules as 'negatively adapted'). The more common rule $NP \rightarrow DT\ NN$ has a higher positive adaptation ('positively adapted'). Since the latter is three times more common, this raises a concern: what if adaptation is an artifact of frequency? This 'frequency' hypothesis posits that a rule recurring in a sentence is simply an artifact of the its higher frequency. The frequency hypothesis could explain an interesting fact: while the majority of rules tokens have positive adaptation, the majority of rule types have negative adaptation. An important corollary of the frequency hypothesis is that we would not expect to find a bias towards local rule re-uses.

```
Iterate through the treebank
  Remember how many words each constituent spans
Iterate through the treebank
  Iterate through each tree
    Upon finding a constituent spanning 1-4 words
      Swap it with a randomly chosen constituent
      of 1-4 words
      Update the remembered size of the swapped
      constituents and their subtrees
Iterate through the treebank 4 more times
  Swap constituents of size 5-9, 10-19, 20-35
  and 35+ words, respectively
```

Figure 2: The treebank randomization algorithm

Nevertheless, the $NP \rightarrow DT\ JJ\ NN$ rule is an exception: most negatively adapted rules have very low frequencies. This raises the possibility that sparse data is the cause of the negatively adapted rules. This makes intuitive sense: we need many rule occurrences to accurately estimate positive or negative adaptation.

We measure the distribution of rule use to explore if negatively adapted rules owe more to frequency effects or to sparse data. This distributional analysis also serves to measure 'decay' effects in structural repetition. The decay effect in priming has been observed elsewhere (Szmrecsanyi, 2005), and suggests that positive adaptation is higher the closer together two rules are.

5.1 Method

We investigate the dispersion of rules by plotting histograms of the distance between subsequent rule uses. The basic premise is to look for evidence of an early peak or skew, which suggests rule re-use. To ensure that the histogram itself is not sensitive to sparse data problems, we group all rules into two categories: those which are positively adapted, and those which are negatively adapted.

If adaptation is not due to frequency alone, we would expect the histograms for both positively and negatively adapted rules to be skewed towards local rule repetition. Detecting a skew requires a baseline without repetition. We propose the concept of 'randomizing' the treebank to create such a baseline. The randomization algorithm is described in Figure 2. The algorithm entails swapping subtrees, taking care that small subtrees are swapped first (otherwise large chunks would be swapped at once, preserving a great deal of context). This removes local effects, giving a distribution due frequency alone.

After applying the randomization algorithm to the treebank, we may construct the distance histogram for both the non-randomized and randomized treebanks. The distance between two occurrences of a rule is calculated as the number of words between the first word on the left corner of each rule. A special case occurs if a rule expansion invokes another use of the same rule. When this happens, we do not count the distance between the first and second expansion. However, the second expansion is still remembered as the most recent.

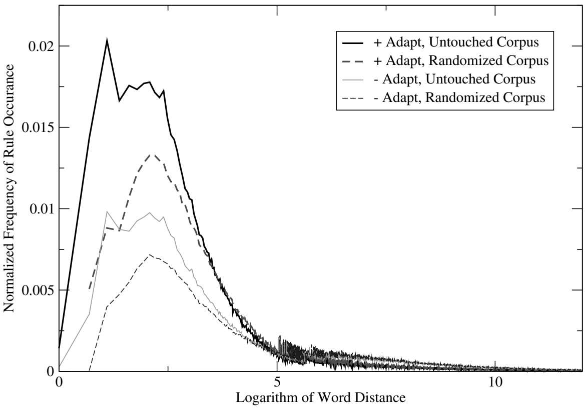


Figure 3: Log of number of words between rule invocations

We group rules into those that have a higher positive adaptation and those that have a higher negative adaptation. We then plot a histogram of rule re-occurrence distance for both groups, in both the non-randomized and randomized corpora.

5.2 Results and Discussion

The resulting plot for the Within model is shown in Figure 3. For both the positive and negatively adapted rules, we find that randomization results in a lower, less skewed peak, and a longer tail. We conclude that rules tend to be repeated close to one another more than we expect by chance, even for negatively adapted rules. This is evidence against the frequency hypothesis, and in favor of the sparse data hypothesis. This means that the small size of the increase in F-score we found in Section 4 is not due to the fact that the adaption is just an artifact of rule frequency. Rather, it can probably be attributed to data sparseness.

Note also that the shape of the histogram provides a decay curve. Speculatively, we suggest that this shape could be used to parameterize the decay effect and therefore provide an estimate for adaptation which is more robust to sparse data. However, we leave the development of such a smoothing function to future research.

6 Conclusions and Future Work

The main contribution of this paper has been to show that an incremental parser can simulate syntactic priming effects in human parsing by incorporating probability models that take account of previous rule use. Frazier et al. (2000) argued that the best account of their observed parallelism advantage was a model in which structure is copied from one coordinate sister to another. Here, we explored a probabilistic variant of the copy mechanism, along with two more general models based on within- and between-sentence priming. Although the copy mechanism provided the strongest parallelism effect in simulating the human reading time data, the effect was also successfully simulated by a general within-sentence priming model. On the basis of simplicity, we therefore argue that it is preferable to assume a simpler and more general mechanism, and that the copy mechanism is not needed. This conclusion is strengthened when we turn to consider the performance of the parser on the standard Penn Treebank test set: the Within model showed a small increase in F-score over the PCFG baseline, while the copy model showed no such advantage.[5]

All the models we proposed offer a broad-coverage account of human parsing, not just a limited model on a hand-selected set of examples, such as the models proposed by Jurafsky (1996) and Hale (2001) (but see Crocker and Brants 2000).

A further contribution of the present paper has been to develop a methodology for analyzing the (re-)use of syntactic rules over time in a corpus. In particular, we have defined an algorithm for randomizing the constituents of a treebank, yielding a baseline estimate of chance repetition.

In the research reported in this paper, we have adopted a very simple model based on an unlexicalized PCFG. In the future, we intend to explore the consequences of introducing lexicalization into the parser. This is particularly interesting from the point of view of psycholinguistic modeling, because there are well known interactions between lexical repetition and syntactic priming, which require lexicalization for a proper treatment. Future work will also involve the use of smoothing to increase the benefit of priming for parsing accuracy. The investigations reported in Section 5 provide a basis for estimating the smoothing parameters.

References

Anderson, John. 1991. Cognitive architectures in a rational analysis. In K. VanLehn, editor, *Architectures for Intelligence*, Lawrence Erlbaum Associates, Hillsdale, N.J., pages 1–24.

Bock, J. Kathryn. 1986. Syntactic persistence in language production. *Cognitive Psychology* 18:355–387.

Church, Kenneth W. 2000. Empirical estimates of adaptation: the chance of two Noriegas is closer to $p/2$ than p^2. In *Proceedings of the 17th Conference on Computational Linguistics*. Saarbrücken, Germany, pages 180–186.

Crocker, Matthew W. and Thorsten Brants. 2000. Wide-coverage probabilistic sentence processing. *Journal of Psycholinguistic Research* 29(6):647–669.

Dubey, Amit, Patrick Sturt, and Frank Keller. 2005. Parallelism in coordination as an instance of syntactic priming: Evidence from corpus-based modeling. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*. Vancouver, pages 827–834.

Frazier, Lyn, Alan Munn, and Chuck Clifton. 2000. Processing coordinate structures. *Journal of Psycholinguistic Research* 29(4):343–370.

Frazier, Lynn and Charles Clifton. 2001. Parsing coordinates and ellipsis: Copy α. *Syntax* 4(1):1–22.

Hale, John. 2001. A probabilistic Earley parser as a psycholinguistic model. In *Proceedings of the 2nd Conference of the North American Chapter of the Association for Computational Linguistics*. Pittsburgh, PA.

Jurafsky, Daniel. 1996. A probabilistic model of lexical and syntactic access and disambiguation. *Cognitive Science* 20(2):137–194.

Keller, Frank. 2003. A probabilistic parser as a model of global processing difficulty. In R. Alterman and D. Kirsh, editors, *Proceedings of the 25th Annual Conference of the Cognitive Science Society*. Boston, pages 646–651.

Klein, Dan and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo, Japan, pages 423–430.

Kuhn, Roland and Renate de Mori. 1990. A cache-based natural language model for speech recognition. *IEEE Transanctions on Pattern Analysis and Machine Intelligence* 12(6):570–583.

Roark, Brian and Mark Johnson. 1999. Efficient probabilistic top-down and left-corner parsing. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. pages 421–428.

Stolcke, Andreas. 1995. An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics* 21(2):165–201.

Szmrecsanyi, Benedikt. 2005. Creatures of habit: A corpus-linguistic analysis of persistence in spoken English. *Corpus Linguistics and Linguistic Theory* 1(1):113–149.

[5] The broad-coverage parsing experiment speaks against a ‘facilitation’ hypothesis, i.e., that the copying and priming mechanisms work together. However, a full test of this (e.g., by combining the two models) is left to future research.

A Fast, Accurate Deterministic Parser for Chinese

Mengqiu Wang Kenji Sagae Teruko Mitamura
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
{mengqiu,sagae,teruko}@cs.cmu.edu

Abstract

We present a novel classifier-based deterministic parser for Chinese constituency parsing. Our parser computes parse trees from bottom up in one pass, and uses classifiers to make shift-reduce decisions. Trained and evaluated on the standard training and test sets, our best model (using stacked classifiers) runs in linear time and has labeled precision and recall above 88% using gold-standard part-of-speech tags, surpassing the best published results. Our SVM parser is 2-13 times faster than state-of-the-art parsers, while producing more accurate results. Our Maxent and DTree parsers run at speeds 40-270 times faster than state-of-the-art parsers, but with 5-6% losses in accuracy.

1 Introduction and Background

Syntactic parsing is one of the most fundamental tasks in Natural Language Processing (NLP). In recent years, Chinese syntactic parsing has also received a lot of attention in the NLP community, especially since the release of large collections of annotated data such as the Penn Chinese Treebank (Xue et al., 2005). Corpus-based parsing techniques that are successful for English have been applied extensively to Chinese. Traditional statistical approaches build models which assign probabilities to every possible parse tree for a sentence. Techniques such as dynamic programming, beam-search, and best-first-search are then employed to find the parse tree with the highest probability. The massively ambiguous nature of wide-coverage statistical parsing,coupled with cubic-time (or worse) algorithms makes this approach too slow for many practical applications.

Deterministic parsing has emerged as an attractive alternative to probabilistic parsing, offering accuracy just below the state-of-the-art in syntactic analysis of English, but running in linear time (Sagae and Lavie, 2005; Yamada and Matsumoto, 2003; Nivre and Scholz, 2004). Encouraging results have also been shown recently by Cheng et al. (2004; 2005) in applying deterministic models to Chinese dependency parsing.

We present a novel classifier-based deterministic parser for Chinese constituency parsing. In our approach, which is based on the shift-reduce parser for English reported in (Sagae and Lavie, 2005), the parsing task is transformed into a succession of classification tasks. The parser makes one pass through the input sentence. At each parse state, it consults a classifier to make shift/reduce decisions. The parser then commits to a decision and enters the next parse state. Shift/reduce decisions are made deterministically based on the local context of each parse state, and no backtracking is involved. This process can be viewed as a greedy search where only one path in the whole search space is considered. Our parser produces both dependency and constituent structures, but in this paper we will focus on constituent parsing.

By separating the classification task from the parsing process, we can take advantage of many machine learning techniques such as classifier ensemble. We conducted experiments with four different classifiers: support vector machines (SVM), Maximum-Entropy (Maxent), Decision Tree (DTree) and memory-based learning (MBL). We also compared the performance of three different classifier ensemble approaches (simple voting, classifier stacking and meta-classifier).

Our best model (using stacked classifiers) runs in linear time and has labeled precision and recall above 88% using gold-standard part-of-speech tags, surpassing the best published results (see Section 5). Our SVM parser is 2-13 times faster than state-of-the-art parsers, while produc-

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 425–432,
Sydney, July 2006. ©2006 Association for Computational Linguistics

ing more accurate results. Our Maxent and DTree parsers are 40-270 times faster than state-of-the-art parsers, but with 5-6% losses in accuracy.

2 Deterministic parsing model

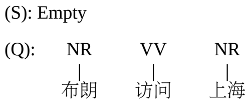
Like other deterministic parsers, our parser assumes input has already been segmented and tagged with part-of-speech (POS) information during a preprocessing step[1]. The main data structures used in the parsing algorithm are a queue and a stack. The input word-POS pairs to be processed are stored in the queue. The stack holds the partial parse trees that are built during parsing. A parse state is represented by the content of the stack and queue.

The classifier makes shift/reduce decisions based on contextual features that represent the parse state. A shift action removes the first item on the queue and puts it onto the stack. A reduce action is in the form of Reduce-{Binary|Unary}-X, where {Binary|Unary} denotes whether one or two items are to be removed from the stack, and X is the label of a new tree node that will be dominating the removed items. Because a reduction is either unary or binary, the resulting parse tree will only have binary and/or unary branching nodes.

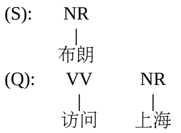
Parse trees are also lexicalized to produce dependency structures. For lexicalization, we used the same head-finding rules reported in (Bikel, 2004). With this additional information, reduce actions are now in the form of Reduce-{Binary |Unary}-X-Direction. The "Direction" tag gives information about whether to take the head-node of the left subtree or the right subtree to be the head of the new tree, in the case of binary reduction. A simple transformation process as described in (Sagae and Lavie, 2005) is employed to convert between arbitrary branching trees and binary trees. This transformation breaks multi-branching nodes down into binary-branching nodes by inserting temporary nodes; temporary nodes are collapsed and removed when we transform a binary tree back into a multi-branching tree.

The parsing process succeeds when all the items in the queue have been processed and there is only one item (the final parse tree) left on the stack. If the classifier returns a shift action when there are no items left on the queue, or a reduce action when there are no items on the stack, the parser fails. In this case, the parser simply combines all the items on the stack into one IP node, and outputs this as a partial parse. Sagae and Lavie (2005) have shown that this algorithm has linear time complexity, assuming that classification takes constant time. The next example illustrates the process for the input "布朗 (Brown) 访问 (visits) 上海 (Shanghai)" that is tagged with the POS sequence "NR (Proper Noun) VV (Verb) NR (Proper Noun)".

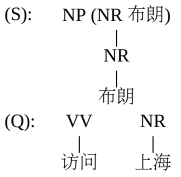
1. In the initial parsing state, the stack (S) is empty, and the queue (Q) holds word and POS tag pairs for the input sentence.



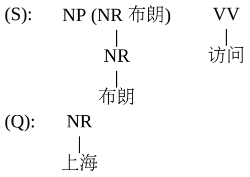
2. The first action item that the classifier gives is a shift action.



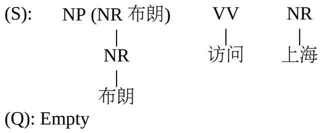
3. The next action is a reduce-Unary-NP, which means reducing the first item on the stack to a NP node. Node (NR 布朗) becomes the head of the new NP node and this information is marked by brackets. The new parse state is:



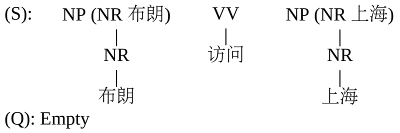
4. The next action is shift.



5. The next action is again shift.



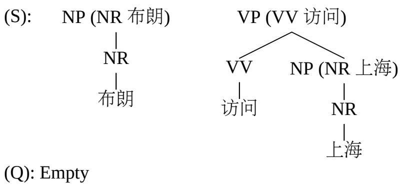
6. The next action is reduce-Unary-NP.



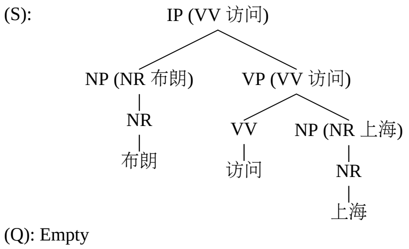
7. The next action is reduce-Binary-VP-Left. The node (VV 访问) will be the head of the

[1] We constructed our own POS tagger based on SVM; see Section 3.3.

new VP node.



8. The next action is reduce-Binary-IP-Right. Since after the action is performed, there will be only one tree node(IP) left on the stack and no items on the queue, this is the final action. The final state is:



3 Classifiers and Feature Selection

Classification is the key component of our parsing model. We conducted experiments with four different types of classifiers.

3.1 Classifiers

Support Vector Machine: Support Vector Machine is a discriminative classification technique which solves the binary classification problem by finding a hyperplane in a high dimensional space that gives the maximum soft margin, based on the Structural Risk Minimization Principle. We used the TinySVM toolkit (Kudo and Matsumoto, 2000), with a degree 2 polynomial kernel. To train a multi-class classifier, we used the one-against-all scheme.

Maximum-Entropy Classifier: In a Maximum-entropy model, the goal is to estimate a set of parameters that would maximize the entropy over distributions that satisfy certain constraints. These constraints will force the model to best account for the training data (Ratnaparkhi, 1999). Maximum-entropy models have been used for Chinese character-based parsing (Fung et al., 2004; Luo, 2003) and POS tagging (Ng and Low, 2004). In our experiments, we used Le's Maxent toolkit (Zhang, 2004). This implementation uses the Limited-Memory Variable Metric method for parameter estimation. We trained all our models using 300 iterations with no event cut-off, and a Gaussian prior smoothing value of 2. Maxent classifiers output not only a single class label, but also a number of possible class labels and their associated probability estimate.

Decision Tree Classifier: Statistical decision tree is a classic machine learning technique that has been extensively applied to NLP. For example, decision trees were used in the SPATTER system (Magerman, 1994) to assign probability distribution over the space of possible parse trees. In our experiment, we used the C4.5 decision tree classifier, and ignored lexical features whose counts were less than 7.

Memory-Based Learning: Memory-Based Learning approaches the classification problem by storing training examples explicitly in memory, and classifying the current case by finding the most similar stored cases (using k-nearest-neighbors). We used the TiMBL toolkit (Daelemans et al., 2004) in our experiment, with $k = 5$.

3.2 Feature selection

For each parse state, a set of features are extracted and fed to each classifier. Features are distributionally-derived or linguistically-based, and carry the context of a particular parse state. When input to the classifier, each feature is treated as a contextual predicate which maps an outcome and a context to $true, false$ value.

The specific features used with the classifiers are listed in Table 1.

Sun and Jurafsky (2003) studied the distributional property of rhythm in Chinese, and used the rhythmic feature to augment a PCFG model for a practical shallow parsing task. This feature has the value 1, 2 or 3 for monosyllabic, bi-syllabic or multi-syllabic nouns or verbs. For noun and verb phrases, the feature is defined as the number of words in the phrase. Sun and Jurafsky found that in NP and VP constructions there are strong constraints on the word length for verbs and nouns (a kind of rhythm), and on the number of words in a constituent. We employed these same rhythmic features to see whether this property holds for the Penn Chinese Treebank data, and if it helps in the disambiguation of phrase types. Experiments show that this feature does increase classification accuracy of the SVM model by about 1%.

In both Chinese and English, there are punctuation characters that come in pairs (e.g., parentheses). In Chinese, such pairs are more frequent (quotes, single quotes, and book-name marks). During parsing, we note how many opening punc-

1	A Boolean feature indicates if a closing punctuation is expected or not.
2	A Boolean value indicates if the queue is empty or not.
3	A Boolean feature indicates whether there is a comma separating S(1) and S(2) or not.
4	Last action given by the classifier, and number of words in S(1) and S(2).
5	Headword and its POS of S(1), S(2), S(3) and S(4), and word and POS of Q(1), Q(2), Q(3) and Q(4).
6	Nonterminal label of the root of S(1) and S(2), and number of punctuations in S(1) and S(2).
7	Rhythmic features and the linear distance between the head-words of the S(1) and S(2).
8	Number of words found so far to be dependents of the head-words of S(1) and S(2).
9	Nonterminal label, POS and headword of the immediate left and right child of the root of S(1) and S(2).
10	Most recently found word and POS pair that is to the left of the head-word of S(1) and S(2).
11	Most recently found word and POS pair that is to the right of the head-word of S(1) and S(2).

Table 1: Features for classification

tuations we have seen on the stack. If the number is odd, then feature 2 will have value 1, otherwise 0. A boolean feature is used to indicate whether or not an odd number of opening punctuations have been seen and a closing punctuation is expected; in this case the feature gives a strong hint to the parser that all the items in the queue before the closing punctuation, and the items on the stack after the opening punctuation should be under a common constituent node which begins and ends with the two punctuations.

3.3 POS tagging

In our parsing model, POS tagging is treated as a separate problem and it is assumed that the input has already been tagged with POS. To compare with previously published work, we evaluated the parser performance on automatically tagged data. We constructed a simple POS tagger using an SVM classifier. The tagger makes two passes over the input sentence. The first pass extracts features from the two words and POS tags that came before the current word, the two words following the current word, and the current word itself (the length of the word, whether the word contains numbers, special symbols that separates foreign first and last names, common Chinese family names, western alphabets or dates). Then the tag is assigned to the word according to SVM classifier's output. In the second pass, additional features such as the POS tags of the two words following the current word, and the POS tag of the current word (assigned in the first pass) are used. This tagger had a measured precision of 92.5% for sentences $\leq$ 40 words.

4 Experiments

We performed experiments using the Penn Chinese Treebank. Sections 001-270 (3484 sentences, 84,873 words) were used for training, 271-300 (348 sentences, 7980 words) for development, and 271-300 (348 sentences, 7980 words) for testing. The whole dataset contains 99629 words, which is about 1/10 of the size of the English Penn Treebank. Standard corpus preparation steps were done prior to parsing, so that empty nodes were removed, and the resulting A over A unary rewrite nodes are collapsed. Functional labels of the nonterminal nodes are also removed, but we did not relabel the punctuations, unlike in (Jiang, 2004). Bracket scoring was done by the EVALB program[2], and preterminals were not counted as constituents. In all our experiments, we used labeled recall (LR), labeled precision (LP) and F1 score (harmonic mean of LR and LP) as our evaluation metrics.

4.1 Results of different classifiers

Table 2 shows the classification accuracy and parsing accuracy of the four different classifiers on the development set for sentences $\leq$ 40 words, with gold-standard POS tagging. The runtime (Time) of each model and number of failed parses (Fail) are also shown.

	Classification	Parsing Accuracy				
Model	Accuracy	LR	LP	F1	Fail	Time
SVM	**94.3%**	**86.9%**	**87.9%**	**87.4%**	**0**	3m 19s
Maxent	92.6%	84.1%	85.2%	84.6%	5	0m 21s
DTree1	92.0%	78.8%	80.3%	79.5%	42	**0m 12s**
DTree2	N/A	81.6%	83.6%	82.6%	30	0m 18s
MBL	90.6%	74.3%	75.2%	74.7%	2	16m 11s

Table 2: Comparison of different classifier models' parsing accuracies on development set for sentences $\leq$ 40 words, with gold-standard POS

For the DTree learner, we experimented with two different classification strategies. In our first approach, the classification is done in a single stage (DTree1). The learner is trained for a multi-

[2] http://nlp.cs.nyu.edu/evalb/

class classification problem where the class labels include shift and all possible reduce actions. But this approach yielded a lot of parse failures (42 out of 350 sentences failed during parsing, and partial parse tree was returned). These failures were mostly due to false shift actions in cases where the queue is empty. To alleviate this problem, we broke the classification process down to two stages (DTree2). A first stage classifier makes a binary decision on whether the action is shift or reduce. If the output is reduce, a second-stage classifier decides which reduce action to take. Results showed that breaking down the classification task into two stages increased overall accuracy, and the number of failures was reduced to 30.

The SVM model achieved the highest classification accuracy and the best parsing results. It also successfully parsed all sentences. The Maxent model's classification error rate (7.4%) was 30% higher than the error rate of the SVM model (5.7%), and its F1 (84.6%) was 3.2% lower than SVM model's F1 (87.4%). But Maxent model was about 9.5 times faster than the SVM model. The DTree classifier achieved 81.6% LR and 83.6% LP. The MBL model did not perform well; although MBL and SVM differed in accuracy by only about 3 percent, the parsing results showed a difference of more than 10 percent. One possible explanation for the poor performance of the MBL model is that all the features we used were binary features, and memory-based learner is known to work better with multivalue features than binary features in natural language learning tasks (van den Bosch and Zavrel, 2000).

In terms of speed and accuracy trade-off, there is a 5.5% trade-off in F1 (relative to SVM's F1) for a roughly 14 times speed-up between SVM and two-stage DTree. Maxent is more balanced in the sense that its accuracy was slightly lower (3.2%) than SVM, and was just about as fast as the two-stage DTree on the development set. The high speed of the DTree and Maxent models make them very attractive in applications where speed is more critical than accuracy. While the SVM model takes more CPU time, we show in Section 5 that when compared to existing parsers, SVM achieves about the same or higher accuracy but is at least twice as fast.

Using gold-standard POS tagging, the best classifier model (SVM) achieved LR of 87.2% and LP of 88.3%, as shown in Table 4. Both measures surpass the previously known best results on parsing using gold-standard tagging. We also tested the SVM model using data automatically tagged by our POS tagger, and it achieved LR of 78.1% and LP of 81.1% for sentences $\leq$ 40 words, as shown in Table 3.

4.2 Classifier Ensemble Experiments

Classifier ensemble by itself has been a fruitful research direction in machine learning in recent years. The basic idea in classifier ensemble is that combining multiple classifiers can often give significantly better results than any single classifier alone. We experimented with three different classifier ensemble strategies: classifier stacking, meta-classifier, and simple voting.

Using the SVM classifier's results as a baseline, we tested these approaches on the development set. In classifier stacking, we collect the outputs from Maxent, DTree and TiMBL, which are all trained on a separate dataset from the training set (section 400-650 of the Penn Chinese Treebank, smaller than the original training set). We use their classification output as features, in addition to the original feature set, to train a new SVM model on the original training set. We achieved LR of 90.3% and LP of 90.5% on the development set, a 3.4% and 2.6% improvement in LR and LP, respectively. When tested on the test set, we gained 1% improvement in F1 when gold-standard POS tagging is used. When tested with automatic tagging, we achieved a 0.5% improvement in F1. Using Bikel's significant tester with 10000 times random shuffle, the p-value for LR and LP are 0.008 and 0.457, respectively. The increase in recall is statistically significant, and it shows classifier stacking can improve performance.

On the other hand, we did not find meta-classification and simple voting very effective. In simple voting, we make the classifiers to vote in each step for every parse action. The F1 of simple voting method is downgraded by 5.9% relative to SVM model's F1. By analyzing the inter-agreement among classifiers, we found that there were no cases where Maxent's top output and DTree's output were both correct and SVM's output was wrong. Using the top output from Maxent and DTree directly does not seem to be complementary to SVM.

In the meta-classifier approach, we first collect the output from each classifier trained on sec-

MODEL	$\leq$ 40 words				$\leq$ 100 words				Unlimited			
	LR	LP	F1	POS	LR	LP	F1	POS	LR	LP	F1	POS
Bikel & Chiang 2000	76.8%	77.8%	77.3%	-	73.3%	74.6%	74.0%	-	-	-	-	-
Levy & Manning 2003	79.2%	78.4%	78.8%	-	-	-	-	-	-	-	-	-
Xiong et al. 2005	78.7%	80.1%	79.4%	-	-	-	-	-	-	-	-	-
Bikel's Thesis 2004	78.0%	81.2%	79.6%	-	74.4%	78.5%	76.4%	-	-	-	-	-
Chiang & Bikel 2002	78.8%	81.1%	79.9%	-	75.2%	78.0%	76.6%	-	-	-	-	-
Jiang's Thesis 2004	80.1%	82.0%	81.1%	92.4%	-	-	-	-	-	-	-	-
Sun & Jurafsky 2004	**85.5**%	**86.4**%	**85.9**%	-	-	-	-	-	**83.3**%	**82.2**%	**82.7**%	-
DTree model	71.8%	76.9%	74.4%	92.5%	69.2%	74.5%	71.9%	92.2%	68.7%	74.2%	71.5%	92.1%
SVM model	78.1%	81.1%	79.6%	92.5%	75.5%	**78.5**%	77.0%	92.2%	75.0%	78.0%	76.5%	92.1%
Stacked classifier model	79.2%	81.1%	80.1%	92.5%	**76.7**%	78.4%	**77.5**%	92.2%	76.2%	78.0%	77.1%	92.1%

Table 3: Comparison with related work on the test set using automatically generated POS

tion 1-210 (roughly 3/4 of the entire training set). Then specifically for Maxent, we collected the top output as well as its associated probability estimate. Then we used the outputs and probability estimate as features to train an SVM classifier that makes a decision on which classifier to pick. Meta-classifier results did not change at all from our baseline. In fact, the meta-classifier always picked SVM as its output. This agrees with our observation for the simple voting case.

5 Comparison with Related Work

Bikel and Chiang (2000) constructed two parsers using a lexicalized PCFG model that is based on Collins' model 2 (Collins, 1999), and a statistical Tree-adjoining Grammar(TAG) model. They used the same train/development/test split, and achieved LR/LP of 76.8%/77.8%. In Bikel's thesis (2004), the same Collins emulation model was used, but with tweaked head-finding rules. Also a POS tagger was used for assigning tags for unseen words. The refined model achieved LR/LP of 78.0%/81.2%. Chiang and Bikel (2002) used inside-outside unsupervised learning algorithm to augment the rules for finding heads, and achieved an improved LR/LP of 78.8%/81.1%. Levy and Manning (2003) used a factored model that combines an unlexicalized PCFG model with a dependency model. They achieved LR/LP of 79.2%/78.4% on a different test/development split. Xiong et al. (2005) used a similar model to the BBN's model in (Bikel and Chiang, 2000), and augmented the model by semantic categorical information and heuristic rules. They achieved LR/LP of 78.7%/80.1%. Hearne and Way (2004) used a Data-Oriented Parsing (DOP) approach that was optimized for top-down computation. They achieved F1 of 71.3 on a different test and training set. Jiang (2004) reported LR/LP of 80.1%/82.0% on sentences $\leq$ 40 words (results not available for sentences $\leq$ 100 words) by applying Collins' parser to Chinese. In Sun and Jurafsky (2004)'s work on Chinese shallow semantic parsing, they also applied Collin's parser to Chinese. They reported up-to-date the best parsing performance on Chinese Treebank. They achieved LR/LP of 85.5%/86.4% on sentences $\leq$ 40 words, and LR/LP of 83.3%/82.2% on sentences $\leq$ 100 words, far surpassing all other previously reported results. Luo (2003) and Fung et al. (2004) addressed the issue of Chinese text segmentation in their work by constructing character-based parsers. Luo integrated segmentation, POS tagging and parsing into one maximum-entropy framework. He achieved a F1 score of 81.4% in parsing. But the score was achieved using 90% of the 250K-CTB (roughly 2.5 times bigger than our training set) for training and 10% for testing. Fung et al.(2004) also took the maximum-entropy modeling approach, but augmented by transformation-based learning. They used the standard training and testing split. When tested with gold-standard segmentation, they achieved a F1 score of 79.56%, but POS-tagged words were treated as constituents in their evaluation.

In comparison with previous work, our parser's accuracy is very competitive. Compared to Jiang's work and Sun and Jurafsky's work, the classifier ensemble model of our parser is lagging behind by 1% and 5.8% in F1, respectively. But compared to all other works, our classifier stacking model gave better or equal results for all three measures. In particular, the classifier ensemble model and SVM model of our parser achieved second and third highest LP, LR and F1 for sentences $\leq$ 100 words as shown in Table 3. (Sun and Jurafsky did not report results on sentences $\leq$ 100 words, but it is worth noting that out of all the test sentences, only 2 sentences have length > 100).

Jiang (2004) and Bikel (2004)[3] also evaluated their parsers on the test set for sentences ≤ 40 words, using gold-standard POS tagged input. Our parser gives significantly better results as shown in Table 4. The implication of this result is two-fold. On one hand, it shows that if POS tagging accuracy can be increased, our parser is likely to benefit more than the other two models; on the other hand, it also indicates that our deterministic model is less resilient to POS errors. Further detailed analysis is called for, to study the extent to which POS tagging errors affects the deterministic parsing model.

Model	LR	LP	F1
Bikel's Thesis 2004	80.9%	84.5%	82.7%
Jiang's Thesis 2004	84.5%	88.0%	86.2%
DTree model	80.5%	83.9%	82.2%
Maxent model	81.4%	82.8%	82.1%
SVM model	87.2%	**88.3%**	87.8%
Stacked classifier model	**88.3%**	88.1%	**88.2%**

Table 4: Comparison with related work on the test set for sentence ≤ 40 words, using gold-standard POS

To measure efficiency, we ran two publicly available parsers (Levy and Manning's PCFG parser (2003) and Bikel's parser (2004)) on the standard test set and compared the runtime[4]. The runtime of these parsers are shown in *minute:second* format in Table 5. Our SVM model is more than 2 times faster than Levy and Manning's parser, and more than 13 times faster than Bikel's parser. Our DTree model is 40 times faster than Levy and Manning's parser, and 270 times faster than Bikel's parser. Another advantage of our parser is that it does not take as much memory as these other parsers do. In fact, none of the models except MBL takes more than 60 megabytes of memory at runtime. In comparison, Levy and Manning's PCFG parser requires more than 400 mega-bytes of memory when parsing long sentences (70 words or longer).

6 Discussion and future work

One unique attraction of this deterministic parsing framework is that advances in machine learning field can be directly applied to parsing, which opens up lots of possibilities for continuous improvements, both in terms of accuracy and efficiency. For example, in this paper we experimented with one method of simple voting. An alternative way of doing simple voting is to let the parsers vote on membership of constituents after each parser has produced its own parse tree (Henderson and Brill, 1999), instead of voting at each step during parsing.

Model	runtime
Bikel	54m 6s
Levy & Manning	8m 12s
Our DTree model	**0m 14s**
Our Maxent model	0m 24s
Our SVM model	3m 50s

Table 5: Comparison of parsing speed

Our initial attempt to increase the accuracy of the DTree model by applying boosting techniques did not yield satisfactory results. In our experiment, we implemented the AdaBoost.M1 (Freund and Schapire, 1996) algorithm using resampling to vary the training set distribution. Results showed AdaBoost suffered severe overfitting problems and hurts accuracy greatly, even with a small number of samples. One possible reason for this is that our sample space is very unbalanced across the different classes. A few classes have lots of training examples while a large number of classes are rare, which could raise the chance of overfitting.

In our experiments, SVM model gave better results than the Maxent model. But it is important to note that although the same set of features were used in both models, a degree 2 polynomial kernel was used in the SVM classifier while Maxent only has degree 1 features. In our future work, we will experiment with degree 2 features and L1 regularization in the Maxent model, which may give us closer performance to the SVM model with a much faster speed.

7 Conclusion

In this paper, we presented a novel deterministic parser for Chinese constituent parsing. Using gold-standard POS tags, our best model (using stacked classifiers) runs in linear time and has labeled recall and precision of 88.3% and 88.1%, respectively, surpassing the best published results. And with a trade-off of 5-6% in accuracy, our DTree and Maxent parsers run at speeds 40-270 times faster than state-of-the-art parsers. Our re-

[3] Bikel's parser used gold-standard POS tags for unseen words only. Also, the results are obtained from a parser trained on 250K-CTB, about 2.5 times bigger than CTB 1.0.

[4] All the experiments were conducted on a Pentium IV 2.4GHz machine with 2GB of RAM.

sults have shown that the deterministic parsing framework is a viable and effective approach to Chinese parsing. For future work, we will further improve the speed and accuracy of our models, and apply them to more Chinese and multilingual natural language applications that require high speed and accurate parsing.

Acknowledgment

This work was supported in part by ARDA's AQUAINT Program. We thank Eric Nyberg for his help during the final preparation of this paper.

References

Daniel M. Bikel and David Chiang. 2000. Two statistical parsing models applied to the Chinese Treebank. In *Proceedings of the Second Chinese Language Processing Workshop, ACL '00*.

Daniel M. Bikel. 2004. *On the Parameter Space of Generative Lexicalized Statistical Parsing Models.* Ph.D. thesis, University of Pennsylvania.

Yuchang Cheng, Masayuki Asahara, and Yuji Matsumoto. 2004. Deterministic dependency structure analyzer for Chinese. In *Proceedings of IJCNLP '04*.

Yuchang Cheng, Masayuki Asahara, and Yuji Matsumoto. 2005. Machine learning-based dependency analyzer for Chinese. In *Proceedings of ICCC '05*.

David Chiang and Daniel M. Bikel. 2002. Recovering latent information in treebanks. In *Proceedings of COLING '02*.

Michael John Collins. 1999. *Head-driven Statistical Models for Natural Language Parsing.* Ph.D. thesis, University of Pennsylvania.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2004. Timbl version 5.1 reference guide. Technical report, Tilburg University.

Yoav Freund and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In *Proceedings of ICML '96*.

Pascale Fung, Grace Ngai, Yongsheng Yang, and Benfeng Chen. 2004. A maximum-entropy Chinese parser augmented by transformation-based learning. *ACM Transactions on Asian Language Information Processing*, 3(2):159–168.

Mary Hearne and Andy Way. 2004. Data-oriented parsing and the Penn Chinese Treebank. In *Proceedings of IJCNLP '04*.

John Henderson and Eric Brill. 1999. Exploiting diversity in natural language processing: Combining parsers. In *Proceedings of EMNLP '99*.

Zhengping Jiang. 2004. Statistical Chinese parsing. Honours thesis, National University of Singapore.

Taku Kudo and Yuji Matsumoto. 2000. Use of support vector learning for chunk identification. In *Proceedings of CoNLL and LLL '00*.

Roger Levy and Christopher D. Manning. 2003. Is it harder to parse Chinese, or the Chinese Treebank? In *Proceedings of ACL '03*.

Xiaoqiang Luo. 2003. A maximum entropy Chinese character-based parser. In *Proceedings of EMNLP '03*.

David M. Magerman. 1994. *Natural Language Parsing as Statistical Pattern Recognition*. Ph.D. thesis, Stanford University.

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 '04*.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING '04*.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the IWPT '05*.

Honglin Sun and Daniel Jurafsky. 2003. The effect of rhythm on structural disambiguation in Chinese. In *Proceedings of SIGHAN Workshop '03*.

Honglin Sun and Daniel Jurafsky. 2004. Shallow semantic parsing of Chinese. In *Proceedings of the HLT/NAACL '04*.

Antal van den Bosch and Jakub Zavrel. 2000. Unpacking multi-valued symbolic features and classes in memory-based language learning. In *Proceedings of ICML '00*.

Deyi Xiong, Shuanglong Li, Qun Liu, Shouxun Lin, and Yueliang Qian. 2005. Parsing the Penn Chinese Treebank with semantic knowledge. In *Proceedings of IJCNLP '05*.

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(2):207–238.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT '03*.

Le Zhang, 2004. *Maximum Entropy Modeling Toolkit for Python and C++*. Reference Manual.

Learning Accurate, Compact, and Interpretable Tree Annotation

Slav Petrov **Leon Barrett** **Romain Thibaux** **Dan Klein**
Computer Science Division, EECS Department
University of California at Berkeley
Berkeley, CA 94720
{petrov, lbarrett, thibaux, klein}@eecs.berkeley.edu

Abstract

We present an automatic approach to tree annotation in which basic nonterminal symbols are alternately split and merged to maximize the likelihood of a training treebank. Starting with a simple X-bar grammar, we learn a new grammar whose nonterminals are subsymbols of the original nonterminals. In contrast with previous work, we are able to split various terminals to different degrees, as appropriate to the actual complexity in the data. Our grammars automatically learn the kinds of linguistic distinctions exhibited in previous work on manual tree annotation. On the other hand, our grammars are much more compact and substantially more accurate than previous work on automatic annotation. Despite its simplicity, our best grammar achieves an F_1 of 90.2% on the Penn Treebank, higher than fully lexicalized systems.

1 Introduction

Probabilistic context-free grammars (PCFGs) underlie most high-performance parsers in one way or another (Collins, 1999; Charniak, 2000; Charniak and Johnson, 2005). However, as demonstrated in Charniak (1996) and Klein and Manning (2003), a PCFG which simply takes the empirical rules and probabilities off of a treebank does not perform well. This naive grammar is a poor one because its context-freedom assumptions are too strong in some places (e.g. it assumes that subject and object NPs share the same distribution) and too weak in others (e.g. it assumes that long rewrites are not decomposable into smaller steps). Therefore, a variety of techniques have been developed to both enrich and generalize the naive grammar, ranging from simple tree annotation and symbol splitting (Johnson, 1998; Klein and Manning, 2003) to full lexicalization and intricate smoothing (Collins, 1999; Charniak, 2000).

In this paper, we investigate the learning of a grammar consistent with a treebank at the level of evaluation symbols (such as NP, VP, etc.) but split based on the likelihood of the training trees. Klein and Manning (2003) addressed this question from a linguistic perspective, starting with a Markov grammar and manually splitting symbols in response to observed linguistic trends in the data. For example, the symbol NP might be split into the subsymbol NPˆS in subject position and the subsymbol NPˆVP in object position. Recently, Matsuzaki et al. (2005) and also Prescher (2005) exhibited an automatic approach in which each symbol is split into a fixed number of subsymbols. For example, NP would be split into NP-1 through NP-8. Their exciting result was that, while grammars quickly grew too large to be managed, a 16-subsymbol induced grammar reached the parsing performance of Klein and Manning (2003)'s manual grammar. Other work has also investigated aspects of automatic grammar refinement; for example, Chiang and Bikel (2002) learn annotations such as head rules in a constrained declarative language for tree-adjoining grammars.

We present a method that combines the strengths of both manual and automatic approaches while addressing some of their common shortcomings. Like Matsuzaki et al. (2005) and Prescher (2005), we induce splits in a fully automatic fashion. However, we use a more sophisticated split-and-merge approach that allocates subsymbols adaptively where they are most effective, like a linguist would. The grammars recover patterns like those discussed in Klein and Manning (2003), heavily articulating complex and frequent categories like NP and VP while barely splitting rare or simple ones (see Section 3 for an empirical analysis).

Empirically, hierarchical splitting increases the accuracy and lowers the variance of the learned grammars. Another contribution is that, unlike previous work, we investigate smoothed models, allowing us to split grammars more heavily before running into the oversplitting effect discussed in Klein and Manning (2003), where data fragmentation outweighs increased expressivity.

Our method is capable of learning grammars of substantially smaller size and higher accuracy than previous grammar refinement work, starting from a simpler initial grammar. For example, even beginning with an X-bar grammar (see Section 1.1) with 98 symbols, our best grammar, using 1043 symbols, achieves a test set F_1 of 90.2%. This is a 27% reduction in error and a significant reduction in size[1] over the most accurate gram-

[1] This is a 97.5% reduction in number of symbols. Matsuzaki et al. (2005) do not report a number of rules, but our small number of symbols and our hierarchical training (which

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 433–440,
Sydney, July 2006.

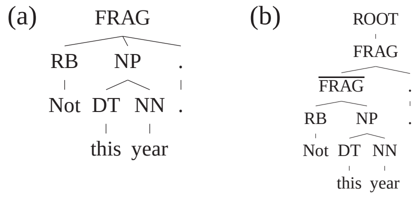


Figure 1: (a) The original tree. (b) The X-bar tree.

mar in Matsuzaki et al. (2005). Our grammar's accuracy was higher than fully lexicalized systems, including the maximum-entropy inspired parser of Charniak and Johnson (2005).

1.1 Experimental Setup

We ran our experiments on the Wall Street Journal (WSJ) portion of the Penn Treebank using the standard setup: we trained on sections 2 to 21, and we used section 1 as a validation set for tuning model hyperparameters. Section 22 was used as development set for intermediate results. All of section 23 was reserved for the final test. We used the EVALB parseval reference implementation, available from Sekine and Collins (1997), for scoring. All reported development set results are averages over four runs. For the final test we selected the grammar that performed best on the development set.

Our experiments are based on a completely unannotated X-bar style grammar, obtained directly from the Penn Treebank by the binarization procedure shown in Figure 1. For each local tree rooted at an evaluation nonterminal X, we introduce a cascade of new nodes labeled $\overline{X}$ so that each has two children. Rather than experiment with head-outward binarization as in Klein and Manning (2003), we simply used a left branching binarization; Matsuzaki et al. (2005) contains a comparison showing that the differences between binarizations are small.

2 Learning

To obtain a grammar from the training trees, we want to learn a set of rule probabilities β on latent annotations that maximize the likelihood of the training trees, despite the fact that the original trees lack the latent annotations. The Expectation-Maximization (EM) algorithm allows us to do exactly that.[2] Given a sentence w and its unannotated tree T, consider a nonterminal A spanning (r,t) and its children B and C spanning (r,s) and (s,t). Let A_x be a subsymbol of A, B_y of B, and C_z of C. Then the inside and outside probabilities $\mathrm{P}_{\text{IN}}(r,t,A_x) \overset{\text{def}}{=} P(w_{r:t}|A_x)$ and $\mathrm{P}_{\text{OUT}}(r,t,A_x) \overset{\text{def}}{=} P(w_{1:r}A_x w_{t:n})$ can be computed recursively:

$$\begin{aligned}
\mathrm{P}_{\text{IN}}(r,t,A_x) &= \sum_{y,z} \begin{array}{l}\beta(A_x \to B_y C_z)\\ \times \mathrm{P}_{\text{IN}}(r,s,B_y)\mathrm{P}_{\text{IN}}(s,t,C_z)\end{array}\\
\mathrm{P}_{\text{OUT}}(r,s,B_y) &= \sum_{x,z} \begin{array}{l}\beta(A_x \to B_y C_z)\\ \times \mathrm{P}_{\text{OUT}}(r,t,A_x)\mathrm{P}_{\text{IN}}(s,t,C_z)\end{array}\\
\mathrm{P}_{\text{OUT}}(s,t,C_z) &= \sum_{x,y} \begin{array}{l}\beta(A_x \to B_y C_z)\\ \times \mathrm{P}_{\text{OUT}}(r,t,A_x)\mathrm{P}_{\text{IN}}(r,s,B_y)\end{array}
\end{aligned}$$

Although we show only the binary component here, of course there are both binary and unary productions that are included. In the Expectation step, one computes the posterior probability of each annotated rule and position in each training set tree T:

$$\begin{aligned}
&P((r,s,t,A_x \to B_y C_z)|w,T) \propto \mathrm{P}_{\text{OUT}}(r,t,A_x)\\
&\quad \times \beta(A_x \to B_y C_z)\mathrm{P}_{\text{IN}}(r,s,B_y)\mathrm{P}_{\text{IN}}(s,t,C_z) \qquad (1)
\end{aligned}$$

In the Maximization step, one uses the above probabilities as weighted observations to update the rule probabilities:

$$\beta(A_x \to B_y C_z) := \frac{\#\{A_x \to B_y C_z\}}{\sum_{y',z'} \#\{A_x \to B_{y'} C_{z'}\}}$$

Note that, because there is no uncertainty about the location of the brackets, this formulation of the inside-outside algorithm is linear in the length of the sentence rather than cubic (Pereira and Schabes, 1992).

For our lexicon, we used a simple yet robust method for dealing with unknown and rare words by extracting a small number of features from the word and then computing appproximate tagging probabilities.[3]

2.1 Initialization

EM is only guaranteed to find a local maximum of the likelihood, and, indeed, in practice it often gets stuck in a suboptimal configuration. If the search space is very large, even restarting may not be sufficient to alleviate this problem. One workaround is to manually specify some of the annotations. For instance, Matsuzaki et al. (2005) start by annotating their grammar with the identity of the parent and sibling, which are observed (i.e. not latent), before adding latent annotations.[4] If these manual annotations are good, they reduce the search space for EM by constraining it to a smaller region. On the other hand, this pre-splitting defeats some of the purpose of automatically learning latent annotations,

encourages sparsity) suggest a large reduction.

[2] Other techniques are also possible; Henderson (2004) uses neural networks to induce latent left-corner parser states.

[3] A word is classified into one of 50 unknown word categories based on the presence of features such as capital letters, digits, and certain suffixes and its tagging probability is given by: $\mathrm{P}'(\text{word}|\text{tag}) = k\,\hat{\mathrm{P}}(\text{class}|\text{tag})$ where k is a constant representing $P(\text{word}|\text{class})$ and can simply be dropped. Rare words are modeled using a combination of their known and unknown distributions.

[4] In other words, in the terminology of Klein and Manning (2003), they begin with a (vertical order=2, horizontal order=1) baseline grammar.

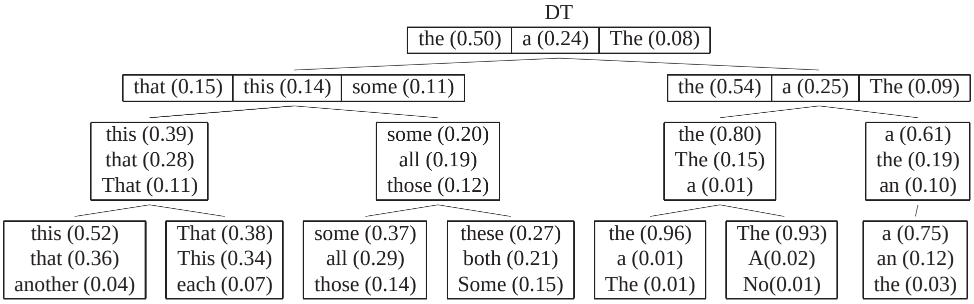

DT
the (0.50) a (0.24) The (0.08)
that (0.15) this (0.14) some (0.11)
the (0.54) a (0.25) The (0.09)
this (0.39) that (0.28) That (0.11)
some (0.20) all (0.19) those (0.12)
the (0.80) The (0.15) a (0.01)
a (0.61) the (0.19) an (0.10)
this (0.52) that (0.36) another (0.04)
That (0.38) This (0.34) each (0.07)
some (0.37) all (0.29) those (0.14)
these (0.27) both (0.21) Some (0.15)
the (0.96) a (0.01) The (0.01)
The (0.93) A(0.02) No(0.01)
a (0.75) an (0.12) the (0.03)

Figure 2: Evolution of the DT tag during hierarchical splitting and merging. Shown are the top three words for each subcategory and their respective probability.

leaving to the user the task of guessing what a good starting annotation might be.

We take a different, fully automated approach. We start with a completely unannotated X-bar style grammar as described in Section 1.1. Since we will evaluate our grammar on its ability to recover the Penn Treebank nonterminals, we must include them in our grammar. Therefore, this initialization is the absolute minimum starting grammar that includes the evaluation nonterminals (and maintains separate grammar symbols for each of them).[5] It is a very compact grammar: 98 symbols,[6] 236 unary rules, and 3840 binary rules. However, it also has a very low parsing performance: 65.8/59.8 LP/LR on the development set.

2.2 Splitting

Beginning with this baseline grammar, we repeatedly split and re-train the grammar. In each iteration we initialize EM with the results of the smaller grammar, splitting every previous annotation symbol in two and adding a small amount of randomness (1%) to break the symmetry. The results are shown in Figure 3. Hierarchical splitting leads to better parameter estimates over directly estimating a grammar with 2^k subsymbols per symbol. While the two procedures are identical for only two subsymbols (F_1: 76.1%), the hierarchical training performs better for four subsymbols (83.7% vs. 83.2%). This advantage grows as the number of subsymbols increases (88.4% vs. 87.3% for 16 subsymbols). This trend is to be expected, as the possible interactions between the subsymbols grows as their number grows. As an example of how staged training proceeds, Figure 2 shows the evolution of the subsymbols of the determiner (DT) tag, which first splits demonstratives from determiners, then splits quantificational elements from demonstratives along one branch and definites from indefinites along the other.

Because EM is a local search method, it is likely to converge to different local maxima for different runs. In our case, the variance is higher for models with few subcategories; because not all dependencies can be expressed with the limited number of subcategories, the results vary depending on which one EM selects first. As the grammar size increases, the important dependencies can be modeled, so the variance decreases.

2.3 Merging

It is clear from all previous work that creating more latent annotations can increase accuracy. On the other hand, oversplitting the grammar can be a serious problem, as detailed in Klein and Manning (2003). Adding subsymbols divides grammar statistics into many bins, resulting in a tighter fit to the training data. At the same time, each bin gives a less robust estimate of the grammar probabilities, leading to overfitting. Therefore, it would be to our advantage to split the latent annotations only where needed, rather than splitting them all as in Matsuzaki et al. (2005). In addition, if all symbols are split equally often, one quickly (4 split cycles) reaches the limits of what is computationally feasible in terms of training time and memory usage.

Consider the comma POS tag. We would like to see only one sort of this tag because, despite its frequency, it always produces the terminal comma (barring a few annotation errors in the treebank). On the other hand, we would expect to find an advantage in distinguishing between various verbal categories and NP types. Additionally, splitting symbols like the comma is not only unnecessary, but potentially harmful, since it needlessly fragments observations of other symbols' behavior.

It should be noted that simple frequency statistics are not sufficient for determining how often to split each symbol. Consider the closed part-of-speech classes (e.g. DT, CC, IN) or the nonterminal ADJP. These symbols are very common, and certainly do contain subcategories, but there is little to be gained from exhaustively splitting them before even beginning to model the rarer symbols that describe the complex inner correlations inside verb phrases. Our solution is to use a split-and-merge approach broadly reminiscent of ISODATA, a classic clustering procedure (Ball and

[5] If our purpose was only to model language, as measured for instance by perplexity on new text, it could make sense to erase even the labels of the Penn Treebank to let EM find better labels by itself, giving an experiment similar to that of Pereira and Schabes (1992).

[6] 45 part of speech tags, 27 phrasal categories and the 26 intermediate symbols which were added during binarization

Hall, 1967).

To prevent oversplitting, we could measure the utility of splitting each latent annotation individually and then split the best ones first. However, not only is this impractical, requiring an entire training phase for each new split, but it assumes the contributions of multiple splits are independent. In fact, extra subsymbols may need to be added to several nonterminals before they can cooperate to pass information along the parse tree. Therefore, we go in the opposite direction; that is, we split every symbol in two, train, and then measure for each annotation the loss in likelihood incurred when removing it. If this loss is small, the new annotation does not carry enough useful information and can be removed. What is more, contrary to the gain in likelihood for splitting, the loss in likelihood for merging can be efficiently approximated.[7]

Let T be a training tree generating a sentence w. Consider a node n of T spanning (r, t) with the label A; that is, the subtree rooted at n generates $w_{r:t}$ and has the label A. In the latent model, its label A is split up into several latent labels, A_x. The likelihood of the data can be recovered from the inside and outside probabilities at n:

$$\mathrm{P}(w,T) = \sum_x \mathrm{P}_{\text{IN}}(r,t,A_x)\mathrm{P}_{\text{OUT}}(r,t,A_x) \qquad (2)$$

Consider merging, at n only, two annotations A_1 and A_2. Since A now combines the statistics of A_1 and A_2, its production probabilities are the sum of those of A_1 and A_2, weighted by their relative frequency p_1 and p_2 in the training data. Therefore the inside score of A is:

$$\mathrm{P}_{\text{IN}}(r,t,A) = p_1\mathrm{P}_{\text{IN}}(r,t,A_1) + p_2\mathrm{P}_{\text{IN}}(r,t,A_2)$$

Since A can be produced as A_1 or A_2 by its parents, its outside score is:

$$\mathrm{P}_{\text{OUT}}(r,t,A) = \mathrm{P}_{\text{OUT}}(r,t,A_1) + \mathrm{P}_{\text{OUT}}(r,t,A_2)$$

Replacing these quantities in (2) gives us the likelihood $\mathrm{P}^n(w,T)$ where these two annotations and their corresponding rules have been merged, around only node n.

We approximate the overall loss in data likelihood due to merging A_1 and A_2 everywhere in all sentences w^i by the product of this loss for each local change:

$$\Delta_{\text{ANNOTATION}}(A_1,A_2) = \prod_i \prod_{n\in T_i} \frac{\mathrm{P}^n(w^i,T_i)}{\mathrm{P}(w^i,T_i)}$$

This expression is an approximation because it neglects interactions between instances of a symbol at multiple places in the same tree. These instances, however, are often far apart and are likely to interact only weakly, and this simplification avoids the prohibitive cost of running an inference algorithm for each tree and annotation. We refer to the operation of splitting annotations and re-merging some them based on likelihood loss as a split-merge (SM) cycle. SM cycles allow us to progressively increase the complexity of our grammar, giving priority to the most useful extensions.

In our experiments, merging was quite valuable. Depending on how many splits were reversed, we could reduce the grammar size at the cost of little or no loss of performance, or even a gain. We found that merging 50% of the newly split symbols dramatically reduced the grammar size after each splitting round, so that after 6 SM cycles, the grammar was only 17% of the size it would otherwise have been (1043 vs. 6273 subcategories), while at the same time there was no loss in accuracy (Figure 3). Actually, the accuracy even increases, by 1.1% at 5 SM cycles. The numbers of splits learned turned out to not be a direct function of symbol frequency; the numbers of symbols for both lexical and nonlexical tags after 4 SM cycles are given in Table 2. Furthermore, merging makes large amounts of splitting possible. It allows us to go from 4 splits, equivalent to the $2^4 = 16$ substates of Matsuzaki et al. (2005), to 6 SM iterations, which take a few days to run on the Penn Treebank.

2.4 Smoothing

Splitting nonterminals leads to a better fit to the data by allowing each annotation to specialize in representing only a fraction of the data. The smaller this fraction, the higher the risk of overfitting. Merging, by allowing only the most beneficial annotations, helps mitigate this risk, but it is not the only way. We can further minimize overfitting by forcing the production probabilities from annotations of the same nonterminal to be similar. For example, a noun phrase in subject position certainly has a distinct distribution, but it may benefit from being smoothed with counts from all other noun phrases. Smoothing the productions of each subsymbol by shrinking them towards their common base symbol gives us a more reliable estimate, allowing them to share statistical strength.

We perform smoothing in a linear way. The estimated probability of a production $p_x = \mathrm{P}(A_x \rightarrow B_y\, C_z)$ is interpolated with the average over all subsymbols of A.

$$p'_x = (1-\alpha)p_x + \alpha\bar{p} \quad \text{where} \quad \bar{p} = \frac{1}{n}\sum_x p_x$$

Here, α is a small constant: we found 0.01 to be a good value, but the actual quantity was surprisingly unimportant. Because smoothing is most necessary when production statistics are least reliable, we expect smoothing to help more with larger numbers of subsymbols. This is exactly what we observe in Figure 3, where smoothing initially hurts (subsymbols are quite distinct

[7] The idea of merging complex hypotheses to encourage generalization is also examined in Stolcke and Omohundro (1994), who used a chunking approach to propose new productions in fully unsupervised grammar induction. They also found it necessary to make local choices to guide their likelihood search.

and do not need their estimates pooled) but eventually helps (as symbols have finer distinctions in behavior and smaller data support).

2.5 Parsing

When parsing new sentences with an annotated grammar, returning the most likely (unannotated) tree is intractable: to obtain the probability of an unannotated tree, one must sum over combinatorially many annotation trees (derivations) for each tree (Sima'an, 1992).

Matsuzaki et al. (2005) discuss two approximations. The first is settling for the most probable derivation rather than most probable parse, i.e. returning the single most likely (Viterbi) annotated tree (derivation). This approximation is justified if the sum is dominated by one particular annotated tree. The second approximation that Matsuzaki et al. (2005) present is the Viterbi parse under a new sentence-specific PCFG, whose rule probabilities are given as the solution of a variational approximation of the original grammar. However, their rule probabilities turn out to be the posterior probability, given the sentence, of each rule being used at each position in the tree. Their algorithm is therefore the *labelled recall* algorithm of Goodman (1996) but applied to rules. That is, it returns the tree whose expected number of correct rules is maximal. Thus, assuming one is interested in a per-position score like F_1 (which is its own debate), this method of parsing is actually more appropriate than finding the most likely parse, not simply a cheap approximation of it, and it need not be derived by a variational argument. We refer to this method of parsing as the *max-rule* parser. Since this method is not a contribution of this paper, we refer the reader to the fuller presentations in Goodman (1996) and Matsuzaki et al. (2005). Note that contrary to the original labelled recall algorithm, which maximizes the number of correct symbols, this tree only contains rules allowed by the grammar. As a result, the percentage of complete matches with the max-rule parser is typically higher than with the Viterbi parser. (37.5% vs. 35.8% for our best grammar).

These posterior rule probabilities are still given by (1), but, since the structure of the tree is no longer known, we must sum over it when computing the inside and outside probabilities:

$$\text{P}_{\text{IN}}(r,t,A_x) = \sum_{B,C,s}\sum_{y,z} \begin{array}{l}\beta(A_x \to B_y C_z)\times \\ \text{P}_{\text{IN}}(r,s,B_y)\text{P}_{\text{IN}}(s,t,C_z)\end{array}$$

$$\text{P}_{\text{OUT}}(r,s,B_y) = \sum_{A,C,t}\sum_{x,z} \begin{array}{l}\beta(A_x \to B_y C_z)\times \\ \text{P}_{\text{OUT}}(r,t,A_x)\text{P}_{\text{IN}}(s,t,C_z)\end{array}$$

$$\text{P}_{\text{OUT}}(s,t,C_z) = \sum_{A,B,r}\sum_{x,y} \begin{array}{l}\beta(A_x \to B_y C_z)\times \\ \text{P}_{\text{OUT}}(r,t,A_x)\text{P}_{\text{IN}}(r,s,B_y)\end{array}$$

For efficiency reasons, we use a coarse-to-fine pruning scheme like that of Caraballo and Charniak (1998). For a given sentence, we first run the inside-outside algorithm using the baseline (unannotated) grammar, producing a packed forest representation of the posterior symbol probabilities for each span. For example, one span might have a posterior probability of 0.8 of the symbol NP, but e^{-10} for PP. Then, we parse with the larger annotated grammar, but, at each span, we prune away any symbols whose posterior probability under the baseline grammar falls below a certain threshold (e^{-8} in our experiments). Even though our baseline grammar has a very low accuracy, we found that this pruning barely impacts the performance of our better grammars, while significantly reducing the computational cost. For a grammar with 479 subcategories (4 SM cycles), lowering the threshold to e^{-15} led to an F_1 improvement of 0.13% (89.03 vs. 89.16) on the development set but increased the parsing time by a factor of 16.

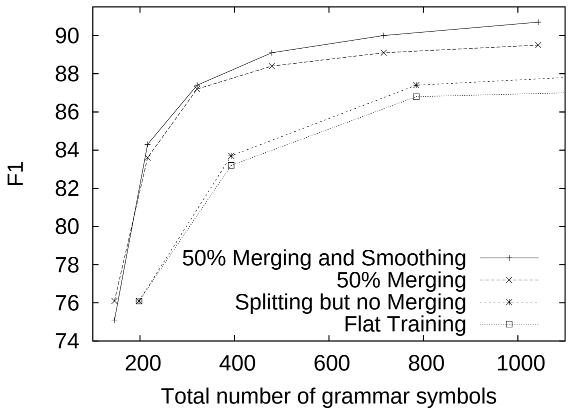


Figure 3: Hierarchical training leads to better parameter estimates. Merging reduces the grammar size significantly, while preserving the accuracy and enabling us to do more SM cycles. Parameter smoothing leads to even better accuracy for grammars with high complexity.

3 Analysis

So far, we have presented a split-merge method for learning to iteratively subcategorize basic symbols like NP and VP into automatically induced subsymbols (subcategories in the original sense of Chomsky (1965)). This approach gives parsing accuracies of up to 90.7% on the development set, substantially higher than previous symbol-splitting approaches, while starting from an extremely simple base grammar. However, in general, any automatic induction system is in danger of being entirely uninterpretable. In this section, we examine the learned grammars, discussing what is learned. We focus particularly on connections with the linguistically motivated annotations of Klein and Manning (2003), which we do generally recover.

Inspecting a large grammar by hand is difficult, but fortunately, our baseline grammar has less than 100 nonterminal symbols, and even our most complicated grammar has only 1043 total (sub)symbols. It is there-

VBZ			
VBZ-0	gives	sells	takes
VBZ-1	comes	goes	works
VBZ-2	includes	owns	is
VBZ-3	puts	provides	takes
VBZ-4	says	adds	Says
VBZ-5	believes	means	thinks
VBZ-6	expects	makes	calls
VBZ-7	plans	expects	wants
VBZ-8	is	’s	gets
VBZ-9	’s	is	remains
VBZ-10	has	’s	is
VBZ-11	does	Is	Does

NNP			
NNP-0	Jr.	Goldman	INC.
NNP-1	Bush	Noriega	Peters
NNP-2	J.	E.	L.
NNP-3	York	Francisco	Street
NNP-4	Inc	Exchange	Co
NNP-5	Inc.	Corp.	Co.
NNP-6	Stock	Exchange	York
NNP-7	Corp.	Inc.	Group
NNP-8	Congress	Japan	IBM
NNP-9	Friday	September	August
NNP-10	Shearson	D.	Ford
NNP-11	U.S.	Treasury	Senate
NNP-12	John	Robert	James
NNP-13	Mr.	Ms.	President
NNP-14	Oct.	Nov.	Sept.
NNP-15	New	San	Wall

JJS			
JJS-0	largest	latest	biggest
JJS-1	least	best	worst
JJS-2	most	Most	least

DT			
DT-0	the	The	a
DT-1	A	An	Another
DT-2	The	No	This
DT-3	The	Some	These
DT-4	all	those	some
DT-5	some	these	both
DT-6	That	This	each
DT-7	this	that	each
DT-8	the	The	a
DT-9	no	any	some
DT-10	an	a	the
DT-11	a	this	the

CD			
CD-0	1	50	100
CD-1	8.50	15	1.2
CD-2	8	10	20
CD-3	1	30	31
CD-4	1989	1990	1988
CD-5	1988	1987	1990
CD-6	two	three	five
CD-7	one	One	Three
CD-8	12	34	14
CD-9	78	58	34
CD-10	one	two	three
CD-11	million	billion	trillion

PRP			
PRP-0	It	He	I
PRP-1	it	he	they
PRP-2	it	them	him

RBR			
RBR-0	further	lower	higher
RBR-1	more	less	More
RBR-2	earlier	Earlier	later

IN			
IN-0	In	With	After
IN-1	In	For	At
IN-2	in	for	on
IN-3	of	for	on
IN-4	from	on	with
IN-5	at	for	by
IN-6	by	in	with
IN-7	for	with	on
IN-8	If	While	As
IN-9	because	if	while
IN-10	whether	if	That
IN-11	that	like	whether
IN-12	about	over	between
IN-13	as	de	Up
IN-14	than	ago	until
IN-15	out	up	down

RB			
RB-0	recently	previously	still
RB-1	here	back	now
RB-2	very	highly	relatively
RB-3	so	too	as
RB-4	also	now	still
RB-5	however	Now	However
RB-6	much	far	enough
RB-7	even	well	then
RB-8	as	about	nearly
RB-9	only	just	almost
RB-10	ago	earlier	later
RB-11	rather	instead	because
RB-12	back	close	ahead
RB-13	up	down	off
RB-14	not	Not	maybe
RB-15	n’t	not	also

Table 1: The most frequent three words in the subcategories of several part-of-speech tags.

fore relatively straightforward to review the broad behavior of a grammar. In this section, we review a randomly-selected grammar after 4 SM cycles that produced an F_1 score on the development set of 89.11. We feel it is reasonable to present only a single grammar because all the grammars are very similar. For example, after 4 SM cycles, the F_1 scores of the 4 trained grammars have a variance of only 0.024, which is tiny compared to the deviation of 0.43 obtained by Matsuzaki et al. (2005)). Furthermore, these grammars allocate splits to nonterminals with a variance of only 0.32, so they agree to within a single latent state.

3.1 Lexical Splits

One of the original motivations for lexicalization of parsers is the fact that part-of-speech (POS) tags are usually far too general to encapsulate a word’s syntactic behavior. In the limit, each word may well have its own unique syntactic behavior, especially when, as in modern parsers, semantic selectional preferences are lumped in with traditional syntactic trends. However, in practice, and given limited data, the relationship between specific words and their syntactic contexts may be best modeled at a level more fine than POS tag but less fine than lexical identity.

In our model, POS tags are split just like any other grammar symbol: the subsymbols for several tags are shown in Table 1, along with their most frequent members. In most cases, the categories are recognizable as either classic subcategories or an interpretable division of some other kind.

Nominal categories are the most heavily split (see Table 2), and have the splits which are most semantic in nature (though not without syntactic correlations). For example, plural common nouns (NNS) divide into the maximum number of categories (16). One category consists primarily of dates, whose typical parent is an NP subsymbol whose typical parent is a root S, essentially modeling the temporal noun annotation discussed in Klein and Manning (2003). Another category specializes in capitalized words, preferring as a parent an NP with an S parent (i.e. subject position). A third category specializes in monetary units, and so on. These kinds of syntactico-semantic categories are typical, and, given distributional clustering results like those of Schuetze (1998), unsurprising. The singular nouns are broadly similar, if slightly more homogenous, being dominated by categories for stocks and trading. The proper noun category (NNP, shown) also splits into the maximum 16 categories, including months, countries, variants of *Co.* and *Inc.*, first names, last names, initials, and so on.

Verbal categories are also heavily split. Verbal subcategories sometimes reflect syntactic selectional preferences, sometimes reflect semantic selectional preferences, and sometimes reflect other aspects of verbal syntax. For example, the present tense third person verb subsymbols (VBZ) are shown. The auxiliaries get three clear categories: *do*, *have*, and *be* (this pattern repeats in other tenses), as well a fourth category for the ambiguous *’s*. Verbs of communication (*says*) and

NNP	62	CC	7	WP$	2	NP	37	CONJP	2
JJ	58	JJR	5	WDT	2	VP	32	FRAG	2
NNS	57	JJS	5	-RRB-	2	PP	28	NAC	2
NN	56	:	5	”	1	ADVP	22	UCP	2
VBN	49	PRP	4	FW	1	S	21	WHADVP	2
RB	47	PRP$	4	RBS	1	ADJP	19	INTJ	1
VBG	40	MD	3	TO	1	SBAR	15	SBARQ	1
VB	37	RBR	3	$	1	QP	9	RRC	1
VBD	36	WP	2	UH	1	WHNP	5	WHADJP	1
CD	32	POS	2	,	1	PRN	4	X	1
IN	27	PDT	2	“	1	NX	4	ROOT	1
VBZ	25	WRB	2	SYM	1	SINV	3	LST	1
VBP	19	-LRB-	2	RP	1	PRT	2		
DT	17	.	2	LS	1	WHPP	2		
NNPS	11	EX	2	#	1	SQ	2		

Table 2: Number of latent annotations determined by our split-merge procedure after 6 SM cycles

propositional attitudes (*beleives*) that tend to take inflected sentential complements dominate two classes, while control verbs (*wants*) fill out another.

As an example of a less-split category, the superlative adjectives (JJS) are split into three categories, corresponding principally to *most*, *least*, and *largest*, with most frequent parents NP, QP, and ADVP, respectively. The relative adjectives (JJR) are split in the same way. Relative adverbs (RBR) are split into a different three categories, corresponding to (usually metaphorical) distance (*further*), degree (*more*), and time (*earlier*). Personal pronouns (PRP) are well-divided into three categories, roughly: nominative case, accusative case, and sentence-initial nominative case, which each correlate very strongly with syntactic position. As another example of a specific trend which was mentioned by Klein and Manning (2003), adverbs (RB) do contain splits for adverbs under ADVPs (*also*), NPs (*only*), and VPs (*not*).

Functional categories generally show fewer splits, but those splits that they do exhibit are known to be strongly correlated with syntactic behavior. For example, determiners (DT) divide along several axes: definite (*the*), indefinite (*a*), demonstrative (*this*), quantificational (*some*), negative polarity (*no*, *any*), and various upper- and lower-case distinctions inside these types. Here, it is interesting to note that these distinctions emerge in a predictable order (see Figure 2 for DT splits), beginning with the distinction between demonstratives and non-demonstratives, with the other distinctions emerging subsequently; this echoes the result of Klein and Manning (2003), where the authors chose to distinguish the demonstrative constrast, but not the additional ones learned here.

Another very important distinction, as shown in Klein and Manning (2003), is the various subdivisions in the preposition class (IN). Learned first is the split between subordinating conjunctions like *that* and proper prepositions. Then, subdivisions of each emerge: *wh*-subordinators like *if*, noun-modifying prepositions like *of*, predominantly verb-modifying ones like *from*, and so on.

Many other interesting patterns emerge, including many classical distinctions not specifically mentioned or modeled in previous work. For example, the *wh*-determiners (WDT) split into one class for *that* and another for *which*, while the *wh*-adverbs align by reference type: event-based *how* and *why* vs. entity-based *when* and *where*. The possesive particle (POS) has one class for the standard *’s*, but another for the plural-only apostrophe. As a final example, the cardinal number nonterminal (CD) induces various categories for dates, fractions, spelled-out numbers, large (usually financial) digit sequences, and others.

ADVP

ADVP-0	RB-13 NP-2	RB-13 PP-3	IN-15 NP-2
ADVP-1	NP-3 RB-10	NP-3 RBR-2	NP-3 IN-14
ADVP-2	IN-5 JJS-1	RB-8 RB-6	RB-6 RBR-1
ADVP-3	RBR-0	RB-12 PP-0	RP-0
ADVP-4	RB-3 RB-6	ADVP-2 SBAR-8	ADVP-2 PP-5
ADVP-5	RB-5	NP-3 RB-10	RB-0
ADVP-6	RB-4	RB-0	RB-3 RB-6
ADVP-7	RB-7	IN-5 JJS-1	RB-6
ADVP-8	RB-0	RBS-0	RBR-1 IN-14
ADVP-9	RB-1	IN-15	RBR-0

SINV

SINV-0	VP-14 NP-7	VP-14	VP-15 NP-7 NP-9
SINV-1	VP-14 NP-7 .-0 S-6 ,-0 VP-14 NP-7 .-0 S-11 VP-14 NP-7 .-0		

Table 3: The most frequent three productions of some latent annotations.

3.2 Phrasal Splits

Analyzing the splits of phrasal nonterminals is more difficult than for lexical categories, and we can merely give illustrations. We show some of the top productions of two categories in Table 3.

A nonterminal split can be used to model an otherwise uncaptured correlation between that symbol’s external context (e.g. its parent symbol) and its internal context (e.g. its child symbols). A particularly clean example of a split correlating external with internal contexts is the inverted sentence category (SINV), which has only two subsymbols, one which usually has the ROOT symbol as its parent (and which has sentence final puncutation as its last child), and a second subsymbol which occurs in embedded contexts (and does not end in punctuation). Such patterns are common, but often less easy to predict. For example, possesive NPs get two subsymbols, depending on whether their possessor is a person / country or an organization. The external correlation turns out to be that people and countries are more likely to possess a subject NP, while organizations are more likely to possess an object NP.

Nonterminal splits can also be used to relay information between distant tree nodes, though untangling this kind of propagation and distilling it into clean examples is not trivial. As one example, the subsymbol S-12 (matrix clauses) occurs only under the ROOT symbol. S-12’s children usually include NP-8, which in turn usually includes PRP-0, the capitalized nominative pronouns, DT-{1,2,6} (the capitalized determiners), and so on. This same propagation occurs even more frequently in the intermediate symbols, with, for example, one subsymbol of $\overline{\text{NP}}$ symbol specializing in propagating proper noun sequences.

Verb phrases, unsurprisingly, also receive a full set of subsymbols, including categories for infinitive VPs, passive VPs, several for intransitive VPs, several for transitive VPs with NP and PP objects, and one for sentential complements. As an example of how lexical splits can interact with phrasal splits, the two most frequent rewrites involving intransitive past tense verbs (VBD) involve two different VPs and VBDs: VP-14 → VBD-13 and VP-15 → VBD-12. The difference is that VP-14s are main clause VPs, while VP-15s are subordinate clause VPs. Correspondingly, VBD-13s are verbs of communication (*said, reported*), while VBD-12s are an assortment of verbs which often appear in subordinate contexts (*did, began*).

Other interesting phenomena also emerge. For example, intermediate symbols, which in previous work were very heavily, manually split using a Markov process, end up encoding processes which are largely Markov, but more complex. For example, some classes of adverb phrases (those with RB-4 as their head) are 'forgotten' by the $\overline{\text{VP}}$ intermediate grammar. The relevant rule is the very probable $\overline{\text{VP}}$-2 → $\overline{\text{VP}}$-2 ADVP-6; adding this ADVP to a growing VP does not change the VP subsymbol. In essense, at least a partial distinction between verbal arguments and verbal adjucts has been learned (as exploited in Collins (1999), for example).

4 Conclusions

By using a split-and-merge strategy and beginning with the barest possible initial structure, our method reliably learns a PCFG that is remarkably good at parsing. Hierarchical split/merge training enables us to learn compact but accurate grammars, ranging from extremely compact (an F_1 of 78% with only 147 symbols) to extremely accurate (an F_1 of 90.2% for our largest grammar with only 1043 symbols). Splitting provides a tight fit to the training data, while merging improves generalization and controls grammar size. In order to overcome data fragmentation and overfitting, we smooth our parameters. Smoothing allows us to add a larger number of annotations, each specializing in only a fraction of the data, without overfitting our training set. As one can see in Table 4, the resulting parser ranks among the best lexicalized parsers, beating those of Collins (1999) and Charniak and Johnson (2005).[8] Its F_1 performance is a 27% reduction in error over Matsuzaki et al. (2005) and Klein and Manning (2003). Not only is our parser more accurate, but the learned grammar is also significantly smaller than that of previous work. While this all is accomplished with only automatic learning, the resulting grammar is human-interpretable. It shows most of the manually introduced annotations discussed by Klein and Manning (2003), but also learns other linguistic phenomena.

≤ 40 words	LP	LR	CB	0CB
Klein and Manning (2003)	86.9	85.7	1.10	60.3
Matsuzaki et al. (2005)	86.6	86.7	1.19	61.1
Collins (1999)	88.7	88.5	0.92	66.7
Charniak and Johnson (2005)	90.1	**90.1**	**0.74**	**70.1**
This Paper	**90.3**	90.0	0.78	68.5
all sentences	LP	LR	CB	0CB
Klein and Manning (2003)	86.3	85.1	1.31	57.2
Matsuzaki et al. (2005)	86.1	86.0	1.39	58.3
Collins (1999)	88.3	88.1	1.06	64.0
Charniak and Johnson (2005)	89.5	**89.6**	**0.88**	**67.6**
This Paper	**89.8**	**89.6**	0.92	66.3

Table 4: Comparison of our results with those of others.

[8] Even with the Viterbi parser our best grammar achieves 88.7/88.9 LP/LR.

References

G. Ball and D. Hall. 1967. A clustering technique for summarizing multivariate data. *Behavioral Science*.

S. Caraballo and E. Charniak. 1998. New figures of merit for best–first probabilistic chart parsing. In *Computational Lingusitics*, p. 275–298.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *ACL'05*, p. 173–180.

E. Charniak. 1996. Tree-bank grammars. In *AAAI '96*, p. 1031–1036.

E. Charniak. 2000. A maximum–entropy–inspired parser. In *NAACL '00*, p. 132–139.

D. Chiang and D. Bikel. 2002. Recovering latent information in treebanks. In *Computational Linguistics*.

N. Chomsky. 1965. *Aspects of the Theory of Syntax*. MIT Press.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, U. of Pennsylvania.

J. Goodman. 1996. Parsing algorithms and metrics. In *ACL '96*, p. 177–183.

J. Henderson. 2004. Discriminative training of a neural network statistical parser. In *ACL '04*.

M. Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24:613–632.

D. Klein and C. Manning. 2003. Accurate unlexicalized parsing. *ACL '03*, p. 423–430.

T. Matsuzaki, Y. Miyao, and J. Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL '05*, p. 75–82.

F. Pereira and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *ACL '92*, p. 128–135.

D. Prescher. 2005. Inducing head-driven PCFGs with latent heads: Refining a tree-bank grammar for parsing. In *ECML'05*.

H. Schuetze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–124.

S. Sekine and M. J. Collins. 1997. EVALB bracket scoring program. `http://nlp.cs.nyu.edu/evalb/`.

K. Sima'an. 1992. Computatoinal complexity of probabilistic disambiguation. *Grammars*, 5:125–151.

A. Stolcke and S. Omohundro. 1994. Inducing probabilistic grammars by bayesian model merging. In *Grammatical Inference and Applications*, p. 106–118.

Semi-Supervised Learning of Partial Cognates using Bilingual Bootstrapping

Oana Frunza and **Diana Inkpen**

School of Information Technology and Engineering
University of Ottawa
Ottawa, ON, Canada, K1N 6N5
{ofrunza,diana}@site.uottawa.ca

Abstract

Partial cognates are pairs of words in two languages that have the same meaning in some, but not all contexts. Detecting the actual meaning of a partial cognate in context can be useful for Machine Translation tools and for Computer-Assisted Language Learning tools. In this paper we propose a supervised and a semi-supervised method to disambiguate partial cognates between two languages: French and English. The methods use only automatically-labeled data; therefore they can be applied for other pairs of languages as well. We also show that our methods perform well when using corpora from different domains.

1 Introduction

When learning a second language, a student can benefit from knowledge in his / her first language (Gass, 1987), (Ringbom, 1987), (LeBlanc *et al.* 1989). Cognates – words that have similar spelling and meaning – can accelerate vocabulary acquisition and facilitate the reading comprehension task. On the other hand, a student has to pay attention to the pairs of words that look and sound similar but have different meanings – false friends pairs, and especially to pairs of words that share meaning in some but not all contexts – the partial cognates.

Carroll (1992) claims that false friends can be a hindrance in second language learning. She suggests that a cognate pairing process between two words that look alike happens faster in the learner's mind than a false-friend pairing. Experiments with second language learners of different stages conducted by Van et al. (1998) suggest that missing false-friend recognition can be corrected when cross-language activation is used – sounds, pictures, additional explanation, feedback.

Machine Translation (MT) systems can benefit from extra information when translating a certain word in context. Knowing if a word in the source language is a cognate or a false friend with a word in the target language can improve the translation results. Cross-Language Information Retrieval systems can use the knowledge of the sense of certain words in a query in order to retrieve desired documents in the target language.

Our task, disambiguating partial cognates, is in a way equivalent to coarse grain cross-language Word-Sense Discrimination. Our focus is disambiguating French partial cognates in context: deciding if they are used as cognates with an English word, or if they are used as false friends.

There is a lot of work done on monolingual Word Sense Disambiguation (WSD) systems that use supervised and unsupervised methods and report good results on Senseval data, but there is less work done to disambiguate cross-language words. The results of this process can be useful in many NLP tasks.

Although French and English belong to different branches of the Indo-European family of languages, their vocabulary share a great number of similarities. Some are words of Latin and Greek origin: e.g., *education* and *theory*. A small number of very old, “genetic" cognates go back all the way to Proto-Indo-European, e.g., *mére - mother* and *pied - foot*. The majority of these pairs of words penetrated the French and English language due to the geographical, historical, and cultural contact between the two countries over

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 441–448,
Sydney, July 2006. ©2006 Association for Computational Linguistics

many centuries (borrowings). Most of the borrowings have changed their orthography, following different orthographic rules (LeBlanc and Seguin, 1996) and most likely their meaning as well. Some of the adopted words replaced the original word in the language, while others were used together but with slightly or completely different meanings.

In this paper we describe a supervised and also a semi-supervised method to discriminate the senses of partial cognates between French and English. In the following sections we present some definitions, the way we collected the data, the methods that we used, and evaluation experiments with results for both methods.

2 Definitions

We adopt the following definitions. The definitions are language-independent, but the examples are pairs of French and English words, respectively.

Cognates, or True Friends (Vrais Amis), are pairs of words that are perceived as similar and are mutual translations. The spelling can be identical or not, e.g., *nature* - *nature*, *reconnaissance* - *recognition*.

False Friends (Faux Amis) are pairs of words in two languages that are perceived as similar but have different meanings, e.g., *main* (= *hand*) - *main* (= *principal* or *essential*), *blesser* (= to *injure*) - *bless* (= bénir).

Partial Cognates are pairs of words that have the same meaning in both languages in some but not all contexts. They behave as cognates or as false friends, depending on the sense that is used in each context. For example, in French, *facteur* means not only *factor*, but also *mailman*, while *étiquette* can also mean *label* or *sticker*, in addition to the cognate sense.

Genetic Cognates are word pairs in related languages that derive directly from the same word in the ancestor (proto-)language. Because of gradual phonetic and semantic changes over long periods of time, genetic cognates often differ in form and/or meaning, e.g., *père* - *father*, *chef* - *head*. This category excludes lexical borrowings, i.e., words transferred from one language to another at some point of time, such as *concierge*.

3 Related Work

As far as we know there is no work done to disambiguate partial cognates between two languages.

Ide (2000) has shown on a small scale that cross-lingual lexicalization can be used to define and structure sense distinctions. Tufis et al. (2004) used cross-lingual lexicalization, wordnets alignment for several languages, and a clustering algorithm to perform WSD on a set of polysemous English words. They report an accuracy of 74%.

One of the most active researchers in identifying cognates between pairs of languages is Kondrak (2001; 2004). His work is more related to the phonetic aspect of cognate identification. He used in his work algorithms that combine different orthographic and phonetic measures, recurrent sound correspondences, and some semantic similarity based on glosses overlap. Guy (1994) identified letter correspondence between words and estimates the likelihood of relatedness. No semantic component is present in the system, the words are assumed to be already matched by their meanings. Hewson (1993), Lowe and Mazadon (1994) used systematic sound correspondences to determine proto-projections for identifying cognate sets.

WSD is a task that has attracted researchers since 1950 and it is still a topic of high interest. Determining the sense of an ambiguous word, using bootstrapping and texts from a different language was done by Yarowsky (1995), Hearst (1991), Diab (2002), and Li and Li (2004).

Yarowsky (1995) has used a few seeds and untagged sentences in a bootstrapping algorithm based on decision lists. He added two constrains – words tend to have one sense per discourse and one sense per collocation. He reported high accuracy scores for a set of 10 words. The monolingual bootstrapping approach was also used by Hearst (1991), who used a small set of hand-labeled data to bootstrap from a larger corpus for training a noun disambiguation system for English. Unlike Yarowsky (1995), we use automatic collection of seeds. Besides our monolingual bootstrapping technique, we also use bilingual bootstrapping.

Diab (2002) has shown that unsupervised WSD systems that use parallel corpora can achieve results that are close to the results of a supervised approach. She used parallel corpora in French, English, and Spanish, automatically-produced with MT tools to determine cross-language lexicalization sets of target words. The major goal of her work was to perform monolingual English WSD. Evaluation was performed on the nouns from the English all words data in Senseval2. Additional knowledge was added to the system from WordNet in order to improve the results. In our experiments we use the parallel data in a different way: we use words from parallel sentences as features for Machine Learning (ML). Li and Li (2004) have shown that word translation and bilingual bootstrapping is a good combination for disambiguation. They were using a set of 7 pairs of Chinese and English words. The two senses of the words were highly distinctive: *e.g. bass* as *fish* or *music; palm* as *tree* or *hand.*

Our work described in this paper shows that monolingual and bilingual bootstrapping can be successfully used to disambiguate partial cognates between two languages. Our approach differs from the ones we mentioned before not only from the point of human effort needed to annotate data – we require almost none, and from the way we use the parallel data to automatically collect training examples for machine learning, but also by the fact that we use only off-the-shelf tools and resources: free MT and ML tools, and parallel corpora. We show that a combination of these resources can be used with success in a task that would otherwise require a lot of time and human effort.

4 Data for Partial Cognates

We performed experiments with ten pairs of partial cognates. We list them in Table 1. For a French partial cognate we list its English cognate and several false friends in English. Often the French partial cognate has two senses (one for cognate, one for false friend), but sometimes it has more than two senses: one for cognate and several for false friends (nonetheless, we treat them together). For example, the false friend words for *note* have one sense for *grades* and one for *bills*.

The partial cognate (PC), the cognate (COG) and false-friend (FF) words were collected from a web resource[1]. The resource contained a list of 400 false-friends with 64 partial cognates. All partial cognates are words frequently used in the language. We selected ten partial cognates presented in Table 1 according to the number of extracted sentences (a balance between the two meanings), to evaluate and experiment our proposed methods.

The human effort that we required for our methods was to add more false-friend English words, than the ones we found in the web resource. We wanted to be able to distinguish the senses of cognate and false-friends for a wider variety of senses. This task was done using a bilingual dictionary[2].

Table 1. The ten pairs of partial cognates.

French partial cognate	English cognate	English false friends
blanc	blank	white, livid
circulation	circulation	traffic
client	client	customer, patron, patient, spectator, user, shopper
corps	corps	body, corpse
détail	detail	retail
mode	mode	fashion, trend, style, vogue
note	note	mark, grade, bill, check, account
police	police	policy, insurance, font, face
responsable	responsible	in charge, responsible party, official, representative, person in charge, executive, officer
route	route	road, roadside

4.1 Seed Set Collection

Both the supervised and the semi-supervised method that we will describe in Section 5 are using a set of seeds. The seeds are parallel sentences, French and English, which contain the partial cognate. For each partial-cognate word, a part of the set contains the cognate sense and another part the false-friend sense.

As we mentioned in Section 3, the seed sentences that we use are not hand-tagged with the sense (the cognate sense or the false-friend sense); they are automatically annotated by the way we collect them. To collect the set of seed sentences we use parallel corpora from Hansard[3], and EuroParl[4], and the, manually aligned BAF corpus.[5]

The cognate sense sentences were created by extracting parallel sentences that had on the French side the French cognate and on the English side the English cognate. See the upper part of Table 2 for an example.

The same approach was used to extract sentences with the false-friend sense of the partial cognate, only this time we used the false-friend English words. See lower the part of Table 2.

[1] http://french.about.com/library/fauxamis/blfauxam_a.htm

[2] http://www.wordreference.com

[3] http://www.isi.edu/natural-language/download/hansard/ and http://www.tsrali.com/

[4] http://people.csail.mit.edu/koehn/publications/europarl/

[5] http://rali.iro.umontreal.ca/Ressources/BAF/

Table 2. Example sentences from parallel corpus.

Fr (PC:COG)	Je *note*, par exemple, que l'accusé a fait une autre déclaration très incriminante à Hall environ deux mois plus tard.
En (COG)	I *note*, for instance, that he made another highly incriminating statement to Hall two months later.
Fr (PC:FF)	S'il gèle les gens ne sont pas capables de régler leur *note* de chauffage
En (FF)	If there is a hard frost, people are unable to pay their *bills*.

To keep the methods simple and language-independent, no lemmatization was used. We took only sentences that had the exact form of the French and English word as described in Table 1. Some improvement might be achieved when using lemmatization. We wanted to see how well we can do by using sentences as they are extracted from the parallel corpus, with no additional pre-processing and without removing any noise that might be introduced during the collection process.

From the extracted sentences, we used 2/3 of the sentences for training (seeds) and 1/3 for testing when applying both the supervised and semi-supervised approach. In Table 3 we present the number of seeds used for training and testing.

We will show in Section 6, that even though we started with a small amount of seeds from a certain domain – the nature of the parallel corpus that we had, an improvement can be obtained in discriminating the senses of partial cognates using free text from other domains.

Table 3. Number of parallel sentences used as seeds.

Partial Cognates	Train CG	Train FF	Test CG	Test FF
Blanc	54	78	28	39
Circulation	213	75	107	38
Client	105	88	53	45
Corps	88	82	44	42
Détail	120	80	60	41
Mode	76	104	126	53
Note	250	138	126	68
Police	154	94	78	48
Responsable	200	162	100	81
Route	69	90	35	46
AVERAGE	132.9	99.1	66.9	50.1

5 Methods

In this section we describe the supervised and the semi-supervised methods that we use in our experiments. We will also describe the data sets that we used for the monolingual and bilingual bootstrapping technique.

For both methods we have the same goal: to determine which of the two senses (the cognate or the false-friend sense) of a partial-cognate word is present in a test sentence. The classes in which we classify a sentence that contains a partial cognate are: COG (cognate) and FF (false-friend).

5.1 Supervised Method

For both the supervised and semi-supervised method we used the bag-of-words (BOW) approach of modeling context, with binary values for the features. The features were words from the training corpus that appeared at least 3 times in the training sentences. We removed the stopwords from the features. A list of stopwords for English and one for French was used. We ran experiments when we kept the stopwords as features but the results did not improve.

Since we wanted to learn the contexts in which a partial cognate has a cognate sense and the contexts in which it has a false-friend sense, the cognate and false friend words were not taken into account as features. Leaving them in would mean to indicate the classes, when applying the methods for the English sentences since all the sentences with the cognate sense contain the cognate word and all the false-friend sentences do not contain it. For the French side all collected sentences contain the partial cognate word, the same for both senses.

As a baseline for the experiments that we present we used the ZeroR classifier from WEKA[6], which predicts the class that is the most frequent in the training corpus. The classifiers for which we report results are: Naïve Bayes with a kernel estimator, Decision Trees - J48, and a Support Vector Machine implementation - SMO. All the classifiers can be found in the WEKA package. We used these classifiers because we wanted to have a probabilistic, a decision-based and a functional classifier. The decision tree classifier allows us to see which features are most discriminative.

Experiments were performed with other classifiers and with different levels of tuning, on a 10-fold cross validation approach as well; the classifiers we mentioned above were consistently the ones that obtained the best accuracy results.

The supervised method used in our experiments consists in training the classifiers on the

[6] http://www.cs.waikato.ac.nz/ml/weka/

automatically-collected training seed sentences, for each partial cognate, and then test their performance on the testing set. Results for this method are presented later, in Table 5.

5.2 Semi-Supervised Method

For the semi-supervised method we add unlabelled examples from monolingual corpora: the French newspaper LeMonde[7] 1994, 1995 (LM), and the BNC[8] corpus, different domain corpora than the seeds. The procedure of adding and using this unlabeled data is described in the Monolingual Bootstrapping (MB) and Bilingual Bootstrapping (BB) sections.

5.2.1 Monolingual Bootstrapping

The monolingual bootstrapping algorithm that we used for experiments on French sentences (MB-F) and on English sentences (MB-E) is:

```
For each pair of partial cognates (PC)
  1. Train a classifier on the training seeds – using the BOW approach and a NB-K classifier with attribute selection on the features.
  2. Apply the classifier on unlabeled data – sentences that contain the PC word, extracted from LeMonde (MB-F) or from BNC (MB-E)
  3. Take the first k newly classified sentences, both from the COG and FF class and add them to the training seeds (the most confident ones – the prediction accuracy greater or equal than a threshold =0.85)
  4. Rerun the experiments training on the new training set
  5. Repeat steps 2 and 3 for t times
endFor
```

For the first step of the algorithm we used NB-K classifier because it was the classifier that consistently performed better. We chose to perform attribute selection on the features after we tried the method without attribute selection. We obtained better results when using attribute selection. This sub-step was performed with the WEKA tool, the Chi-Square attribute selection was chosen.

In the second step of the MB algorithm the classifier that was trained on the training seeds was then used to classify the unlabeled data that was collected from the two additional resources. For the MB algorithm on the French side we trained the classifier on the French side of the training seeds and then we applied the classifier to classify the sentences that were extracted from LeMonde and contained the partial cognate. The same approach was used for the MB on the English side only this time we were using the English side of the training seeds for training the classifier and the BNC corpus to extract new examples. In fact, the MB-E step is needed only for the BB method.

Only the sentences that were classified with a probability greater than 0.85 were selected for later use in the bootstrapping algorithm.

The number of sentences that were chosen from the new corpora and used in the first step of the MB and BB are presented in Table 4.

Table 4. Number of sentences selected from the LeMonde and BNC corpus.

PC	LM COG	LM FF	BNC COG	BNC FF
Blanc	45	250	0	241
Circulation	250	250	70	180
Client	250	250	77	250
Corps	250	250	131	188
Détail	250	163	158	136
Mode	151	250	176	262
Note	250	250	178	281
Police	250	250	186	200
Responsable	250	250	177	225
Route	250	250	217	118

For the partial-cognate *Blanc* with the cognate sense, the number of sentences that had a probability distribution greater or equal with the threshold was low. For the rest of partial cognates the number of selected sentences was limited by the value of parameter ***k*** in the algorithm.

5.2.2 Bilingual Bootstrapping

The algorithm for bilingual bootstrapping that we propose and tried in our experiments is:

1. Translate the English sentences that were collected in the MB-E step into French using an online MT[9] tool and add them to the French seed training data.
*2. Repeat the MB-F and MB-E steps for **T** times.*

For the both monolingual and bilingual bootstrapping techniques the value of the parameters **t** and **T** is 1 in our experiments.

[7] http://www.lemonde.fr/
[8] http://www.natcorp.ox.ac.uk/
[9] http://www.freetranslation.com/free/web.asp

6 Evaluation and Results

In this section we present the results that we obtained with the supervised and semi-supervised methods that we applied to disambiguate partial cognates.

Due to space issue we show results only for testing on the testing sets and not for the 10-fold cross validation experiments on the training data. For the same reason, we present the results that we obtained only with the French side of the parallel corpus, even though we trained classifiers on the English sentences as well. The results for the 10-fold cross validation and for the English sentences are not much different than the ones from Table 5 that describe the supervised method results on French sentences.

Table 5. Results for the Supervised Method.

PC	ZeroR	NB-K	Trees	SMO
Blanc	58%	95.52%	98.5%	98.5%
Circulation	74%	91.03%	80%	89.65%
Client	54.08%	67.34%	66.32%	61.22%
Corps	51.16%	62%	61.62%	69.76%
Détail	59.4%	85.14%	85.14%	87.12%
Mode	58.24%	89.01%	89.01%	90%
Note	64.94%	89.17%	77.83%	85.05%
Police	61.41%	79.52%	93.7%	94.48%
Responsable	55.24%	85.08%	70.71%	75.69%
Route	56.79%	54.32%	56.79%	56.79%
AVERAGE	59.33%	**80.17%**	77.96%	80.59%

Table 6 and Table 7 present results for the MB and BB. More experiments that combined MB and BB techniques were also performed. The results are presented in Table 9.

Our goal is to disambiguate partial cognates in general, not only in the particular domain of Hansard and EuroParl. For this reason we used another set of automatically determined sentences from a multi-domain parallel corpus.

The set of new sentences (multi-domain) was extracted in the same manner as the seeds from Hansard and EuroParl. The new parallel corpus is a small one, approximately 1.5 million words, but contains texts from different domains: magazine articles, modern fiction, texts from international organizations and academic textbooks. We are using this set of sentences in our experiments to show that our methods perform well on multi-domain corpora and also because our aim is to be able to disambiguate PC in different domains. From this parallel corpus we were able to extract the number of sentences shown in Table 8.

With this new set of sentences we performed different experiments both for MB and BB. All results are described in Table 9. Due to space issue we report the results only on the average that we obtained for all the 10 pairs of partial cognates.

The symbols that we use in Table 9 represent:

S – the seed training corpus, TS – the seed test set, BNC and LM – sentences extracted from LeMonde and BNC (Table 4), and NC – the sentences that were extracted from the multi-domain new corpus. When we use the + symbol we put together all the sentences extracted from the respective corpora.

Table 6. Monolingual Bootstrapping on the French side.

PC	ZeroR	NB-K	Dec.Tree	SMO
Blanc	58.20%	97.01%	97.01%	98.5%
Circulation	73.79%	90.34%	70.34%	84.13%
Client	54.08%	71.42%	54.08%	64.28%
Corps	51.16%	78%	56.97%	69.76%
Détail	59.4%	88.11%	85.14%	82.17%
Mode	58.24%	89.01%	90.10%	85%
Note	64.94%	85.05%	71.64%	80.41%
Police	61.41%	71.65%	92.91%	71.65%
Responsable	55.24%	87.29%	77.34%	81.76%
Route	56.79%	51.85%	56.79%	56.79%
AVERAGE	59.33%	**80.96%**	75.23%	77.41%

Table 7. Bilingual Bootstrapping.

PC	ZeroR	NB-K	Dec.Tree	SMO
Blanc	58.2%	95.52%	97.01%	98.50%
Circulation	73.79%	92.41%	63.44%	87.58%
Client	45.91%	70.4%	45.91%	63.26%
Corps	48.83%	83%	67.44%	82.55%
Détail	59%	91.08%	85.14%	86.13%
Mode	58.24%	87.91%	90.1%	87%
Note	64.94%	85.56%	77.31%	79.38%
Police	61.41%	80.31%	96.06%	96.06%
Responsable	44.75%	87.84%	74.03%	79.55%
Route	43.2%	60.49%	45.67%	64.19%
AVERAGE	55.87%	**83.41%**	74.21%	82.4%

Table 8. New Corpus (NC) sentences.

PC	COG	FF
Blanc	18	222
Circulation	26	10
Client	70	44
Corps	4	288
Détail	50	0
Mode	166	12
Note	214	20
Police	216	6
Responsable	104	66
Route	6	100

6.1 Discussion of the Results

The results of the experiments and the methods that we propose show that we can use with success unlabeled data to learn from, and that the noise that is introduced due to the seed set collection is tolerable by the ML techniques that we use.

Some results of the experiments we present in Table 9 are not as good as others. What is important to notice is that every time we used MB or BB or both, there was an improvement. For some experiments MB did better, for others BB was the method that improved the performance; nonetheless for some combinations MB together with BB was the method that worked best.

In Tables 5 and 7 we show that BB improved the results on the NB-K classifier with 3.24%, compared with the supervised method (no bootstrapping), when we tested only on the test set (TS), the one that represents 1/3 of the initially-collected parallel sentences. This improvement is not statistically significant, according to a t-test.

In Table 9 we show that our proposed methods bring improvements for different combinations of training and testing sets. Table 9, lines 1 and 2 show that BB with NB-K brought an improvement of 1.95% from no bootstrapping, when we tested on the multi-domain corpus NC. For the same setting, there was an improvement of 1.55% when we tested on TS (Table 9, lines 6 and 8). When we tested on the combination TS+NC, again BB brought an improvement of 2.63% from no bootstrapping (Table 9, lines 10 and 12). The difference between MB and BB with this setting is 6.86% (Table 9, lines 11 and 12). According to a t-test the 1.95% and 6.86% improvements are statistically significant.

Table 9. Results for different experiments with monolingual and bilingual bootstrapping (MB and BB).

Train	Test	ZeroR	NB-K	Trees	SMO
S (no bootstrapping)	NC	67%	***71.97%***	73.75%	76.75%
S+BNC (BB)	NC	64%	**73.92%**	60.49%	74.80%
S+LM (MB)	NC	67.85%	67.03%	64.65%	65.57%
S +LM+BNC (MB+BB)	NC	64.19%	70.57%	57.03%	66.84%
S+LM+BNC (MB+BB)	TS	55.87%	81.98%	74.37%	78.76%
S+NC (no bootstr.)	TS	57.44%	***82.03%***	76.91%	80.71%
S+NC+LM (MB)	TS	57.44%	82.02%	73.78%	77.03%
S+NC+BNC (BB)	TS	56.63%	**83.58%**	68.36%	82.34%
S+NC+LM+ BNC(MB+BB)	TS	58%	83.10%	75.61%	79.05%
S (no bootstrapping)	TS+NC	62.70%	***77.20%***	77.23%	79.26%
S+LM (MB)	TS+NC	62.70%	**72.97%**	70.33%	71.97%
S+BNC (BB)	TS+NC	61.27%	**79.83%**	67.06%	78.80%
S+LM+BNC (MB+BB)	TS+NC	61.27%	77.28%	65.75%	73.87%

The number of features that were extracted from the seeds was more than double at each MB and BB experiment, showing that even though we started with seeds from a language restricted domain, the method is able to capture knowledge form different domains as well. Besides the change in the number of features, the domain of the features has also changed form the parliamentary one to others, more general, showing that the method will be able to disambiguate sentences where the partial cognates cover different types of context.

Unlike previous work that has done with monolingual or bilingual bootstrapping, we tried to disambiguate not only words that have senses that are very different e.g. *plant* – with a sense of biological plant or with the sense of factory. In our set of partial cognates the French word *route* is a difficult word to disambiguate even for humans: it has a cognate sense when it refers to a maritime or trade route and a false-friend sense when it is used as road. The same observation applies to *client (*the cognate sense is *client*, and the false friend sense is *customer*, *patron*, or *patient)* and to *circulation (*cognate in *air* or *blood circulation*, false friend in *street traffic).*

7 Conclusion and Future Work

We showed that with simple methods and using available tools we can achieve good results in the task of partial cognate disambiguation.

The accuracy might be increased by using dependencies relations, lemmatization, part-of-speech tagging – extract sentences where the partial cognate has the same POS, and other types of data representation combined with different semantic tools (e.g. decision lists, rule based systems).

In our experiments we use a machine language representation – binary feature values, and we show that nonetheless machines are capable of learning from new information, using an iterative approach, similar to the learning process of humans. New information was collected and extracted by classifiers when additional corpora were used for training.

In addition to the applications that we mentioned in Section 1, partial cognates can also be useful in Computer-Assisted Language Learning (CALL) tools. Search engines for E-Learning can find useful a partial cognate annotator. A teacher that prepares a test to be integrated into a CALL tool can save time by using our methods to automatically disambiguate partial cognates, even though the automatic classifications need to be checked by the teacher.

In future work we plan to try different representations of the data, to use knowledge of the relations that exists between the partial cognate and the context words, and to run experiments when we iterate the MB and BB steps more than once.

References

Susane Carroll 1992. On Cognates. *Second Language Research,* 8(2):93-119

Mona Diab and Philip Resnik. 2002. An unsupervised method for word sense tagging using parallel corpora. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics (ACL 2002),* Philadelphia, pp. 255-262.

S. M. Gass. 1987. The use and acquisition of the second language lexicon (Special issue). *Studies in Second Language Acquisition,* 9 (2).

Jacques B. M. Guy. 1994. An algorithm for identifying cognates in bilingual word lists and its applicability to machine translation. *Journal of Quantitative Linguistics*, 1(1):35-42.

Marti Hearst 1991. Noun homograph disambiguation using local context in large text corpora. *7th Annual Conference of the University of Waterloo Center for the new OED and Text Research,* Oxford.

W.J.B Van Heuven, A. Dijkstra, and J. Grainger. 1998. Orthographic neighborhood effects in bilingual word recognition. *Journal of Memory and Language* 39: 458-483.

John Hewson 1993. A Computer-Generated Dictionary of Proto-Algonquian. Ottawa: Canadian Museum of Civilization.

Nancy Ide. 2000 Cross-lingual sense determination: Can it work? *Computers and the Humanities,* 34:1-2, *Special Issue on the Proceedings of the SIGLEX SENSEVAL Workshop,* pp.223-234.

Grzegorz Kondrak. 2004. Combining Evidence in Cognate Identification. *Proceedings of Canadian AI 2004: 17th Conference of the Canadian Society for Computational Studies of Intelligence,* pp.44-59.

Grzegorz Kondrak. 2001. Identifying Cognates by Phonetic and Semantic Similarity. *Proceedings of NAACL 2001: 2nd Meeting of the North American Chapter of the Association for Computational Linguistics,* pp.103-110.

Raymond LeBlanc and Hubert Séguin. 1996. Les congénères homographes et parographes anglais-français. *Twenty-Five Years of Second Language Teaching at the University of Ottawa,* pp.69-91.

Hang Li and Cong Li. 2004. Word translation disambiguation using bilingual bootstrap. *Computational Linguistics*, 30(1):1-22.

John B. Lowe and Martine Mauzaudon. 1994. The reconstruction engine: a computer implementation of the comparative method. *Computational Linguistic*s, 20:381-417.

Hakan Ringbom. 1987. *The Role of the First Language in Foreign Language Learning*. Multilingual Matters Ltd., Clevedon, England.

Dan Tufis, Ion Radu, Nancy Ide 2004. Fine-Grained Word Sense Disambiguation Based on Parallel Corpora, Word Alignment, Word Clustering and Aligned WordNets. *Proceedings of the 20th International Conference on Computational Linguistics,* COLING 2004, Geneva, pp. 1312-1318.

David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33th Annual Meeting of the Association for Computational Linguistics,* Cambridge, MA, pp 189-196.

Direct Word Sense Matching for Lexical Substitution

Ido Dagan[1], **Oren Glickman**[1], **Alfio Gliozzo**[2], **Efrat Marmorshtein**[1], **Carlo Strapparava**[2]
[1]Department of Computer Science, Bar Ilan University, Ramat Gan, 52900, Israel
[2]ITC-Irst, via Sommarive, I-38050, Trento, Italy

Abstract

This paper investigates conceptually and empirically the novel *sense matching* task, which requires to recognize whether the senses of two synonymous words match in context. We suggest direct approaches to the problem, which avoid the intermediate step of explicit word sense disambiguation, and demonstrate their appealing advantages and stimulating potential for future research.

1 Introduction

In many language processing settings it is needed to recognize that a given word or term may be substituted by a synonymous one. In a typical information seeking scenario, an information need is specified by some given *source* words. When looking for texts that match the specified need the source words might be substituted with synonymous *target* words. For example, given the source word 'weapon' a system may substitute it with the target synonym 'arm'.

This scenario, which is generally referred here as *lexical substitution*, is a common technique for increasing recall in Natural Language Processing (NLP) applications. In Information Retrieval (IR) and Question Answering (QA) it is typically termed query/question expansion (Moldovan and Mihalcea, 2000; Negri, 2004). Lexical Substitution is also commonly applied to identify synonyms in text summarization, for paraphrasing in text generation, or is integrated into the features of supervised tasks such as Text Categorization and Information Extraction. Naturally, lexical substitution is a very common first step in textual entailment recognition, which models semantic inference between a pair of texts in a generalized application independent setting (Dagan et al., 2005).

To perform lexical substitution NLP applications typically utilize a knowledge source of synonymous word pairs. The most commonly used resource for lexical substitution is the manually constructed WordNet (Fellbaum, 1998). Another option is to use statistical word similarities, such as in the database constructed by Dekang Lin (Lin, 1998). We generically refer to such resources as *substitution lexicons*.

When using a substitution lexicon it is assumed that there are *some* contexts in which the given synonymous words share the same meaning. Yet, due to polysemy, it is needed to verify that the senses of the two words do indeed match in a given context. For example, there are contexts in which the source word 'weapon' may be substituted by the target word 'arm'; however one should recognize that 'arm' has a different sense than 'weapon' in sentences such as "repetitive movements could cause injuries to hands, wrists and arms."

A commonly proposed approach to address sense matching in lexical substitution is applying Word Sense Disambiguation (WSD) to identify the senses of the source and target words. Then, substitution is applied only if the words have the same sense (or synset, in WordNet terminology). In settings in which the source is given as a single term without context, sense disambiguation is performed only for the target word; substitution is then applied only if the target word's sense matches at least one of the possible senses of the source word.

One might observe that such application of WSD addresses the task at hand in a somewhat indirect manner. In fact, lexical substitution only requires knowing that the source and target senses

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 449–456,
Sydney, July 2006. ©2006 Association for Computational Linguistics

do match, but it does not require that the matching senses will be explicitly identified. Selecting explicitly the right sense in context, which is then followed by verifying the desired matching, might be solving a harder intermediate problem than required. Instead, we can define the *sense matching* problem directly as a binary classification task for a pair of synonymous source and target words. This task requires to decide whether the senses of the two words do or do not match in a given context (but it does not require to identify explicitly the identity of the matching senses).

A highly related task was proposed in (McCarthy, 2002). McCarthy's proposal was to ask systems to suggest possible "semantically similar replacements" of a target word in context, where alternative replacements should be grouped together. While this task is somewhat more complicated as an evaluation setting than our binary recognition task, it was motivated by similar observations and applied goals. From another perspective, sense matching may be viewed as a lexical sub-case of the general textual entailment recognition setting, where we need to recognize whether the meaning of the target word "entails" the meaning of the source word in a given context.

This paper provides a first investigation of the sense matching problem. To allow comparison with the classical WSD setting we derived an evaluation dataset for the new problem from the Senseval-3 English lexical sample dataset (Mihalcea and Edmonds, 2004). We then evaluated alternative supervised and unsupervised methods that perform sense matching either *indirectly* or *directly* (i.e. with or without the intermediate sense identification step). Our findings suggest that in the supervised setting the results of the direct and indirect approaches are comparable. However, addressing directly the binary classification task has practical advantages and can yield high precision values, as desired in precision-oriented applications such as IR and QA.

More importantly, direct sense matching sets the ground for implicit unsupervised approaches that may utilize practically unlimited volumes of unlabeled training data. Furthermore, such approaches circumvent the sisyphean need for specifying explicitly a set of stipulated senses. We present an initial implementation of such an approach using a one-class classifier, which is trained on unlabeled occurrences of the source word and applied to occurrences of the target word. Our current results outperform the unsupervised baseline and put forth a whole new direction for future research.

2 WSD and Lexical Expansion

Despite certain initial skepticism about the usefulness of WSD in practical tasks (Voorhees, 1993; Sanderson, 1994), there is some evidence that WSD can improve performance in typical NLP tasks such as IR and QA. For example, (Shütze and Pederson, 1995) gives clear indication of the potential for WSD to improve the precision of an IR system. They tested the use of WSD on a standard IR test collection (TREC-1B), improving precision by more than 4%.

The use of WSD has produced successful experiments for query expansion techniques. In particular, some attempts exploited WordNet to enrich queries with semantically-related terms. For instance, (Voorhees, 1994) manually expanded 50 queries over the TREC-1 collection using synonymy and other WordNet relations. She found that the expansion was useful with short and incomplete queries, leaving the task of proper automatic expansion as an open problem.

(Gonzalo et al., 1998) demonstrates an increment in performance over an IR test collection using the sense data contained in SemCor over a purely term based model. In practice, they experimented searching SemCor with disambiguated and expanded queries. Their work shows that a WSD system, even if not performing perfectly, combined with synonymy enrichment increases retrieval performance.

(Moldovan and Mihalcea, 2000) introduces the idea of using WordNet to extend Web searches based on semantic similarity. Their results showed that WSD-based query expansion actually improves retrieval performance in a Web scenario. Recently (Negri, 2004) proposed a sense-based relevance feedback scheme for query enrichment in a QA scenario (TREC-2003 and ACQUAINT), demonstrating improvement in retrieval performance.

While all these works clearly show the potential usefulness of WSD in practical tasks, nonetheless they do not necessarily justify the efforts for refining fine-grained sense repositories and for building large sense-tagged corpora. We suggest that the sense matching task, as presented in the introduction, may relieve major drawbacks of applying WSD in practical scenarios.

3 Problem Setting and Dataset

To investigate the direct sense matching problem it is necessary to obtain an appropriate dataset of examples for this binary classification task, along with gold standard annotation. While there is no such standard (application independent) dataset available it is possible to derive it automatically from existing WSD evaluation datasets, as described below. This methodology also allows comparing direct approaches for sense matching with classical indirect approaches, which apply an intermediate step of identifying the most likely WordNet sense.

We derived our dataset from the Senseval-3 English lexical sample dataset (Mihalcea and Edmonds, 2004), taking all 25 nouns, adjectives and adverbs in this sample. Verbs were excluded since their sense annotation in Senseval-3 is not based on WordNet senses. The Senseval dataset includes a set of example occurrences in context for each word, split to training and test sets, where each example is manually annotated with the corresponding WordNet synset.

For the sense matching setting we need examples of pairs of *source-target* synonymous words, where at least one of these words should occur in a given context. Following an applicative motivation, we mimic an IR setting in which a single source word query is expanded (substituted) by a synonymous target word. Then, it is needed to identify contexts in which the target word appears in a sense that matches the source word. Accordingly, we considered each of the 25 words in the Senseval sample as a target word for the sense matching task. Next, we had to pick for each target word a corresponding synonym to play the role of the source word. This was done by creating a list of all WordNet synonyms of the target word, under all its possible senses, and picking randomly one of the synonyms as the source word. For example, the word 'disc' is one of the words in the Senseval lexical sample. For this target word the synonym 'record' was picked, which matches 'disc' in its musical sense. Overall, 59% of all possible synsets of our target words included an additional synonym, which could play the role of the source word (that is, 41% of the synsets consisted of the target word only). Similarly, 62% of the test examples of the target words were annotated by a synset that included an additional synonym.

While creating source-target synonym pairs it was evident that many WordNet synonyms correspond to very infrequent senses or word usages, such as the WordNet synonyms *germ* and *source*. Such source synonyms are useless for evaluating sense matching with the target word since the senses of the two words would rarely match in perceivable contexts. In fact, considering our motivation for lexical substitution, it is usually desired to exclude such obscure synonym pairs from substitution lexicons in practical applications, since they would mostly introduce noise to the system. To avoid this problem the list of WordNet synonyms for each target word was filtered by a lexicographer, who excluded manually obscure synonyms that seemed worthless in practice. The source synonym for each target word was then picked randomly from the filtered list. Table 1 shows the 25 source-target pairs created for our experiments. In future work it may be possible to apply automatic methods for filtering infrequent sense correspondences in the dataset, by adopting algorithms such as in (McCarthy et al., 2004).

Having source-target synonym pairs, a classification instance for the sense matching task is created from each example occurrence of the target word in the Senseval dataset. A classification instance is thus defined by a pair of source and target words and a given occurrence of the target word in context. The instance should be classified as *positive* if the sense of the target word in the given context matches one of the possible senses of the source word, and as *negative* otherwise. Table 2 illustrates positive and negative example instances for the source-target synonym pair 'record-disc', where only occurrences of 'disc' in the musical sense are considered positive.

The gold standard annotation for the binary sense matching task can be derived automatically from the Senseval annotations and the corresponding WordNet synsets. An example occurrence of the target word is considered positive if the annotated synset for that example includes also the source word, and Negative otherwise. Notice that different positive examples might correspond to different senses of the target word. This happens when the source and target share several senses, and hence they appear together in several synsets. Finally, since in Senseval an example may be an-

source-target	*source-target*	*source-target*	*source-target*	*source-target*
statement-argument	subdivision-arm	atm-atmosphere	hearing-audience	camber-bank
level-degree	deviation-difference	dissimilar-different	trouble-difficulty	record-disc
raging-hot	ikon-image	crucial-important	sake-interest	bare-simple
opinion-judgment	arrangement-organization	newspaper-paper	company-party	substantial-solid
execution-performance	design-plan	protection-shelter	variety-sort	root-source

Table 1: Source and target pairs

sentence	*annotation*
This is anyway a stunning *disc*, thanks to the playing of the Moscow Virtuosi with Spivakov.	positive
He said computer networks would not be affected and copies of information should be made on floppy *discs*.	negative
Before the dead soldier was placed in the ditch his personal possessions were removed, leaving one *disc* on the body for identification purposes	negative

Table 2: positive and negative examples for the source-target synonym pair 'record-disc'

notated with more than one sense, it was considered positive if any of the annotated synsets for the target word includes the source word.

Using this procedure we derived gold standard annotations for all the examples in the Senseval-3 training section for our 25 target words. For the test set we took up to 40 test examples for each target word (some words had fewer test examples), yielding 913 test examples in total, out of which 239 were positive. This test set was used to evaluate the sense matching methods described in the next section.

4 Investigated Methods

As explained in the introduction, the sense matching task may be addressed by two general approaches. The traditional *indirect* approach would first disambiguate the target word relative to a predefined set of senses, using standard WSD methods, and would then verify that the selected sense matches the source word. On the other hand, a *direct* approach would address the binary sense matching task directly, without selecting explicitly a concrete sense for the target word. This section describes the alternative methods we investigated under supervised and unsupervised settings. The supervised methods utilize manual sense annotations for the given source and target words while unsupervised methods do not require any annotated sense examples. For the indirect approach we assume the standard WordNet sense repository and corresponding annotations of the target words with WordNet synsets.

4.1 Feature set and classifier

As a vehicle for investigating different classification approaches we implemented a "vanilla" state of the art architecture for WSD. Following common practice in feature extraction (e.g. (Yarowsky, 1994)), and using the mxpost[1] part of speech tagger and WordNet's lemmatization, the following feature set was used: bag of word lemmas for the context words in the preceding, current and following sentence; unigrams of lemmas and parts of speech in a window of +/- three words, where each position provides a distinct feature; and bigrams of lemmas in the same window. The SVM-Light (Joachims, 1999) classifier was used in the supervised settings with its default parameters. To obtain a multi-class classifier we used a standard one-vs-all approach of training a binary SVM for each possible sense and then selecting the highest scoring sense for a test example.

To verify that our implementation provides a reasonable replication of state of the art WSD we applied it to the standard Senseval-3 Lexical Sample WSD task. The obtained accuracy[2] was 66.7%, which compares reasonably with the mid-range of systems in the Senseval-3 benchmark (Mihalcea and Edmonds, 2004). This figure is just a few percent lower than the (quite complicated) best Senseval-3 system, which achieved about 73% accuracy, and it is much higher than the standard Senseval baselines. We thus regard our classifier as a fair vehicle for comparing the alternative approaches for sense matching on equal grounds.

[1] ftp://ftp.cis.upenn.edu/pub/adwait/jmx/jmx.tar.gz

[2] The standard classification accuracy measure equals precision and recall as defined in the Senseval terminology when the system classifies all examples, with no abstentions.

4.2 Supervised Methods

4.2.1 Indirect approach

The *indirect* approach for sense matching follows the traditional scheme of performing WSD for lexical substitution. First, the WSD classifier described above was trained for the target words of our dataset, using the Senseval-3 sense annotated training data for these words. Then, the classifier was applied to the test examples of the target words, selecting the most likely sense for each example. Finally, an example was classified as positive if the selected synset for the target word includes the source word, and as negative otherwise.

4.2.2 Direct approach

As explained above, the *direct* approach addresses the binary sense matching task directly, without selecting explicitly a sense for the target word. In the supervised setting it is easy to obtain such a binary classifier using the annotation scheme described in Section 3. Under this scheme an example was annotated as positive (for the binary sense matching task) if the source word is included in the Senseval gold standard synset of the target word. We trained the classifier using the set of Senseval-3 training examples for each target word, considering their derived binary annotations. Finally, the trained classifier was applied to the test examples of the target words, yielding directly a binary positive-negative classification.

4.3 Unsupervised Methods

It is well known that obtaining annotated training examples for WSD tasks is very expensive, and is often considered infeasible in unrestricted domains. Therefore, many researchers investigated unsupervised methods, which do not require annotated examples. Unsupervised approaches have usually been investigated within Senseval using the "All Words" dataset, which does not include training examples. In this paper we preferred using the same test set which was used for the supervised setting (created from the Senseval-3 "Lexical Sample" dataset, as described above), in order to enable comparison between the two settings. Naturally, in the unsupervised setting the sense labels in the training set were not utilized.

4.3.1 Indirect approach

State-of-the-art unsupervised WSD systems are quite complex and they are not easy to be replicated. Thus, we implemented the unsupervised version of the Lesk algorithm (Lesk, 1986) as a reference system, since it is considered a standard simple baseline for unsupervised approaches. The Lesk algorithm is one of the first algorithms developed for semantic disambiguation of all-words in unrestricted text. In its original unsupervised version, the only resource required by the algorithm is a machine readable dictionary with one definition for each possible word sense. The algorithm looks for words in the sense definitions that overlap with context words in the given sentence, and chooses the sense that yields maximal word overlap. We implemented a version of this algorithm using WordNet sense-definitions with context length of ±10 words before and after the target word.

4.3.2 The direct approach: one-class learning

The unsupervised settings for the direct method are more problematic because most of unsupervised WSD algorithms (such as the Lesk algorithm) rely on dictionary definitions. For this reason, standard unsupervised techniques cannot be applied in a direct approach for sense matching, in which the only external information is a substitution lexicon.

In this subsection we present a direct unsupervised method for sense matching. It is based on the assumption that typical contexts in which both the source and target words appear correspond to their matching senses. Unlabeled occurrences of the source word can then be used to provide evidence for lexical substitution because they allow us to recognize whether the sense of the target word matches that of the source. Our strategy is to represent in a learning model the typical contexts of the source word in unlabeled training data. Then, we exploit such model to match the contexts of the target word, providing a decision criterion for sense matching. In other words, we expect that under a matching sense the target word would occur in prototypical contexts of the source word.

To implement such approach we need a learning technique that does not rely on the availability of negative evidence, that is, a one-class learning algorithm. In general, the classification performance of one-class approaches is usually quite poor, if compared to supervised approaches for the same tasks. However, in many practical settings one-class learning is the only available solution.

For our experiments we adopted the one-class SVM learning algorithm (Schölkopf et al., 2001) implemented in the LIBSVM package,[3] and represented the unlabeled training examples by adopting the feature set described in Subsection 4.1. Roughly speaking, a one-class SVM estimates the smallest hypersphere enclosing most of the training data. New test instances are then classified positively if they lie inside the sphere, while outliers are regarded as negatives. The ratio between the width of the enclosed region and the number of misclassified training examples can be varied by setting the parameter $\nu \in (0, 1)$. Smaller values of ν will produce larger positive regions, with the effect of increasing recall.

The appealing advantage of adopting one-class learning for sense matching is that it allows us to define a very elegant learning scenario, in which it is possible to train "off-line" a different classifier for each (source) word in the lexicon. Such a classifier can then be used to match the sense of any possible target word for the source which is given in the substitution lexicon. This is in contrast to the direct supervised method proposed in Subsection 4.2, where a different classifier for each pair of source - target words has to be defined.

5 Evaluation

5.1 Evaluation measures and baselines

In the lexical substitution (and expansion) setting, the standard WSD metrics (Mihalcea and Edmonds, 2004) are not suitable, because we are interested in the binary decision of whether the target word matches the sense of a given source word. In analogy to IR, we are more interested in positive assignments, while the opposite case (i.e. when the two words cannot be substituted) is less interesting. Accordingly, we utilize the standard definitions of precision, recall and F_1 typically used in IR benchmarks. In the rest of this section we will report micro averages for these measures on the test set described in Section 3.

Following the Senseval methodology, we evaluated two different baselines for unsupervised and supervised methods. The random baseline, used for the unsupervised algorithms, was obtained by choosing either the positive or the negative class at random resulting in $P = 0.262$, $R = 0.5$, $F_1 = 0.344$. The *Most Frequent* baseline has been used for the supervised algorithms and is obtained by assigning the positive class when the percentage of positive examples in the training set is above 50%, resulting in $P = 0.65$, $R = 0.41$, $F_1 = 0.51$.

5.2 Supervised Methods

Both the indirect and the direct supervised methods presented in Subsection 4.2 have been tested and compared to the most frequent baseline.

Indirect. For the indirect methodology we trained the supervised WSD system for each target word on the sense-tagged training sample. As described in Subsection 4.2, we implemented a simple SVM-based WSD system (see Section 4.2) and applied it to the sense-matching task. Results are reported in Table 3. The direct strategy surpasses the most frequent baseline F1 score, but the achieved precision is still below it. We note that in this multi-class setting it is less straightforward to tradeoff recall for precision, as all senses compete with each other.

Direct. In the direct supervised setting, sense matching is performed by training a binary classifier, as described in Subsection 4.2.

The advantage of adopting a binary classification strategy is that the precision/recall tradeoff can be tuned in a meaningful way. In SVM learning, such tuning is achieved by varying the parameter J, that allows us to modify the cost function of the SVM learning algorithm. If $J = 1$ (default), the weight for the positive examples is equal to the weight for the negatives. When $J > 1$, negative examples are penalized (increasing recall), while, whenever $0 < J < 1$, positive examples are penalized (increasing precision). Results obtained by varying this parameter are reported in Figure 1.

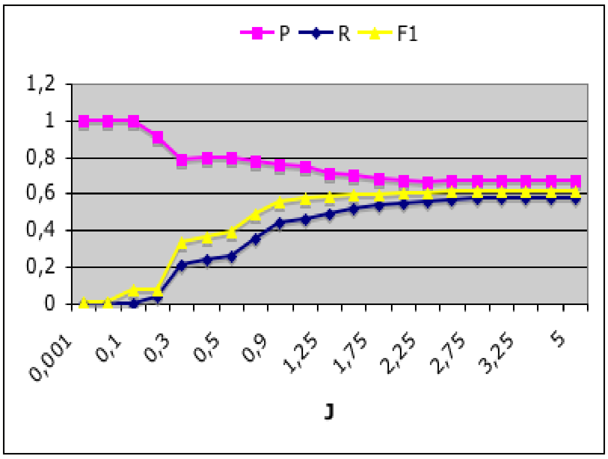


Figure 1: Direct supervised results varying J

[3] Freely available from www.csie.ntu.edu.tw/ /∼cjlin/libsvm.

Supervised		P	R	F_1	Unsupervised		P	R	F_1
Most Frequent	Baseline	0.65	0.41	0.51	Random	Baseline	0.26	0.50	0.34
Multiclass SVM	Indirect	0.59	0.63	0.61	Lesk	Indirect	0.24	0.19	0.21
Binary SVM ($J = 0.5$)	Direct	0.80	0.26	0.39	One-Class $\nu = 0.3$	Direct	0.26	0.72	0.39
Binary SVM ($J = 1$)	Direct	0.76	0.46	0.57	One-Class $\nu = 0.5$	Direct	0.29	0.56	0.38
Binary SVM ($J = 2$)	Direct	0.68	0.53	0.60	One-Class $\nu = 0.7$	Direct	0.28	0.36	0.32
Binary SVM ($J = 3$)	Direct	0.69	0.55	0.61	One-Class $\nu = 0.9$	Direct	0.23	0.10	0.14

Table 3: Classification results on the sense matching task

Adopting the standard parameter settings (i.e. $J = 1$, see Table 3), the F_1 of the system is slightly lower than for the indirect approach, while it reaches the indirect figures when J increases. More importantly, reducing J allows us to boost precision towards 100%. This feature is of great interest for lexical substitution, particularly in precision oriented applications like IR and QA, for filtering irrelevant candidate answers or documents.

5.3 Unsupervised methods

Indirect. To evaluate the indirect unsupervised settings we implemented the Lesk algorithm, described in Subsection 4.3.1, and evaluated it on the sense matching task. The obtained figures, reported in Table 3, are clearly below the baseline, suggesting that simple unsupervised indirect strategies cannot be used for this task. In fact, the error of the first step, due to low WSD accuracy of the unsupervised technique, is propagated in the second step, producing poor sense matching. Unfortunately, state-of-the-art unsupervised systems are actually not much better than Lesk on all-words task (Mihalcea and Edmonds, 2004), discouraging the use of unsupervised indirect methods for the sense matching task.

Direct. Conceptually, the most appealing solution for the sense matching task is the one-class approach proposed for the direct method (Section 4.3.2). To perform our experiments, we trained a different one-class SVM for each source word, using a sample of its unlabeled occurrences in the BNC corpus as training set. To avoid huge training sets and to speed up the learning process, we fixed the maximum number of training examples to 10000 occurrences per word, collecting on average about 6500 occurrences per word.

For each target word in the test sample, we applied the classifier of the corresponding source word. Results for different values of ν are reported in Figure 2 and summarized in Table 3.

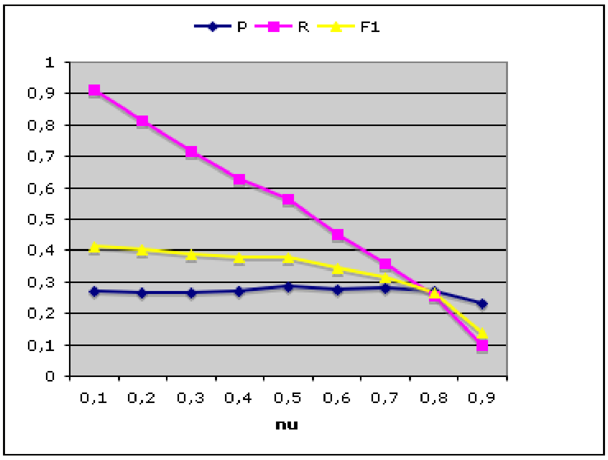


Figure 2: One-class evaluation varying ν

While the results are somewhat above the baseline, just small improvements in precision are reported, and recall is higher than the baseline for $\nu < 0.6$. Such small improvements may suggest that we are following a relevant direction, even though they may not be useful yet for an applied sense-matching setting.

Further analysis of the classification results for each word revealed that optimal F_1 values are obtained by adopting different values of ν for different words. In the optimal (in retrospect) parameter settings for each word, performance for the test set is noticeably boosted, achieving $P = 0.40$, $R = 0.85$ and $F_1 = 0.54$. Finding a principled unsupervised way to automatically tune the ν parameter is thus a promising direction for future work.

Investigating further the results per word, we found that the correlation coefficient between the optimal ν values and the degree of polysemy of the corresponding source words is 0.35. More interestingly, we noticed a negative correlation (r = -0.30) between the achieved F_1 and the degree of polysemy of the word, suggesting that polysemous source words provide poor training models for sense matching. This can be explained by observing that polysemous source words can be substituted with the target words only for a strict subset of their senses. On the other hand, our one-class algorithm was trained on *all* the examples of the source word, which include irrelevant examples that yield noisy training sets. A possible solution may be obtained using clustering-based word sense discrimination methods (Pedersen and Bruce, 1997; Schütze, 1998), in order to train different one-class models from different sense clusters. Overall, the analysis suggests that future research may obtain better binary classifiers based just on unlabeled examples of the source word.

6 Conclusion

This paper investigated the *sense matching* task, which captures directly the polysemy problem in lexical substitution. We proposed a direct approach for the task, suggesting the advantages of natural control of precision/recall tradeoff, avoiding the need in an explicitly defined sense repository, and, most appealing, the potential for novel completely unsupervised learning schemes. We speculate that there is a great potential for such approaches, and suggest that sense matching may become an appealing problem and possible track in lexical semantic evaluations.

Acknowledgments

This work was partly developed under the collaboration ITC-irst/University of Haifa.

References

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment.

C. Fellbaum. 1998. *WordNet. An Electronic Lexical Database*. MIT Press.

J. Gonzalo, F. Verdejo, I. Chugur, and J. Cigarran. 1998. Indexing with wordnet synsets can improve text retrieval. In *ACL*, Montreal, Canada.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in kernel methods: support vector learning*, chapter 11, pages 169 – 184. MIT Press.

M. Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the ACM-SIGDOC Conference*, Toronto, Canada.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics*, pages 768–774, Morristown, NJ, USA. Association for Computational Linguistics.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004. Automatic identification of infrequent word senses. In *Proceedings of COLING*, pages 1220–1226.

Diana McCarthy. 2002. Lexical substitution as a task for wsd evaluation. In *Proceedings of the ACL-02 workshop on Word sense disambiguation*, pages 109–115, Morristown, NJ, USA. Association for Computational Linguistics.

R. Mihalcea and P. Edmonds, editors. 2004. *Proceedings of SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, July.

D. Moldovan and R. Mihalcea. 2000. Using wordnet and lexical operators to improve internet searches. *IEEE Internet Computing*, 4(1):34–43, January.

M. Negri. 2004. Sense-based blind relevance feedback for question answering. In *SIGIR-2004 Workshop on Information Retrieval For Question Answering (IR4QA)*, Sheffield, UK, July.

T. Pedersen and R. Bruce. 1997. Distinguishing word sense in untagged text. In *EMNLP*, Providence, August.

M. Sanderson. 1994. Word sense disambiguation and information retrieval. In *SIGIR*, Dublin, Ireland, June.

B. Schölkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. 2001. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471.

H. Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1).

H. Shütze and J. Pederson. 1995. Information retrieval based on word senses. In *Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas.

E. Voorhees. 1993. Using WordNet to disambiguate word sense for text retrieval. In *SIGIR*, Pittsburgh, PA.

E. Voorhees. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 17th ACM SIGIR Conference*, Dublin, Ireland, June.

D. Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and french. In *ACL*, pages 88–95, Las Cruces, New Mexico.

An Equivalent Pseudoword Solution to Chinese Word Sense Disambiguation

Zhimao Lu[+] **Haifeng Wang**[++] **Jianmin Yao**[+++] **Ting Liu**[+] **Sheng Li**[+]

[+] Information Retrieval Laboratory, School of Computer Science and Technology,
Harbin Institute of Technology, Harbin, 150001, China
{lzm, tliu, lisheng}@ir-lab.org

[++] Toshiba (China) Research and Development Center
5/F., Tower W2, Oriental Plaza, No. 1, East Chang An Ave., Beijing, 100738, China
wanghaifeng@rdc.toshiba.com.cn

[+++] School of Computer Science and Technology
Soochow University, Suzhou, 215006, China
jyao@suda.edu.cn

Abstract

This paper presents a new approach based on Equivalent Pseudowords (EPs) to tackle Word Sense Disambiguation (WSD) in Chinese language. EPs are particular artificial ambiguous words, which can be used to realize unsupervised WSD. A Bayesian classifier is implemented to test the efficacy of the EP solution on Senseval-3 Chinese test set. The performance is better than state-of-the-art results with an average F-measure of 0.80. The experiment verifies the value of EP for unsupervised WSD.

1 Introduction

Word sense disambiguation (WSD) has been a hot topic in natural language processing, which is to determine the sense of an ambiguous word in a specific context. It is an important technique for applications such as information retrieval, text mining, machine translation, text classification, automatic text summarization, and so on.

Statistical solutions to WSD acquire linguistic knowledge from the training corpus using machine learning technologies, and apply the knowledge to disambiguation. The first statistical model of WSD was built by Brown et al. (1991). Since then, most machine learning methods have been applied to WSD, including decision tree, Bayesian model, neural network, SVM, maximum entropy, genetic algorithms, and so on. For different learning methods, supervised methods usually achieve good performance at a cost of human tagging of training corpus. The precision improves with larger size of training corpus. Compared with supervised methods, unsupervised methods do not require tagged corpus, but the precision is usually lower than that of the supervised methods. Thus, knowledge acquisition is critical to WSD methods.

This paper proposes an unsupervised method based on equivalent pseudowords, which acquires WSD knowledge from raw corpus. This method first determines equivalent pseudowords for each ambiguous word, and then uses the equivalent pseudowords to replace the ambiguous word in the corpus. The advantage of this method is that it does not need parallel corpus or seed corpus for training. Thus, it can use a large-scale monolingual corpus for training to solve the data-sparseness problem. Experimental results show that our unsupervised method performs better than the supervised method.

The remainder of the paper is organized as follows. Section 2 summarizes the related work. Section 3 describes the conception of Equivalent Pseudoword. Section 4 describes EP-based Unsupervised WSD Method and the evaluation result. The last section concludes our approach.

2 Related Work

For supervised WSD methods, a knowledge acquisition bottleneck is to prepare the manually

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 457–464,
Sydney, July 2006. ©2006 Association for Computational Linguistics

tagged corpus. Unsupervised method is an alternative, which often involves automatic generation of tagged corpus, bilingual corpus alignment, etc. The value of unsupervised methods lies in the knowledge acquisition solutions they adopt.

2.1 Automatic Generation of Training Corpus

Automatic corpus tagging is a solution to WSD, which generates large-scale corpus from a small seed corpus. This is a weakly supervised learning or semi-supervised learning method. This reinforcement algorithm dates back to Gale et al. (1992a). Their investigation was based on a 6-word test set with 2 senses for each word.

Yarowsky (1994 and 1995), Mihalcea and Moldovan (2000), and Mihalcea (2002) have made further research to obtain large corpus of higher quality from an initial seed corpus. A semi-supervised method proposed by Niu et al. (2005) clustered untagged instances with tagged ones starting from a small seed corpus, which assumes that similar instances should have similar tags. Clustering was used instead of bootstrapping and was proved more efficient.

2.2 Method Based on Parallel Corpus

Parallel corpus is a solution to the bottleneck of knowledge acquisition. Ide et al. (2001 and 2002), Ng et al. (2003), and Diab (2003, 2004a, and 2004b) made research on the use of alignment for WSD.

Diab and Resnik (2002) investigated the feasibility of automatically annotating large amounts of data in parallel corpora using an unsupervised algorithm, making use of two languages simultaneously, only one of which has an available sense inventory. The results showed that word-level translation correspondences are a valuable source of information for sense disambiguation.

The method by Li and Li (2002) does not require parallel corpus. It avoids the alignment work and takes advantage of bilingual corpus.

In short, technology of automatic corpus tagging is based on the manually labeled corpus. That is to say, it still need human intervention and is not a completely unsupervised method. Large-scale parallel corpus; especially word-aligned corpus is highly unobtainable, which has limited the WSD methods based on parallel corpus.

3 Equivalent Pseudoword

This section describes how to obtain equivalent pseudowords without a seed corpus.

Monosemous words are unambiguous priori knowledge. According to our statistics, they account for 86%~89% of the instances in a dictionary and 50% of the items in running corpus, they are potential knowledge source for WSD.

A monosemous word is usually synonymous to some polysemous words. For example the words "信守, 严守, 恪守, 遵照, 遵从, 遵循, 遵守" has similar meaning as one of the senses of the ambiguous word "保守", while "康健, 强健, 健旺, 健壮, 壮健, 强壮, 精壮, 壮实, 敦实, 硬朗, 康泰, 健朗, 健硕" are the same for "健康". This is quite common in Chinese, which can be used as a knowledge source for WSD.

3.1 Definition of Equivalent Pseudoword

If the ambiguous words in the corpus are replaced with its synonymous monosemous word, then is it convenient to acquire knowledge from raw corpus? For example in table 1, the ambiguous word "把握" has three senses, whose synonymous monosemous words are listed on the right column. These synonyms contain some information for disambiguation task.

An artificial ambiguous word can be coined with the monosemous words in table 1. This process is similar to the use of general pseudowords (Gale et al., 1992b; Gaustad, 2001; Nakov and Hearst, 2003), but has some essential differences. This artificial ambiguous word need to simulate the function of the real ambiguous word, and to acquire semantic knowledge as the real ambiguous word does. Thus, we call it an *equivalent pseudoword* (EP) for its equivalence with the real ambiguous word. It's apparent that the equivalent pseudoword has provided a new way to unsupervised WSD.

把握(ba3 wo4)	S_1	信心/自信心
	S_2	握住/在握/把住/抓住/控制
	S_3	领会/理解/领悟/深谙/体会

Table 1. Synonymous Monosemous Words for the Ambiguous Word "把握"

The equivalence of the EP with the real ambiguous word is a kind of semantic synonym or similarity, which demands a maximum similarity between the two words. An ambiguous word has the same number of EPs as of senses. Each EP's sense maps to a sense of ambiguous word.

The semantic equivalence demands further equivalence at each sense level. Every corresponding sense should have the maximum similarity, which is the strictest limit to the construction of an EP.

The starting point of unsupervised WSD based on EP is that EP can substitute the original word for knowledge acquisition in model training. Every instance of each morpheme of the EP can be viewed as an instance of the ambiguous word, thus the training set can be enlarged easily. EP is a solution to data sparseness for lack of human tagging in WSD.

3.2 Basic Assumption for EP-based WSD

It is based on the following assumptions that EPs can substitute the original ambiguous word for knowledge acquisition in WSD model training.

Assumption 1: Words of the same meaning play the same role in a language. The sense is an important attribute of a word. This plays as the basic assumption in this paper.

Assumption 2: Words of the same meaning occur in similar context. This assumption is widely used in semantic analysis and plays as a basis for much related research. For example, some researchers cluster the contexts of ambiguous words for WSD, which shows good performance (Schutze, 1998).

Because an EP has a higher similarity with the ambiguous word in syntax and semantics, it is a useful knowledge source for WSD.

3.3 Design and Construction of EPs

Because of the special characteristics of EPs, it's more difficult to construct an EP than a general pseudo word. To ensure the maximum similarity between the EP and the original ambiguous word, the following principles should be followed.

1) Every EP should map to one and only one original ambiguous word.

2) The morphemes of an EP should map one by one to those of the original ambiguous word.

3) The sense of the EP should be the same as the corresponding ambiguous word, or has the maximum similarity with the word.

4) The morpheme of a pseudoword stands for a sense, while the sense should consist of one or more morphemes.

5) The morpheme should be a monosemous word.

The fourth principle above is the biggest difference between the EP and a general pseudo word. The sense of an EP is composed of one or several morphemes. This is a remarkable feature of the EP, which originates from its equivalent linguistic function with the original word. To construct the EP, it must be ensured that the sense of the EP maps to that of the original word. Usually, a candidate monosemous word for a morpheme stands for part of the linguistic function of the ambiguous word, thus we need to choose several morphemes to stand for one sense.

The relatedness of the senses refers to the similarity of the contexts of the original ambiguous word and its EP. The similarity between the words means that they serve as synonyms for each other. This principle demands that both semantic and pragmatic information should be taken into account in choosing a morpheme word.

3.4 Implementation of the EP-based Solution

An appropriate machine-readable dictionary is needed for construction of the EPs. A Chinese thesaurus is adopted and revised to meet this demand.

Extended Version of TongYiCiCiLin

To extend the TongYiCiCiLin (Cilin) to hold more words, several linguistic resources are adopted for manually adding new words. An extended version of the Cilin is achieved, which includes 77,343 items.

A hierarchy of three levels is organized in the extended Cilin for all items. Each node in the lowest level, called a minor class, contains several words of the same class. The words in one minor class are divided into several groups according to their sense similarity and relatedness, and each group is further divided into several lines, which can be viewed as the fifth level of the thesaurus. The 5-level hierarchy of the extended Cilin is shown in figure 1. The lower the level is, the more specific the sense is. The fifth level often contains a few words or only one word, which is called an atom word group, an atom class or an atom node. The words in the same atom node hold the smallest semantic distance.

From the root node to the leaf node, the sense is described more and more detailed, and the words in the same node are more and more related. Words in the same fifth level node have the same sense and linguistic function, which ensures that they can substitute for each other without leading to any change in the meaning of a sentence.

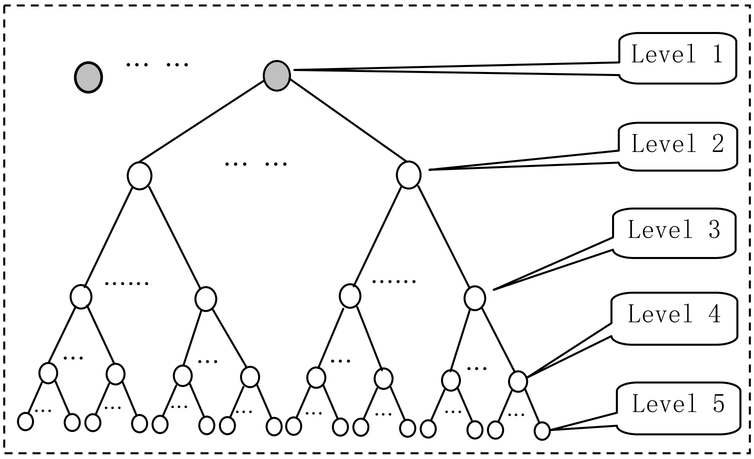


Figure 1. Organization of Cilin (extended)

The extended version of extended Cilin is freely downloadable from the Internet and has been used by over 20 organizations in the world[1].

Construction of EPs

According to the position of the ambiguous word, a proper word is selected as the morpheme of the EP. Almost every ambiguous word has its corresponding EP constructed in this way.

The first step is to decide the position of the ambiguous word starting from the leaf node of the tree structure. Words in the same leaf node are identical or similar in the linguistic function and word sense. Other words in the leaf node of the ambiguous word are called brother words of it. If there is a monosemous brother word, it can be taken as a candidate morpheme for the EP. If there does not exist such a brother word, trace to the fourth level. If there is still no monosemous brother word in the fourth level, trace to the third level. Because every node in the third level contains many words, candidate morpheme for the ambiguous can usually be found.

In most cases, candidate morphemes can be found at the fifth level. It is not often necessary to search to the fourth level, less to the third. According to our statistics, the extended Cilin contains about monosemous words for 93% of the ambiguous words in the fifth level, and 97% in the fourth level. There are only 112 ambiguous words left, which account for the other 3% and mainly are functional words. Some of the 3% words are rarely used, which cannot be found in even a large corpus. And words that lead to semantic misunderstanding are usually content words. In WSD research for English, only nouns, verbs, adjectives and adverbs are considered. From this aspect, the extended version of Cilin meets our demand for the construction of EPs.

If many monosemous brother words are found in the fourth or third level, there are many candidate morphemes to choose from. A further selection is made based on calculation of sense similarity. More similar brother words are chosen.

Computing of EPs

Generally, several morpheme words are needed for better construction of an EP. We assume that every morpheme word stands for a specific sense and does not influence each other. It is more complex to construct an EP than a common pseudo word, and the formulation and statistical information are also different.

An EP is described as follows:

$$\begin{array}{l}
\underline{W_{EP}\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}\\
\begin{array}{cccccc}
S_1: & W_{11}, & W_{12}, & W_{13}, & \cdots & W_{1k_1}\\
S_2: & W_{21}, & W_{22}, & W_{23}, & \cdots & W_{2k_2}\\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots\\
S_i: & W_{i1}, & W_{i2}, & W_{i3}, & \cdots & W_{ik_i}
\end{array}
\end{array}$$

Where W_{EP} is the EP word, S_i is a sense of the ambiguous word, and W_{ik} is a morpheme word of the EP.

The statistical information of the EP is calculated as follows:

1） $C(S_i)$ stands for the frequency of the S_i:

$$C(S_i) = \sum_k C(W_{ik})$$

2） $C(S_i, W_f)$ stands for the co-occurrence frequency of S_i and the contextual word W_f:

$$C(S_i, W_f) = \sum_k C(W_{ik}, W_f)$$

[1] It is located at http://www.ir-lab.org/.

Ambiguous word	citation (Qin and Wang, 2005)	Ours	Ambiguous word	citation (Qin and Wang, 2005)	Ours
把握(ba3 wo4)	0.56	0.87	没有(mei2 you3)	0.75	0.68
包(bao1)	0.59	0.75	起来(qi3 lai2)	0.82	0.54
材料(cai2 liao4)	0.67	0.79	钱(qian2)	0.75	0.62
冲击(chong1 ji1)	0.62	0.69	日子(ri4 zi3)	0.75	0.68
穿(chuan1)	0.80	0.57	少(shao3)	0.69	0.56
地方(di4 fang1)	0.65	0.65	突出(tu1 chu1)	0.82	0.86
分子(fen1 zi3)	0.91	0.81	研究(yan2 jiu1)	0.69	0.63
运动(yun4 dong4)	0.61	0.82	活动(huo2 dong4)	0.79	0.88
老(lao3)	0.59	0.50	走(zou3)	0.72	0.60
路(lu4)	0.74	0.64	坐(zuo4)	0.90	0.73
Average	0.72	0.69	Note: Average of the 20 words		

Table 2. The F-measure for the Supervised WSD

4 EP-based Unsupervised WSD Method

EP is a solution to the semantic knowledge acquisition problem, and it does not limit the choice of statistical learning methods. All of the mathematical modeling methods can be applied to EP-based WSD methods. This section focuses on the application of the EP concept to WSD, and chooses Bayesian method for the classifier construction.

4.1 A Sense Classifier Based on the Bayesian Model

Because the model acquires knowledge from the EPs but not from the original ambiguous word, the method introduced here does not need human tagging of training corpus.

In the training stage for WSD, statistics of EPs and context words are obtained and stored in a database. Senseval-3 data set plus unsupervised learning method are adopted to investigate into the value of EP in WSD. To ensure the comparability of experiment results, a Bayesian classifier is used in the experiments.

Bayesian Classifier

Although the Bayesian classifier is simple, it is quite efficient, and it shows good performance on WSD.

The Bayesian classifier used in this paper is described in (1)

$$S(w_i) = \operatorname{argmax}_{S_k}\left[\log P(S_k) + \sum_{v_j \in c_i} \log P(v_j \mid S_k)\right] \quad (1)$$

Where w_i is the ambiguous word, $P(S_k)$ is the occurrence probability of the sense S_k, $P(v_j \mid S_k)$ is the conditional probability of the context word v_j, and c_i is the set of the context words.

To simplify the experiment process, the Naive Bayesian modeling is adopted for the sense classifier. Feature selection and ensemble classification are not applied, which is both to simplify the calculation and to prove the effect of EPs in WSD.

Experiment Setup and Results

The Senseval-3 Chinese ambiguous words are taken as the testing set, which includes 20 words, each with 2-8 senses. The data for the ambiguous words are divided into a training set and a testing set by a ratio of 2:1. There are 15-20 training instances for each sense of the words, and occurs by the same frequency in the training and test set.

Supervised WSD is first implemented using the Bayesian model on the Senseval-3 data set. With a context window of (-10, +10), the open test results are shown in table 2.

The F-measure in table 2 is defined in (2).

$$F = \frac{2 \times P \times R}{P + R} \quad (2)$$

Where P and R refer to the precision and recall of the sense tagging respectively, which are calculated as shown in (3) and (4)

$$P = \frac{C(\text{correct})}{C(\text{tagged})} \quad (3)$$

$$R = \frac{C(\text{correct})}{C(\text{all})} \quad (4)$$

Where *C*(tagged) is the number of tagged instances of senses, *C*(correct) is the number of correct tags, and *C*(all) is the number of tags in the gold standard set. Every sense of the ambiguous word has a P value, a R value and a F value. The F value in table 2 is a weighted average of all the senses.

In the EP-based unsupervised WSD experiment, a 100M corpus (People's Daily for year 1998) is used for the EP training instances. The Senseval-3 data is used for the test. In our experiments, a context window of (-10, +10) is taken. The detailed results are shown in table 3.

4.2 Experiment Analysis and Discussion

Experiment Evaluation Method

Two evaluation criteria are used in the experiments, which are the F-measure and precision. Precision is a usual criterion in WSD performance analysis. Only in recent years, the precision, recall, and F-measure are all taken to evaluate the WSD performance.

In this paper, we will only show the f-measure score because it is a combined score of precision and recall.

Result Analysis on Bayesian Supervised WSD Experiment

The experiment results in table 2 reveals that the results of supervised WSD and those of (Qin and Wang, 2005) are different. Although they are all based on the Bayesian model, Qin and Wang (2005) used an ensemble classifier. However, the difference of the average value is not remarkable.

As introduced above, in the supervised WSD experiment, the various senses of the instances are evenly distributed. The lower bound as Gale et al. (1992c) suggested should be very low and it is more difficult to disambiguate if there are more senses. The experiment verifies this reasoning, because the highest F-measure is less than 90%, and the lowest is less than 60%, averaging about 70%.

With the same number of senses and the same scale of training data, there is a big difference between the WSD results. This shows that other factors exist which influence the performance other than the number of senses and training data size. For example, the discriminability among the senses is an important factor. The WSD task becomes more difficult if the senses of the ambiguous word are more similar to each other.

Experiment Analysis of the EP-based WSD

The EP-based unsupervised method takes the same open test set as the supervised method. The unsupervised method shows a better performance, with the highest F-measure score at 100%, lowest at 59% and average at 80%. The results shows that EP is useful in unsupervised WSD.

Sequence Number	Ambiguous word	F-measure	Sequence Number	Ambiguous word	F-measure (%)
1	把握(ba3 wo4)	0.93	11	没有(mei2 you3)	1.00
2	包(bao1)	0.74	12	起来(qi3 lai2)	0.59
3	料(cai2 liao4)	0.80	13	钱(qian2)	0.71
4	冲击(chong1 ji1)	0.85	14	日子(ri4 zi3)	0.62
5	穿(chuan1)	0.79	15	少(shao3)	0.82
6	地方(di4 fang1)	0.78	16	突出(tu1 chu1)	0.93
7	分子(fen1 zi3)	0.94	17	研究(yan2 jiu1)	0.71
8	运动(yun4 dong4)	0.94	18	活动(huo2 dong4)	0.89
9	老(lao3)	0.85	19	走(zou3)	0.68
10	路(lu4)	0.81	20	坐(zuo4)	0.67
Average	0.80	Note: Average of the 20 words			

Table 3. The Results for Unsupervised WSD based on EPs

From the results in table 2 and table 3, it can be seen that 16 among the 20 ambiguous words show better WSD performance in unsupervised SWD than in supervised WSD, while only 2 of them shows similar results and 2 performs worse . The average F-measure of the unsupervised method is higher by more than 10%. The reason lies in the following aspects:

1) Because there are several morpheme words for every sense of the word in construction of the EP, rich semantic information can be acquired in the training step and is an advantage for sense disambiguation.

2) Senseval-3 has provided a small-scale training set, with 15-20 training instances for each sense, which is not enough for the WSD modeling. The lack of training information leads to a low performance of the supervised methods.

3) With a large-scale training corpus, the unsupervised WSD method has got plenty of training instances for a high performance in disambiguation.

4) The discriminability of some ambiguous word may be low, but the corresponding EPs could be easier to disambiguate. For example, the ambiguous word "穿" has two senses which are difficult to distinguish from each other, but its Eps' senses of "越过/穿过/穿越" and "戳/捅/通/扎"can be easily disambiguated. It is the same for the word "冲击", whose Eps' senses are "撞击/磕碰/碰撞" and "损害/伤害". EP-based knowledge acquisition of these ambiguous words for WSD has helped a lot to achieve high performance.

5 Conclusion

As discussed above, the supervised WSD method shows a low performance because of its dependency on the size of the training data. This reveals its weakness in knowledge acquisition bottleneck. EP-based unsupervised method has overcame this weakness. It requires no manually tagged corpus to achieve a satisfactory performance on WSD. Experimental results show that EP-based method is a promising solution to the large-scale WSD task. In future work, we will examine the effectiveness of EP-based method in other WSD techniques.

References

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1991. Word-Sense Disambiguation Using Statistical Methods. In *Proc. of the 29th Annual Meeting of the Association for Computational Linguistics (ACL-1991)*, pages 264-270.

Mona Talat Diab. 2003. *Word Sense Disambiguation Within a Multilingual Framework*. PhD thesis, University of Maryland College Park.

Mona Diab. 2004a. Relieving the Data Acquisition Bottleneck in Word Sense Disambiguation. In *Proc. of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-2004),* pages 303-310.

Mona T. Diab. 2004b. An Unsupervised Approach for Bootstrapping Arabic Sense Tagging. In *Proc. of Arabic Script Based Languages Workshop at COLING 2004*, pages 43-50.

Mona Diab and Philip Resnik. 2002. An Unsupervised Method for Word Sense Tagging Using Parallel Corpora. In *Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 255-262.

William Gale, Kenneth Church, and David Yarowsky. 1992a. Using Bilingual Materials to Develop Word Sense Disambiguation Methods. In *Proc. of the 4th International Conference on Theoretical and Methodolgical Issues in Machine Translation(TMI-92)*, pages 101-112.

William Gale, Kenneth Church, and David Yarowsky. 1992b. Work on Statistical Methods for Word Sense Disambiguation. In *Proc. of AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 54-60.

William Gale, Kenneth Ward Church, and David Yarowsky. 1992c. Estimating Upper and Lower Bounds on the Performance of Word Sense Disambiguation Programs. In *Proc. of the 30th Annual Meeting of the Association for Computational Linguistics (ACL-1992)*, pages 249-256.

Tanja Gaustad. 2001. Statistical Corpus-Based Word Sense Disambiguation: Pseudowords vs. Real Ambiguous Words. In *Proc. of the 39th ACL/EACL, Student Research Workshop,* pages 61-66.

Nancy Ide, Tomaz Erjavec, and Dan Tufiş. 2001. Automatic Sense Tagging Using Parallel Corpora. In *Proc. of the Sixth Natural Language Processing Pacific Rim Symposium,* pages 83-89.

Nancy Ide, Tomaz Erjavec, and Dan Tufis. 2002. Sense Discrimination with Parallel Corpora. In *Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, pages 54-60.

Cong Li and Hang Li. 2002. Word Translation Disambiguation Using Bilingual Bootstrapping. In *Proc. of the 40th Annual Meeting of the Association* *for Computational Linguistics (ACL-2002),* pages 343-351.

Rada Mihalcea and Dan Moldovan. 2000. An Iterative Approach to Word Sense Disambiguation. In *Proc. of Florida Artificial Intelligence Research Society Conference (FLAIRS 2000)*, pages 219-223.

Rada F. Mihalcea. 2002. Bootstrapping Large Sense Tagged Corpora. In *Proc. of the 3rd International Conference on Languages Resources and Evaluations (LREC 2002)*, pages 1407-1411.

Preslav I. Nakov and Marti A. Hearst. 2003. Category-based Pseudowords. In *Companion Volume to the Proceedings of HLT-NAACL 2003, Short Papers,* pages 67-69.

Hwee Tou. Ng, Bin Wang, and Yee Seng Chan. 2003. Exploiting Parallel Texts for Word Sense Disambiguation: An Empirical Study. In *Proc. of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, pages 455-462.

Zheng-Yu Niu, Dong-Hong Ji, and Chew-Lim Tan. 2005. Word Sense Disambiguation Using Label Propagation Based Semi-Supervised Learning. In *Proc. of the 43th Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, pages 395-402.

Ying Qin and Xiaojie Wang. 2005. A Track-based Method on Chinese WSD. In *Proc. of Joint Symposium of Computational Linguistics of China (JSCL-2005)*, pages 127-133.

Hinrich. Schutze. 1998. Automatic Word Sense Discrimination. *Computational Linguistics,* 24(1): 97-123.

David Yarowsky. 1994. Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French. In *Proc. of the 32nd Annual Meeting of the Association for Computational Linguistics(ACL-1994)*, pages 88-95.

David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-1995)*, pages 189-196.

Improving the Scalability of Semi-Markov Conditional Random Fields for Named Entity Recognition

Daisuke Okanohara† Yusuke Miyao† Yoshimasa Tsuruoka ‡ Jun'ichi Tsujii†‡§
†Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
‡School of Informatics, University of Manchester
POBox 88, Sackville St, MANCHESTER M60 1QD, UK
§SORST, Solution Oriented Research for Science and Technology
Honcho 4-1-8, Kawaguchi-shi, Saitama, Japan
{hillbig,yusuke,tsuruoka,tsujii}@is.s.u-tokyo.ac.jp

Abstract

This paper presents techniques to apply semi-CRFs to Named Entity Recognition tasks with a tractable computational cost. Our framework can handle an NER task that has long named entities and many labels which increase the computational cost. To reduce the computational cost, we propose two techniques: the first is the use of feature forests, which enables us to pack feature-equivalent states, and the second is the introduction of a filtering process which significantly reduces the number of candidate states. This framework allows us to use a rich set of features extracted from the chunk-based representation that can capture informative characteristics of entities. We also introduce a simple trick to transfer information about distant entities by embedding label information into non-entity labels. Experimental results show that our model achieves an F-score of 71.48% on the JNLPBA 2004 shared task without using any external resources or post-processing techniques.

1 Introduction

The rapid increase of information in the biomedical domain has emphasized the need for automated information extraction techniques. In this paper we focus on the Named Entity Recognition (NER) task, which is the first step in tackling more complex tasks such as relation extraction and knowledge mining.

Biomedical NER (Bio-NER) tasks are, in general, more difficult than ones in the news domain. For example, the best F-score in the shared task of Bio-NER in COLING 2004 JNLPBA (Kim et al., 2004) was 72.55% (Zhou and Su, 2004) [1], whereas the best performance at MUC-6, in which systems tried to identify general named entities such as person or organization names, was an accuracy of 95% (Sundheim, 1995).

Many of the previous studies of Bio-NER tasks have been based on machine learning techniques including Hidden Markov Models (HMMs) (Bikel et al., 1997), the dictionary HMM model (Kou et al., 2005) and Maximum Entropy Markov Models (MEMMs) (Finkel et al., 2004). Among these methods, conditional random fields (CRFs) (Lafferty et al., 2001) have achieved good results (Kim et al., 2005; Settles, 2004), presumably because they are free from the so-called label bias problem by using a global normalization.

Sarawagi and Cohen (2004) have recently introduced semi-Markov conditional random fields (semi-CRFs). They are defined on semi-Markov chains and attach labels to the subsequences of a sentence, rather than to the tokens[2]. The semi-Markov formulation allows one to easily construct entity-level features. Since the features can capture all the characteristics of a subsequence, we can use, for example, a dictionary feature which measures the similarity between a candidate segment and the closest element in the dictionary. Kou et al. (2005) have recently showed that semi-CRFs perform better than CRFs in the task of recognition of protein entities.

The main difficulty of applying semi-CRFs to Bio-NER lies in the computational cost at training

[1] Krauthammer (2004) reported that the inter-annotator agreement rate of human experts was 77.6% for bio-NLP, which suggests that the upper bound of the F-score in a Bio-NER task may be around 80%.

[2] Assuming that non-entity words are placed in unit-length segments.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 465–472,
Sydney, July 2006. ©2006 Association for Computational Linguistics

Table 1: Length distribution of entities in the training set of the shared task in 2004 JNLPBA

Length	# entity	Ratio
1	21646	42.19
2	15442	30.10
3	7530	14.68
4	3505	6.83
5	1379	2.69
6	732	1.43
7	409	0.80
8	252	0.49
>8	406	0.79
total	51301	100.00

because the number of named entity classes tends to be large, and the training data typically contain many long entities, which makes it difficult to enumerate all the entity candidates in training. Table 1 shows the length distribution of entities in the training set of the shared task in 2004 JNLPBA. Formally, the computational cost of training semi-CRFs is $O(KLN)$, where L is the upper bound length of entities, N is the length of sentence and K is the size of label set. And that of training in first order semi-CRFs is $O(K^2LN)$. The increase of the cost is used to transfer non-adjacent entity information.

To improve the scalability of semi-CRFs, we propose two techniques: the first is to introduce a filtering process that significantly reduces the number of candidate entities by using a "lightweight" classifier, and the second is to use *feature forest* (Miyao and Tsujii, 2002), with which we pack the feature equivalent states. These enable us to construct semi-CRF models for the tasks where entity names may be long and many class-labels exist at the same time. We also present an extended version of semi-CRFs in which we can make use of information about a preceding named entity in defining features within the framework of first order semi-CRFs. Since the preceding entity is not necessarily adjacent to the current entity, we achieve this by embedding the information on preceding labels for named entities into the labels for non-named entities.

2 CRFs and Semi-CRFs

CRFs are undirected graphical models that encode a conditional probability distribution using a given set of features. CRFs allow both discriminative training and bi-directional flow of probabilistic information along the sequence. In NER, we often use linear-chain CRFs, which define the conditional probability of a state sequence **y** = y_1, ..., y_n given the observed sequence **x** = x_1,...,x_n by:

$$p(\mathbf{y}|\mathbf{x}, \lambda) = \frac{1}{Z(\mathbf{x})} \exp(\Sigma_{i=1}^{n} \Sigma_j \lambda_j f_j(y_{i-1}, y_i, \mathbf{x}, i)), \tag{1}$$

where $f_j(y_{i-1}, y_i, \mathbf{x}, i)$ is a feature function and $Z(\mathbf{x})$ is the normalization factor over all the state sequences for the sequence **x**. The model parameters are a set of real-valued weights $\lambda = \{\lambda_j\}$, each of which represents the weight of a feature. All the feature functions are real-valued and can use adjacent label information.

Semi-CRFs are actually a restricted version of order-L CRFs in which all the labels in a chunk are the same. We follow the definitions in (Sarawagi and Cohen, 2004). Let **s** = $\langle s_1, ..., s_p \rangle$ denote a segmentation of x, where a segment s_j = $\langle t_j$, u_j, $y_j \rangle$ consists of a start position t_j, an end position u_j, and a label y_j. We assume that segments have a positive length bounded above by the pre-defined upper bound L ($t_j \leq u_j$, $u_j - t_j + 1 \leq L$) and completely cover the sequence **x** without overlapping, that is, **s** satisfies $t_1 = 1$, $u_p = |\mathbf{x}|$, and $t_{j+1} = u_j + 1$ for $j = 1, ..., p-1$. Semi-CRFs define a conditional probability of a state sequence **y** given an observed sequence **x** by:

$$p(\mathbf{y}|\mathbf{x}, \lambda) = \frac{1}{Z(\mathbf{x})} \exp(\Sigma_j \Sigma_i \lambda_i f_i(s_j)), \tag{2}$$

where $f_i(s_j) := f_i(y_{j-1}, y_j, \mathbf{x}, t_j, u_j)$ is a feature function and $Z(\mathbf{x})$ is the normalization factor as defined for CRFs. The inference problem for semi-CRFs can be solved by using a semi-Markov analog of the usual Viterbi algorithm. The computational cost for semi-CRFs is $O(KLN)$ where L is the upper bound length of entities, N is the length of sentence and K is the number of label set. If we use previous label information, the cost becomes $O(K^2LN)$.

3 Using Non-Local Information in Semi-CRFs

In conventional CRFs and semi-CRFs, one can only use the information on the adjacent previous label when defining the features on a certain state or entity. In NER tasks, however, information about a distant entity is often more useful than

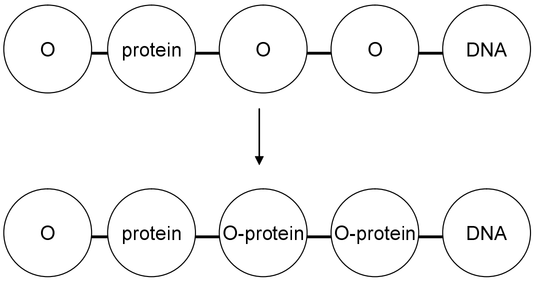


Figure 1: Modification of "**O**" (other labels) to transfer information on a preceding named entity.

information about the previous state (Finkel et al., 2005). For example, consider the sentence "... *including Sp1 and CP1.*" where the correct labels of "*Sp1*" and "*CP1*" are both "protein". It would be useful if the model could utilize the (non-adjacent) information about "*Sp1*" being "protein" to classify "*CP1*" as "protein". On the other hand, information about adjacent labels does not necessarily provide useful information because, in many cases, the previous label of a named entity is "O", which indicates a non-named entity. For 98.0% of the named entities in the training data of the shared task in the 2004 JNLPBA, the label of the preceding entity was "O".

In order to incorporate such non-local information into semi-CRFs, we take a simple approach. We divide the label of "O" into "O-protein" and "O" so that they convey the information on the preceding named entity. Figure 1 shows an example of this conversion, in which the two labels for the third and fourth states are converted from "O" to "O-protein". When we define the features for the fifth state, we can use the information on the preceding entity "protein" by looking at the fourth state. Since this modification changes only the label set, we can do this within the framework of semi-CRF models. This idea is originally proposed in (Peshkin and Pfeffer, 2003). However, they used a dynamic Bayesian network (DBNs) rather than a semi-CRF, and semi-CRFs are likely to have significantly better performance than DBNs.

In previous work, such non-local information has usually been employed at a post-processing stage. This is because the use of long distance dependency violates the locality of the model and prevents us from using dynamic programming techniques in training and inference. Skip-CRFs (Sutton and McCallum, 2004) are a direct implementation of long distance effects to the model. However, they need to determine the structure for propagating non-local information in advance. In a recent study by Finkel et al., (2005), non-local information is encoded using an independence model, and the inference is performed by Gibbs sampling, which enables us to use a state-of-the-art factored model and carry out training efficiently, but inference still incurs a considerable computational cost. Since our model handles limited type of non-local information, i.e. the label of the preceding entity, the model can be solved without approximation.

4 Reduction of Training/Inference Cost

The straightforward implementation of this modeling in semi-CRFs often results in a prohibitive computational cost.

In biomedical documents, there are quite a few entity names which consist of many words (names of 8 words in length are not rare). This makes it difficult for us to use semi-CRFs for biomedical NER, because we have to set L to be eight or larger, where L is the upper bound of the length of possible chunks in semi-CRFs. Moreover, in order to take into account the dependency between named entities of different classes appearing in a sentence, we need to incorporate multiple labels into a single probabilistic model. For example, in the shared task in COLING 2004 JNLPBA (Kim et al., 2004) the number of labels is six ("protein", "DNA", "RNA", "cell_line", "cell_type" and "other"). This also increases the computational cost of a semi-CRF model.

To reduce the computational cost, we propose two methods (see Figure 2). The first is employing a filtering process using a lightweight classifier to remove unnecessary state candidates beforehand (Figure 2 (2)), and the second is the using the *feature forest model* (Miyao and Tsujii, 2002) (Figure 2 (3)), which employs dynamic programming at training "*as much as possible*".

4.1 Filtering with a naive Bayes classifier

We introduce a filtering process to remove low probability candidate states. This is the first step of our NER system. After this filtering step, we construct semi-CRFs on the remaining candidate states using a feature forest. Therefore the aim of this filtering is to reduce the number of candidate states, without removing correct entities. This idea

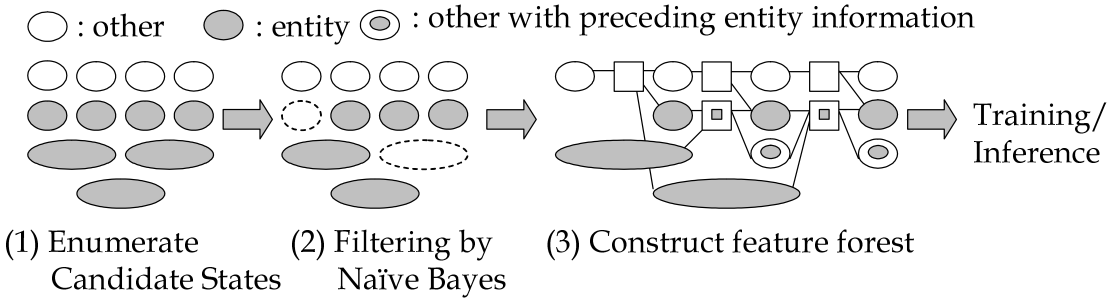

: other
: entity
: other with preceding entity information
Training/ Inference
(1) Enumerate Candidate States
(2) Filtering by Naïve Bayes
(3) Construct feature forest

Figure 2: The framework of our system. We first enumerate all possible candidate states, and then filter out low probability states by using a light-weight classifier, and represent them by using feature forest.

Table 2: Features used in the naive Bayes Classifier for the entity candidate: w_s, w_{s+1}, ..., w_e. sp_i is the result of shallow parsing at w_i.

Feature Name	Example of Features
Start/End Word	w_s, w_e
Inside Word	w_s, w_{s+1}, ... , w_e
Context Word	w_{s-1}, w_{e+1}
Start/End SP	sp_s, sp_e
Inside SP	sp_s, sp_{s+1}, ..., sp_e
Context SP	sp_{s-1}, sp_{e+1}

is similar to the method proposed by Tsuruoka and Tsujii (2005) for chunk parsing, in which implausible phrase candidates are removed beforehand.

We construct a binary naive Bayes classifier using the same training data as those for semi-CRFs. In training and inference, we enumerate all possible chunks (the max length of a chunk is L as for semi-CRFs) and then classify those into "entity" or "other". Table 2 lists the features used in the naive Bayes classifier. This process can be performed independently of semi-CRFs

Since the purpose of the filtering is to reduce the computational cost, rather than to achieve a good F-score by itself, we chose the threshold probability of filtering so that the recall of filtering results would be near 100 %.

4.2 Feature Forest

In estimating semi-CRFs, we can use an efficient dynamic programming algorithm, which is similar to the forward-backward algorithm (Sarawagi and Cohen, 2004). The proposal here is a more general framework for estimating sequential conditional random fields.

This framework is based on *the feature forest model*, which was originally proposed for disambiguation models for parsing (Miyao and Tsujii, 2002). A feature forest model is a maximum entropy model defined over *feature forests*, which are abstract representations of an exponential number of sequence/tree structures. A feature forest is an "and/or" graph: in Figure 3, circles represent

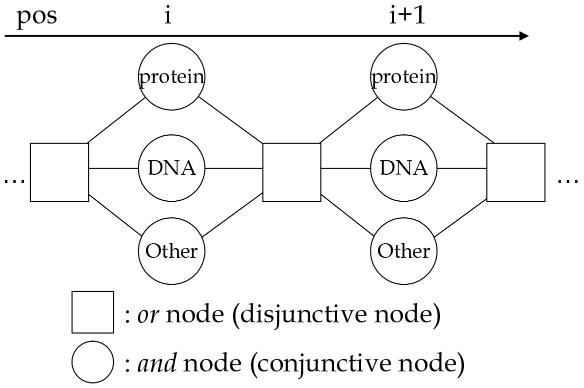


Figure 3: Example of feature forest representation of linear chain CRFs. Feature functions are assigned to "and" nodes.

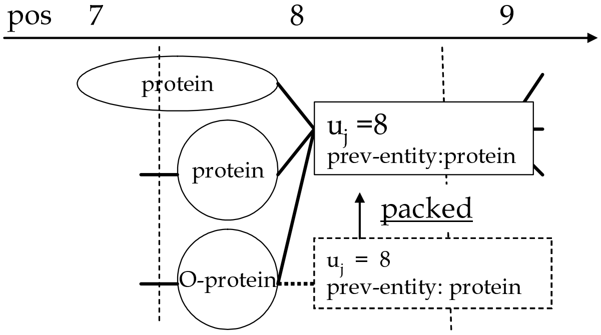


Figure 4: Example of packed representation of semi-CRFs. The states that have the same end position and prev-entity label are packed.

"and" nodes (*conjunctive* nodes), while boxes denote "or" nodes (*disjunctive* nodes). Feature functions are assigned to "and" nodes. We can use the information of the previous "and" node for designing the feature functions through the previous "or" node. Each sequence in a feature forest is obtained by choosing a conjunctive node for each disjunctive node. For example, Figure 3 represents $3 \times 3 = 9$ sequences, since each disjunctive node has three candidates. It should be noted that feature forests can represent an exponential number of sequences with a polynomial number of conjunctive/disjunctive nodes.

One can estimate a maximum entropy model for the whole sequence with dynamic programming by representing the probabilistic events, i.e. sequence of named entity tags, by feature forests (Miyao and Tsujii, 2002).

In the previous work (Lafferty et al., 2001; Sarawagi and Cohen, 2004), "or" nodes are considered implicitly in the dynamic programming framework. In feature forest models, "or" nodes are packed when they have same conditions. For example, "or" nodes are packed when they have same end positions and same labels in the first order semi-CRFs,

In general, we can pack different "or" nodes that yield equivalent feature functions in the following nodes. In other words, "or" nodes are packed when the following states use partial information on the preceding states. Consider the task of tagging *entity* and *O-entity*, where the latter tag is actually *O* tags that distinguish the preceding named entity tags. When we simply apply first-order semi-CRFs, we must distinguish states that have different previous states. However, when we want to distinguish only the preceding named entity tags rather than the immediate previous states, feature forests can represent these events more compactly (Figure 4). We can implement this as follows. In each "or" node, we generate the following "and" nodes and their feature functions. Then we check whether there exist "or" node which has same conditions by using its information about "end position" and "previous entity". If so, we connect the "and" node to the corresponding "or" node. If not, we generate a new "or" node and continue the process.

Since the states with label *O-entity* and *entity* are packed, the computational cost of training in our model (First order semi-CRFs) becomes the half of the original one.

5 Experiments

5.1 Experimental Setting

Our experiments were performed on the training and evaluation set provided by the shared task in COLING 2004 JNLPBA (Kim et al., 2004). The training data used in this shared task came from the GENIA version 3.02 corpus. In the task there are five semantic labels: protein, DNA, RNA, cell_line and cell_type. The training set consists of 2000 abstracts from MEDLINE, and the evaluation set consists of 404 abstracts. We divided the original training set into 1800 abstracts and 200 abstracts, and the former was used as the training data and the latter as the development data. For semi-CRFs, we used *amis*[3] for training the semi-CRF with feature-forest. We used *GENIA taggar*[4] for POS-tagging and shallow parsing.

We set $L = 10$ for training and evaluation when we do not state L explicitly , where L is the upper bound of the length of possible chunks in semi-CRFs.

5.2 Features

Table 3 lists the features used in our semi-CRFs. We describe the chunk-dependent features in detail, which cannot be encoded in token-level features.

"**Whole chunk**" is the normalized names attached to a chunk, which performs like the closed dictionary. "**Length**" and "**Length and End-Word**" capture the tendency of the length of a named entity. "**Count feature**" captures the tendency for named entities to appear repeatedly in the same sentence.

"**Preceding Entity and Prev Word**" are features that capture specifically words for conjunctions such as "*and*" or ", (comma)", e.g., for the phrase "*OCIM1 and K562*", both "*OCIM1*" and "*K562*" are assigned cell_line labels. Even if the model can determine only that "*OCIM1*" is a cell_line , this feature helps "*K562*" to be assigned the label cell_line.

5.3 Results

We first evaluated the filtering performance. Table 4 shows the result of the filtering on the training

[3] http://www-tsujii.is.s.u-tokyo.ac.jp/amis/

[4] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/tagger/
Note that the evaluation data are not used for training the GENIA tagger.

Table 3: Feature templates used for the chunk **s** := w_s w_{s+1} ... w_e where w_s and w_e represent the words at the beginning and ending of the target chunk respectively. p_i is the part of speech tag of w_i and sc_i is the shallow parse result of w_i.

Feature Name	description of features
Non-Chunk Features	
Word/POS/SC with Position	BEGIN + w_s, END + w_e, IN + w_{s+1}, ..., IN + w_{e-1}, BEGIN + p_s,...
Context Uni-gram/Bi-gram	w_{s-1}, w_{e+1}, w_{s-2} + w_{s-1}, w_{e+1} + w_{e+2}, w_{s-1} + w_{e+1}
Prefix/Suffix of Chunk	2/3-gram character prefix of w_s, 2/3/4-gram character suffix of w_e
Orthography	capitalization and word formation of $w_s...w_e$
Chunk Features	
Whole chunk	w_s + w_{s+1} + ... + w_e
Word/POS/SC End Bi-grams	w_{e-1} + w_e, $p_{e-1} + p_e$, sc_{e-1} + sc_e
Length, Length and End Word	|**s**|, |**s**|+w_e
Count Feature	the frequency of $w_s w_{s+1}..w_e$ in a sentence is greater than one
Preceding Entity Features	
Preceding Entity /and Prev Word	$PrevState$, $PrevState$ + w_{s-1}

Table 4: Filtering results using the naive Bayes classifier. The number of entity candidates for the training set was 4179662, and that of the development set was 418628.

Training set		
Threshold probability	reduction ratio	recall
1.0×10^{-12}	0.14	0.984
1.0×10^{-15}	0.20	0.993
Development set		
Threshold probability	reduction ratio	recall
1.0×10^{-12}	0.14	0.985
1.0×10^{-15}	0.20	0.994

and evaluation data. The naive Bayes classifiers effectively reduced the number of candidate states with very few falsely removed correct entities.

We then examined the effect of filtering on the final performance. In this experiment, we could not examine the performance without filtering using all the training data, because training on all the training data without filtering required much larger memory resources (estimated to be about 80G Byte) than was possible for our experimental setup. We thus compared the result of the recognizers with and without filtering using only 2000 sentences as the training data. Table 5 shows the result of the total system with different filtering thresholds. The result indicates that the filtering method achieved very well without decreasing the overall performance.

We next evaluate the effect of filtering, chunk information and non-local information on final performance. Table 6 shows the performance result for the recognition task. L means the upper bound of the length of possible chunks in semi-CRFs. We note that we cannot examine the result of $L = 10$ without filtering because of the intractable computational cost. The row "w/o Chunk Feature" shows the result of the system which does not employ Chunk-Features in Table 3 at training and inference. The row "Preceding Entity" shows the result of a system which uses **Preceding Entity** and **Preceding Entity and Prev Word** features. The results indicate that the chunk features contributed to the performance, and the filtering process enables us to use full chunk representation ($L = 10$). The results of McNemar's test suggest that the system with chunk features is significantly better than the system without it (the p-value is less than $1.0 < 10^{-4}$). The result of the preceding entity information improves the performance. On the other hand, the system with preceding information is not significantly better than the system without it[5]. Other non-local information may improve performance with our framework and this is a topic for future work.

Table 7 shows the result of the overall performance in our best setting, which uses the information about the preceding entity and 1.0×10^{-15} threshold probability for filtering. We note that the result of our system is similar to those of other sys-

[5] The result of the classifier on development data is 74.64 (without preceding information) and 75.14 (with preceding information).

Table 5: Performance with filtering on the development data. ($< 1.0 \times 10^{-12}$) means the threshold probability of the filtering is 1.0×10^{-12}.

	Recall	Precision	F-score	Memory Usage (MB)	Training Time (s)
Small Training Data = 2000 sentences					
Without filtering	65.77	72.80	69.10	4238	7463
Filtering ($< 1.0 \times 10.0^{-12}$)	64.22	70.62	67.27	600	1080
Filtering ($< 1.0 \times 10.0^{-15}$)	65.34	72.52	68.74	870	2154
All Training Data = 16713 sentences					
Without filtering	Not available			Not available	
Filtering ($< 1.0 \times 10.0^{-12}$)	70.05	76.06	72.93	10444	14661
Filtering ($< 1.0 \times 10.0^{-15}$)	**72.09**	**78.47**	**75.14**	15257	31636

Table 6: Overall performance on the evaluation set. L is the upper bound of the length of possible chunks in semi-CRFs.

	Recall	Precision	F-score
$L < 5$	64.33	65.51	64.92
$L = 10$ + Filtering ($< 1.0 \times 10.0^{-12}$)	70.87	68.33	69.58
$L = 10$ + Filtering ($< 1.0 \times 10.0^{-15}$)	72.59	70.16	71.36
w/o Chunk Feature	70.53	69.92	70.22
+ Preceding Entity	**72.65**	**70.35**	**71.48**

tems in several respects, that is, the performance of cell_line is not good, and the performance of the right boundary identification (78.91% in F-score) is better than that of the left boundary identification (75.19% in F-score).

Table 8 shows a comparison between our system and other state-of-the-art systems. Our system has achieved a comparable performance to these systems and would be still improved by using external resources or conducting pre/post processing. For example, Zhou et. al (2004) used post processing, abbreviation resolution and external dictionary, and reported that they improved F-score by 3.1%, 2.1% and 1.2% respectively. Kim et. al (2005) used the original GENIA corpus to employ the information about other semantic classes for identifying term boundaries. Finkel et. al (2004) used gazetteers, web-querying, surrounding abstracts, and frequency counts from the BNC corpus. Settles (2004) used semantic domain knowledge of 17 types of lexicon. Since our approach and the use of external resources/knowledge do not conflict but are complementary, examining the combination of those techniques should be an interesting research topic.

Table 7: Performance of our system on the evaluation set

Class	Recall	Precision	F-score
protein	77.74	68.92	73.07
DNA	69.03	70.16	69.59
RNA	69.49	67.21	68.33
cell_type	65.33	82.19	72.80
cell_line	57.60	53.14	55.28
overall	72.65	70.35	71.48

Table 8: Comparison with other systems

System	Recall	Precision	F-score
Zhou et. al (2004)	75.99	69.42	72.55
Our system	72.65	70.35	71.48
Kim et.al (2005)	72.77	69.68	71.19
Finkel et. al (2004)	68.56	71.62	70.06
Settles (2004)	70.3	69.3	69.8

6 Conclusion

In this paper, we have proposed a single probabilistic model that can capture important characteristics of biomedical named entities. To overcome the prohibitive computational cost, we have presented an efficient training framework and a filtering method which enabled us to apply first order semi-CRF models to sentences having many labels and entities with long names. Our results showed that our filtering method works very well without decreasing the overall performance. Our system achieved an F-score of 71.48% without the use of gazetteers, post-processing or external resources. The performance of our system came close to that of the current best performing system which makes extensive use of external resources and rule based post-processing.

The contribution of the non-local information introduced by our method was not significant in the experiments. However, other types of non-local information have also been shown to be effective (Finkel et al., 2005) and we will examine the effectiveness of other non-local information which can be embedded into label information.

As the next stage of our research, we hope to apply our method to shallow parsing, in which segments tend to be long and non-local information is important.

References

Daniel M. Bikel, Richard Schwartz, and Ralph Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Proc. of the Fifth Conference on Applied Natural Language Processing*.

Jenny Finkel, Shipra Dingare, Huy Nguyen, Malvina Nissim, Gail Sinclair, and Christopher Manning. 2004. Exploiting context for biomedical entity recognition: From syntax to the web. In *Proc. of JNLPBA-04*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proc. of ACL 2005*, pages 363–370.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proc. of JNLPBA-04*, pages 70–75.

Seonho Kim, Juntae Yoon, Kyung-Mi Park, and Hae-Chang Rim. 2005. Two-phase biomedical named entity recognition using a hybrid method. In *Proc. of the Second International Joint Conference on Natural Language Processing (IJCNLP-05)*.

Zhenzhen Kou, William W. Cohen, and Robert F. Murphy. 2005. High-recall protein entity recognition using a dictionary. *Bioinformatics 2005 21*.

Micahel Krauthammer and Goran Nenadic. 2004. Term identification in the biomedical literature. *Jornal of Biomedical Informatics*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML 2001*.

Yusuke Miyao and Jun'ichi Tsujii. 2002. Maximum entropy estimation for feature forests. In *Proc. of HLT 2002*.

Peshkin and Pfeffer. 2003. Bayesian information extraction network. In *IJCAI*.

Sunita Sarawagi and William W. Cohen. 2004. Semi-markov conditional random fields for information extraction. In *NIPS 2004*.

Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proc. of JNLPBA-04*.

Beth M. Sundheim. 1995. Overview of results of the MUC-6 evaluation. In *Sixth Message Understanding Conference (MUC-6)*, pages 13–32.

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *ICML workshop on Statistical Relational Learning*.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2005. Chunk parsing revisited. In *Proceedings of the 9th International Workshop on Parsing Technologies (IWPT 2005)*.

GuoDong Zhou and Jian Su. 2004. Exploring deep knowledge resources in biomedical name recognition. In *Proc. of JNLPBA-04*.

Factorizing Complex Models: A Case Study in Mention Detection

Radu Florian, Hongyan Jing, Nanda Kambhatla and **Imed Zitouni**
IBM TJ Watson Research Center
Yorktown Heights, NY 10598
{raduf,hjing,nanda,izitouni}@us.ibm.com

Abstract

As natural language understanding research advances towards deeper knowledge modeling, the tasks become more and more complex: we are interested in more nuanced word characteristics, more linguistic properties, deeper semantic and syntactic features. One such example, explored in this article, is the mention detection and recognition task in the Automatic Content Extraction project, with the goal of identifying named, nominal or pronominal references to real-world entities—mentions—and labeling them with three types of information: entity type, entity subtype and mention type. In this article, we investigate three methods of assigning these related tags and compare them on several data sets. A system based on the methods presented in this article participated and ranked very competitively in the ACE'04 evaluation.

1 Introduction

Information extraction is a crucial step toward understanding and processing natural language data, its goal being to identify and categorize important information conveyed in a discourse. Examples of information extraction tasks are identification of the actors and the objects in written text, the detection and classification of the relations among them, and the events they participate in. These tasks have applications in, among other fields, summarization, information retrieval, data mining, question answering, and language understanding.

One of the basic tasks of information extraction is the *mention detection* task. This task is very similar to *named entity recognition* (NER), as the objects of interest represent very similar concepts. The main difference is that the latter will identify, however, only *named* references, while mention detection seeks named, nominal and pronominal references. In this paper, we will call the identified references *mentions* – using the ACE (NIST, 2003) nomenclature – to differentiate them from *entities* which are the real-world objects (the actual person, location, etc) to which the mentions are referring to[1].

Historically, the goal of the NER task was to find named references to entities and quantity references – time, money (MUC-6, 1995; MUC-7, 1997). In recent years, Automatic Content Extraction evaluation (NIST, 2003; NIST, 2004) expanded the task to also identify nominal and pronominal references, and to group the mentions into sets referring to the same entity, making the task more complicated, as it requires a co-reference module. The set of identified properties has also been extended to include the *mention type* of a reference (whether it is named, nominal or pronominal), its *subtype* (a more specific type dependent on the main entity type), and its *genericity* (whether the entity points to a specific entity, or a generic one[2]), besides the customary main entity type. To our knowledge, little research has been done in the natural language processing context or otherwise on investigating the specific problem of how such multiple labels are best assigned. This article compares three methods for such an assignment.

The simplest model which can be considered for the task is to create an atomic tag by "gluing" together the sub-task labels and considering the new label atomic. This method transforms the problem into a regular sequence classification task, similar to part-of-speech tagging, text chunking, and named entity recognition tasks. We call this model the *all-in-one* model. The immediate drawback of this model is that it creates a large classification space (the cross-product of the sub-task classification spaces) and that, during decoding, partially similar classifications will compete instead of cooperate - more details are presented in Section 3.1. Despite (or maybe due to) its relative simplicity, this model obtained good results in several instances in the past, for POS tagging in morphologically rich languages (Hajic and Hladká, 1998)

[1] In a pragmatic sense, entities are sets of mentions which co-refer.

[2] This last attribute, genericity, depends only loosely on local context. As such, it should be assigned while examining all mentions in an entity, and for this reason is beyond the scope of this article.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 473–480,
Sydney, July 2006. ©2006 Association for Computational Linguistics

and mention detection (Jing et al., 2003; Florian et al., 2004).

At the opposite end of classification methodology space, one can use a *cascade* model, which performs the sub-tasks sequentially in a predefined order. Under such a model, described in Section 3.3, the user will build separate models for each subtask. For instance, it could first identify the mention boundaries, then assign the entity type, subtype, and mention level information. Such a model has the immediate advantage of having smaller classification spaces, with the drawback that it requires a specific model invocation path.

In between the two extremes, one can use a *joint* model, which models the classification space in the same way as the all-in-one model, but where the classifications are not atomic. This system incorporates information about sub-model parts, such as whether the current word starts an entity (of any type), or whether the word is part of a nominal mention.

The paper presents a novel contrastive analysis of these three models, comparing them on several datasets in three languages selected from the ACE 2003 and 2004 evaluations. The methods described here are independent of the underlying classifiers, and can be used with any sequence classifiers. All experiments in this article use our in-house implementation of a maximum entropy classifier (Florian et al., 2004), which we selected because of its flexibility of integrating arbitrary types of features. While we agree that the particular choice of classifier will undoubtedly introduce some classifier bias, we want to point out that the described procedures have more to do with the organization of the search space, and will have an impact, one way or another, on most sequence classifiers, including conditional random field classifiers.[3]

The paper is organized as follows: Section 2 describes the multi-task classification problem and prior work, Section 3.3 presents and contrasts the three meta-classification models. Section 4 outlines the experimental setup and the obtained results, and Section 5 concludes the paper.

2 Multi-Task Classification

Many tasks in Natural Language Processing involve labeling a word or sequence of words with a specific property; classic examples are part-of-speech tagging, text chunking, word sense disambiguation and sentiment classification. Most of the time, the word labels are atomic labels, containing a very specific piece of information (e.g. the word is noun plural, or starts a noun phrase, etc). There are cases, though, where the labels consist of several related, but not entirely correlated, properties; examples include mention detection—the task we are interested in—, syntactic parsing with functional tag assignment (besides identifying the syntactic parse, also label the constituent nodes with their functional category, as defined in the Penn Treebank (Marcus et al., 1993)), and, to a lesser extent, part-of-speech tagging in highly inflected languages.[4]

The particular type of mention detection that we are examining in this paper follows the ACE general definition: each mention in the text (a reference to a real-world entity) is assigned three types of information:[5]

- An entity type, describing the type of the entity it points to (e.g. person, location, organization, etc)
- An entity subtype, further detailing the type (e.g. organizations can be commercial, governmental and non-profit, while locations can be a nation, population center, or an international region)
- A mention type, specifying the way the entity is realized – a mention can be named (e.g. *John Smith*), nominal (e.g. *professor*), or pronominal (e.g. *she*).

Such a problem – where the classification consists of several subtasks or attributes – presents additional challenges, when compared to a standard sequence classification task. Specifically, there are inter-dependencies between the subtasks that need to be modeled explicitly; predicting the tags independently of each other will likely result in inconsistent classifications. For instance, in our running example of mention detection, the subtype task is dependent on the entity type; one could not have a *person* with the subtype *non-profit*. On the other hand, the mention type is relatively independent of the entity type and/or subtype: each entity type could be realized under any mention type and vice-versa.

The multi-task classification problem has been subject to investigation in the past. Caruana et al. (1997) analyzed the multi-task learning

[3] While not wishing to delve too deep into the issue of label bias, we would also like to point out (as it was done, for instance, in (Klein, 2003)) that the label bias of MEMM classifiers can be significantly reduced by allowing them to examine the right context of the classification point - as we have done with our model.

[4] The goal there is to also identify word properties such as gender, number, and case (for nouns), mood and tense (for verbs), etc, besides the main POS tag. The task is slightly different, though, as these properties tend to have a stronger dependency on the lexical form of the classified word.

[5] There is a fourth assigned type – a flag specifying whether a mention is specific (i.e. it refers at a clear entity), generic (refers to a generic type, e.g. "the scientists believe .."), unspecified (cannot be determined from the text), or negative (e.g. "no person would do this"). The classification of this type is beyond the goal of this paper.

(MTL) paradigm, where individual related tasks are trained together by sharing a common representation of knowledge, and demonstrated that this strategy yields better results than one-task-at-a-time learning strategy. The authors used a backpropagation neural network, and the paradigm was tested on several machine learning tasks. It also contains an excellent discussion on how and why the MTL paradigm is superior to single-task learning. Florian and Ngai (2001) used the same multitask learning strategy with a transformation-based learner to show that usually disjointly handled tasks perform slightly better under a joint model; the experiments there were run on POS tagging and text chunking, Chinese word segmentation and POS tagging. Sutton et al. (2004) investigated the multitask classification problem and used a dynamic conditional random fields method, a generalization of linear-chain conditional random fields, which can be viewed as a probabilistic generalization of cascaded, weighted finite-state transducers. The subtasks were represented in a single graphical model that explicitly modeled the sub-task dependence and the uncertainty between them. The system, evaluated on POS tagging and base-noun phrase segmentation, improved on the sequential learning strategy.

In a similar spirit to the approach presented in this article, Florian (2002) considers the task of named entity recognition as a two-step process: the first is the identification of mention boundaries and the second is the classification of the identified chunks, therefore considering a label for each word being formed from two sub-labels: one that specifies the position of the current word relative in a mention (outside any mentions, starts a mention, is inside a mention) and a label specifying the mention type . Experiments on the CoNLL'02 data show that the two-process model yields considerably higher performance.

Hacioglu et al. (2005) explore the same task, investigating the performance of the AIO and the cascade model, and find that the two models have similar performance, with the AIO model having a slight advantage. We expand their study by adding the hybrid *joint* model to the mix, and further investigate different scenarios, showing that the cascade model leads to superior performance most of the time, with a few ties, and show that the cascade model is especially beneficial in cases where *partially-labeled* data (only some of the component labels are given) is available. It turns out though, (Hacioglu, 2005) that the cascade model in (Hacioglu et al., 2005) did not change to a "mention view" sequence classification[6] (as we did in Section 3.3) in the tasks following the entity detection, to allow the system to use longer range features.

3 Classification Models

This section presents the three multi-task classification models, which we will experimentally contrast in Section 4. We are interested in performing sequence classification (e.g. assigning a label to each word in a sentence, otherwise known as tagging). Let $\mathcal{X}$ denote the space of sequence elements (words) and $\mathcal{Y}$ denote the space of classifications (labels), both of them being finite spaces. Our goal is to build a classifier

$$h : \mathcal{X}^+ \to \mathcal{Y}^+$$

which has the property that $|h(\bar{x})| = |\bar{x}|, \forall \bar{x} \in \mathcal{X}^+$ (i.e. the size of the input sequence is preserved). This classifier will select the a posteriori most likely label sequence $\bar{y} = \arg\max_{\bar{y'}} p\left(\bar{y'}|\bar{x}\right)$; in our case $p\left(\bar{y}|\bar{x}\right)$ is computed through the standard Markov assumption:

$$p\left(y_{1,m}|\ \bar{x}\right) = \prod_i p\left(y_i|\bar{x}, y_{i-n+1,i-1}\right) \qquad (1)$$

where $y_{i,j}$ denotes the sequence of labels $y_i..y_j$. Furthermore, we will assume that each label y is composed of a number of sub-labels $y = \left(y^1 y^2 \ldots y^k\right)$[7]; in other words, we will assume the factorization of the label space into k subspaces $\mathcal{Y} = \mathcal{Y}^1 \times \mathcal{Y}^2 \times \ldots \times \mathcal{Y}^k$.

The classifier we used in the experimental section is a maximum entropy classifier (similar to (McCallum et al., 2000))—which can integrate several sources of information in a rigorous manner. It is our empirical observation that, from a performance point of view, being able to use a diverse and abundant feature set is more important than classifier choice, and the maximum entropy framework provides such a utility.

3.1 The All-In-One Model

As the simplest model among those presented here, the all-in-one model ignores the natural factorization of the output space and considers all labels as atomic, and then performs regular sequence classification. One way to look at this process is the following: the classification space $\mathcal{Y} = \mathcal{Y}^1 \times \mathcal{Y}^2 \times \ldots \times \mathcal{Y}^k$ is first mapped onto a same-dimensional space $\mathcal{Z}$ through a one-to-one mapping $o : \mathcal{Y} \to \mathcal{Z}$; then the features of the system are defined on the space $\mathcal{X}^+ \times \mathcal{Z}$, instead of $\mathcal{X}^+ \times \mathcal{Y}$.

While having the advantage of being simple, it suffers from some theoretical disadvantages:

- The classification space can be very large, being the product of the dimensions of sub-task spaces. In the case of the 2004 ACE data there are 7 entity types, 4 mention types and many subtypes; the observed number of actual

[6] As opposed to a "word view".

[7] We can assume, without any loss of generality, that all labels have the same number of sub-labels.

All-In-One Model	Joint Model
B-PER	B-
B-LOC	
B-ORG	
B-MISC	

Table 1: Features predicting start of an entity in the *all-in-one* and *joint* models

sub-label combinations on the training data is 401. Since the dynamic programing (Viterbi) search's runtime dependency on the classification space is $O\left(|\mathcal{Z}|^n\right)$ (n is the Markov dependency size), using larger spaces will negatively impact the decoding run time.[8]

- The probabilities $p\left(z_i|\bar{x}, z_{i-n,i-1}\right)$ require large data sets to be computed properly. If the training data is limited, the probabilities might be poorly estimated.
- The model is not friendly to partial evaluation or weighted sub-task evaluation: different, but partially similar, labels will compete against each other (because the system will return a probability distribution over the classification space), sometimes resulting in wrong partial classification.[9]
- The model cannot *directly* use data that is only partially labeled (i.e. not all sub-labels are specified).

Despite the above disadvantages, this model has performed well in practice: Hajic and Hladká (1998) applied it successfully to find POS sequences for Czech and Florian et al. (2004) reports good results on the 2003 ACE task. Most systems that participated in the CoNLL 2002 and 2003 shared tasks on named entity recognition (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) applied this model, as they modeled the identification of mention boundaries and the assignment of mention type at the same time.

3.2 The Joint Model

The joint model differs from the all-in-one model in the fact that the labels are no longer atomic: the features of the system can inspect the constituent sub-labels. This change helps alleviate the data sparsity encountered by the previous model by allowing sub-label modeling. The joint model theoretically compares favorably with the all-in-one model:

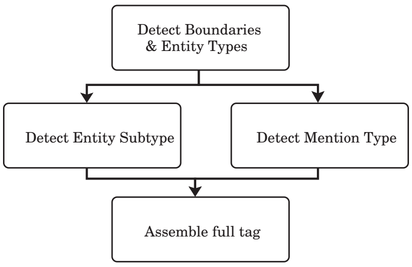


Figure 1: Cascade flow example for mention detection.

- The probabilities $p\left(y_i|\bar{x}, y_{i-n,i-1}\right) = p\left(\left(y_i^1,\ldots,y_i^k\right)|\bar{x},\left(y^j_{i-n,i-1}\right)_{j=1,k}\right)$ might require less training data to be properly estimated, as different sub-labels can be modeled separately.
- The joint model can use features that predict just one or a subset of the sub-labels. Table 1 presents the set of basic features that predict the start of a mention for the CoNLL shared tasks for the two models. While the joint model can encode the start of a mention in one feature, the all-in-one model needs to use four features, resulting in fewer counts per feature and, therefore, yielding less reliably estimated features (or, conversely, it needs more data for the same estimation confidence).
- The model can predict some of the sub-tags ahead of the others (i.e. create a dependency structure on the sub-labels). The model used in the experimental section predicts the sub-labels by using only sub-labels for the previous words, though.
- It is possible, though computationally expensive, for the model to use additional data that is only partially labeled, with the model change presented later in Section 3.4.

3.3 The Cascade Model

For some tasks, there might already exist a natural hierarchy among the sub-labels: some sub-labels could benefit from knowing the value of other, primitive, sub-labels. For example,

- For mention detection, identifying the mention boundaries can be considered as a primitive task. Then, knowing the mention boundaries, one can assign an entity type, subtype, and mention type to each mention.
- In the case of parsing with functional tags, one can perform syntactic parsing, then assign the functional tags to the internal constituents.

[8] From a practical point of view, it might not be very important, as the search is pruned in most cases to only a few hypotheses (beam-search); in our case, pruning the beam only introduced an insignificant model search error (0.1 F-measure).

[9] To exemplify, consider that the system outputs the following classifications and probabilities: O (0.2), B-PER-NAM (0.15), B-PER-NOM (0.15); even the latter 2 suggest that the word is the start of a person mention, the O label will win because the two labels competed against each other.

Words	Since	**Donna**	**Karan**	**International**	went	public	in	1996	...
Labels	O	B-ORG	I-ORG	I-ORG	O	O	O	O	...

Figure 2: Sequence tagging for mention detection: the case for a cascade model.

- For POS tagging, one can detect the main POS first, then detect the other specific properties, making use of the fact that one knows the main tag.

The cascade model is essentially a factorization of individual classifiers for the sub-tasks; in this framework, we will assume that there is a more or less natural dependency structure among subtasks, and that models for each of the subtasks will be built and applied in the order defined by the dependency structure. For example, as shown in Figure 1, one can detect mention boundaries and entity type (at the same time), then detect mention type and subtype in "parallel" (i.e. no dependency exists between these last 2 sub-tags).

A very important advantage of the cascade model is apparent in classification cases where identifying chunks is involved (as is the case with mention detection), similar to advantages that rescoring hypotheses models have: in the second stage, the chunk classification stage, it can switch to a *mention view*, where the classification units are entire mentions and words outside of mentions. This allows the system to make use of aggregate features over the mention words (e.g. all the words are capitalized), and to also effectively use a larger Markov window (instead of 2-3 words, it will use 2-3 chunks/words around the word of interest). Figure 2 contains an example of such a case: the cascade model will have to predict the type of the entire phrase *Donna Karan International*, in the context '*Since <chunk> went public in ..*', which will give it a better opportunity to classify it as an organization. In contrast, because the joint model and AIO have a *word view* of the sentence, will lack the benefit of examining the larger region, and will not have access at features that involve partial future classifications (such as the fact that another mention of a particular type follows).

Compared with the other two models, this classification method has the following advantages:

- The classification spaces for each subtask are considerably smaller; this fact enables the creation of better estimated models
- The problem of partially-agreeing competing labels is completely eliminated
- One can easily use different/additional data to train any of the sub-task models.

3.4 Adding Partially Labeled Data

Annotated data can be sometimes expensive to come by, especially if the label set is complex. But not all sub-tasks were created equal: some of them might be easier to predict than others and, therefore, require less data to train effectively in a cascade setup. Additionally, in realistic situations, some sub-tasks might be considered to have more informational content than others, and have precedence in evaluation. In such a scenario, one might decide to invest resources in annotating additional data only for the particularly interesting sub-task, which could reduce this effort significantly.

To test this hypothesis, we annotated additional data with the entity type only. The cascade model can incorporate this data easily: it just adds it to the training data for the entity type classifier model. While it is not immediately apparent how to incorporate this new data into the all-in-one and joint models, in order to maintain fairness in comparing the models, we modified the procedures to allow for the inclusion. Let $\mathcal{T}$ denote the original training data, and $\mathcal{T}'$ denote the additional training data.

For the all-in-one model, the additional training data cannot be incorporated directly; this is an inherent deficiency of the AIO model. To facilitate a fair comparison, we will incorporate it in an indirect way: we train a classifier C on the additional training data $\mathcal{T}'$, which we then use to classify the original training data $\mathcal{T}$. Then we train the all-in-one classifier on the original training data $\mathcal{T}$, adding the features defined on the output of applying the classifier C on $\mathcal{T}$.

The situation is better for the joint model: the new training data $\mathcal{T}'$ *can* be incorporated directly into the training data $\mathcal{T}$.[10] The maximum entropy model estimates the model parameters by maximizing the data log-likelihood

$$L = \sum_{(x,y)} \hat{p}(x,y) \log q_\lambda (y|x)$$

where $\hat{p}(x,y)$ is the observed probability distribution of the pair (x,y) and $q_\lambda(y|x) = \frac{1}{Z}\prod_j \exp(\lambda_j \cdot f_j(x,y))$ is the conditional ME probability distribution as computed by the model. In the case where some of the data is partially annotated, the log-likelihood becomes

$$L = \sum_{(x,y)\in\mathcal{T}\cup\mathcal{T}'} \hat{p}(x,y) \log q_\lambda (y|x)$$

[10] The solution we present here is particular for MEMM models (though similar solutions may exist for other models as well). We also assume the reader is familiar with the normal MaxEnt training procedure; we present here only the differences to the standard algorithm. See (Manning and Schütze, 1999) for a good description.

$$= \sum_{(x,y)\in\mathcal{T}} \hat{p}(x,y)\log q_\lambda(y|x) + \sum_{(x,y)\in\mathcal{T}'} \hat{p}(x,y)\log q_\lambda(y|x) \qquad (2)$$

The only technical problem that we are faced with here is that we cannot directly estimate the observed probability $\hat{p}(x,y)$ for examples in $\mathcal{T}'$, since they are only partially labeled. Borrowing the idea from the expectation-maximization algorithm (Dempster et al., 1977), we can replace this probability by the re-normalized system proposed probability: for $(x, y_x) \in \mathcal{T}'$, we define

$$\hat{q}(x,y) = \hat{p}(x)\underbrace{\delta(y\in y_x)\frac{q_\lambda(y|x)}{\sum_{y'\in y_x} q_\lambda(y'|x)}}_{=\hat{q}_\lambda(y|x)}$$

where y_x is the subset of labels from $\mathcal{Y}$ which are consistent with the partial classification of x in $\mathcal{T}'$. $\delta(y\in y_x)$ is 1 if and only if y is consistent with the partial classification y_x.[11] The log-likelihood computation in Equation (2) becomes

$$L = \sum_{(x,y)\in\mathcal{T}} \hat{p}(x,y)\log q_\lambda(y|x) + \sum_{(x,y)\in\mathcal{T}'} \hat{q}(x,y)\log q_\lambda(y|x)$$

To further simplify the evaluation, the quantities $\hat{q}(x,y)$ are recomputed every few steps, and are considered constant as far as finding the optimum λ values is concerned (the partial derivative computations and numerical updates otherwise become quite complicated, and the solution is no longer unique). Given this new evaluation function, the training algorithm will proceed exactly the same way as in the normal case where all the data is fully labeled.

4 Experiments

All the experiments in this section are run on the ACE 2003 and 2004 data sets, in all the three languages covered: Arabic, Chinese, and English. Since the evaluation test set is not publicly available, we have split the publicly available data into a 80%/20% data split. To facilitate future comparisons with work presented here, and to simulate a realistic scenario, the splits are created based on article dates: the test data is selected as the last 20% of the data in chronological order. This way, the documents in the training and test data sets do not overlap in time, and the ones in the test data are posterior to the ones in the training data. Table 2 presents the number of documents in the training/test datasets for the three languages.

Language	Training	Test
Arabic	511	178
Chinese	480	166
English 2003	658	139
English 2004	337	114

Table 2: Datasets size (number of documents)

Each word in the training data is labeled with one of the following properties:[12]

- if it is not part of any entity, it's labeled as O
- if it is part of an entity, it contains a tag specifying whether it starts a mention (B-) or is inside a mention (I-). It is also labeled with the entity type of the mention (seven possible types: person, organization, location, facility, geo-political entity, weapon, and vehicle), the mention type (named, nominal, pronominal, or premodifier[13]), and the entity subtype (depends on the main entity type).

The underlying classifier used to run the experiments in this article is a maximum entropy model with a Gaussian prior (Chen and Rosenfeld, 1999), making use of a large range of features, including lexical (words and morphs in a 3-word window, prefixes and suffixes of length up to 4, WordNet (Miller, 1995) for English), syntactic (POS tags, text chunks), gazetteers, and the output of other information extraction models. These features were described in (Florian et al., 2004), and are not discussed here. All three methods (AIO, joint, and cascade) instantiate classifiers based on the same feature types whenever possible. In terms of language-specific processing, the Arabic system uses as input morphological segments, while the Chinese system is a character-based model (the input elements $x \in \mathcal{X}$ are characters), but it has access to word segments as features.

Performance in the ACE task is officially evaluated using a special-purpose measure, the ACE value metric (NIST, 2003; NIST, 2004). This metric assigns a score based on the similarity between the system's output and the gold-standard at both mention and entity level, and assigns different weights to different entity types (e.g. the person entity weights considerably more than a facility entity, at least in the 2003 and 2004 evaluations). Since this article focuses on the mention detection task, we decided to use the more intuitive (unweighted) F-measure: the harmonic mean of precision and recall.

[11] For instance, the full label *B-PER* is consistent with the partial label *B*, but not with *O* or *I*.

[12] The mention encoding is the IOB2 encoding presented in (Tjong Kim Sang and Veenstra, 1999) and introduced by (Ramshaw and Marcus, 1994) for the task of base noun phrase chunking.

[13] This is a special class, used for mentions that modify other labeled mentions; e.g. French in *"French* wine". This tag is specific only to ACE'04.

For the cascade model, the sub-task flow is presented in Figure 1. In the first step, we identify the mention boundaries together with their entity type (e.g. person, organization, etc). In preliminary experiments, we tried to "cascade" this task. The performance was similar on both strategies; the separated model would yield higher recall at the expense of precision, while the combined model would have higher precision, but lower recall. We decided to use in the system with higher precision. Once the mentions are identified and classified with the entity type property, the data is passed, in parallel, to the mention type detector and the subtype detector.

For English and Arabic, we spent three person-weeks to annotate additional data labeled with only the entity type information: 550k words for English and 200k words for Arabic. As mentioned earlier, adding this data to the cascade model is a trivial task: the data just gets added to the training data, and the model is retrained. For the AIO model, we have build another mention classifier on the additional training data, and labeled the original ACE training data with it. It is important to note here that the ACE training data (called $\mathcal{T}$ in Section 3.4) is consistent with the additional training data $\mathcal{T}'$: the annotation guidelines for $\mathcal{T}'$ are the same as for the original ACE data, but we only labeled entity type information. The resulting classifications are then used as features in the final AIO classifier. The joint model uses the additional partially-labeled data in the way described in Section 3.4; the probabilities $\hat{q}\,(x, y)$ are updated every 5 iterations.

Table 3 presents the results: overall, the cascade model performs significantly better than the all-in-one model in four out the six tested cases - the numbers presented in bold reflect that the difference in performance to the AIO model is statistically significant.[14] The joint model, while managing to recover some ground, falls in between the AIO and the cascade models.

When additional partially-labeled data was available, the cascade and joint models receive a statistically significant boost in performance, while the all-in-one model's performance barely changes. This fact can be explained by the fact that the entity type-only model is in itself errorful; measuring the performance of the model on the training data yields a performance of 82 F-measure;[15] therefore the AIO model will only access partially-correct data, and is unable to make effective use of it. In contrast, the training data for the entity type in the cascade model effectively triples, and this change is reflected positively in the 1.5 increase in F-measure.

Language	Data$^+$	A-I-O	Joint	Cascade
Arabic'04	no	59.2	59.1	**59.7**
	yes	59.4	**60.0**	**60.7**
English'04	no	72.1	72.3	**73.7**
	yes	72.5	**74.1**	**75.2**
Chinese'04	no	71.2	71.7	71.7
English '03	no	79.5	79.5	79.7

Table 3: Experimental results: F-measure on the full label

Language	Data$^+$	A-I-O	Joint	Cascade
Arabic'04	no	66.3	66.5	**67.5**
	yes	66.4	**67.9**	**68.9**
English'04	no	77.9	78.1	**79.2**
	yes	78.3	**80.5**	**82.6**
Chinese'04	no	75.4	76.1	**76.8**
English '03	no	80.4	80.4	81.1

Table 4: F-measure results on entity type only

Not all properties are equally valuable: the entity type is arguably more interesting than the other properties. If we restrict ourselves to evaluating the entity type output only (by projecting the output label to the entity type only), the difference in performance between the all-in-one model and cascade is even more pronounced, as shown in Table 4. The cascade model outperforms here both the all-in-one and joint models in all cases except English'03, where the difference is not statistically significant.

As far as run-time speed is concerned, the AIO and cascade models behave similarly: our implementation tags approximately 500 tokens per second (averaged over the three languages, on a Pentium 3, 1.2Ghz, 2Gb of memory). Since a MaxEnt implementation is mostly dependent on the number of features that fire on average on a example, and not on the total number of features, the joint model runs twice as slow: the average number of features firing on a particular example is considerably higher. On average, the joint system can tag approximately 240 words per second. The train time is also considerably longer; it takes 15 times as long to train the joint model as it takes to train the all-in-one model (60 mins/iteration compared to 4 mins/iteration); the cascade model trains faster than the AIO model.

One last important fact that is worth mentioning is that a system based on the cascade model participated in the ACE'04 competition, yielding very competitive results in all three languages.

[14] To assert the statistical significance of the results, we ran a paired Wilcoxon test over the series obtained by computing F-measure on each document in the test set. The results are significant at a level of at least 0.009.

[15] Since the additional training data is consistent in the labeling of the entity type, such a comparison is indeed possible. The above mentioned score is on entity types only.

5 Conclusion

As natural language processing becomes more sophisticated and powerful, we start focus our attention on more and more properties associated with the objects we are seeking, as they allow for a deeper and more complex representation of the real world. With this focus comes the question of how this goal should be accomplished – either detect all properties at once, one at a time through a pipeline, or a hybrid model. This paper presents three methods through which multi-label sequence classification can be achieved, and evaluates and contrasts them on the Automatic Content Extraction task. On the ACE mention detection task, the cascade model which predicts first the mention boundaries and entity types, followed by mention type and entity subtype outperforms the simple all-in-one model in most cases, and the joint model in a few cases.

Among the proposed models, the cascade approach has the definite advantage that it can easily and productively incorporate additional partially-labeled data. We also presented a novel modification of the joint system training that allows for the direct incorporation of additional data, which increased the system performance significantly. The all-in-one model can only incorporate additional data in an indirect way, resulting in little to no overall improvement.

Finally, the performance obtained by the cascade model is very competitive: when paired with a coreference module, it ranked very well in the "Entity Detection and Tracking" task in the ACE'04 evaluation.

References

R. Caruana, L. Pratt, and S. Thrun. 1997. Multitask learning. *Machine Learning*, 28:41.

Stanley F. Chen and Ronald Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, Computer Science Department, Carnegie Mellon University.

A. P. Dempster, 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.

R. Florian and G. Ngai. 2001. Multidimensional transformation-based learning. In *Proceedings of CoNLL'01*, pages 1–8.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N Nicolov, and S Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 1–8.

R. Florian. 2002. Named entity recognition as a house of cards: Classifier stacking. In *Proceedings of CoNLL-2002*, pages 175–178.

Kadri Hacioglu, Benjamin Douglas, and Ying Chen. 2005. Detection of entity mentions occuring in english and chinese text. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 379–386, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Kadri Hacioglu. 2005. Private communication.

J. Hajic and Hladká. 1998. Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *Proceedings of the 36th Annual Meeting of the ACL and the 17th ICCL*, pages 483–490, Montréal, Canada.

H. Jing, R. Florian, X. Luo, T. Zhang, and A. Ittycheriah. 2003. HowtogetaChineseName(Entity): Segmentation and combination issues. In *Proceedings of EMNLP'03*, pages 200–207.

Dan Klein. 2003. Maxent models, conditional estimation, and optimization, without the magic. Tutorial presented at NAACL-03 and ACL-03.

C. D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of ICML-2000*.

G. A. Miller. 1995. WordNet: A lexical database. *Communications of the ACM*, 38(11).

MUC-6. 1995. The sixth message understanding conference. www.cs.nyu.edu/cs/faculty/grishman/muc6.html.

MUC-7. 1997. The seventh message understanding conference. www.itl.nist.gov/iad/894.02/related_projects/muc/proceedings/muc_7_toc.html.

NIST. 2003. The ACE evaluation plan. www.nist.gov/speech/tests/ace/index.htm.

NIST. 2004. The ACE evaluation plan. www.nist.gov/speech/tests/ace/index.htm.

L. Ramshaw and M. Marcus. 1994. Exploring the statistical derivation of transformational rule sequences for part-of-speech tagging. In *Proceedings of the ACL Workshop on Combining Symbolic and Statistical Approaches to Language*, pages 128–135.

C. Sutton, K. Rohanimanesh, and A. McCallum. 2004. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *In Proceedings of the Twenty-First International Conference on Machine Learning (ICML-2004)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

E. F. Tjong Kim Sang and J. Veenstra. 1999. Representing text chunks. In *Proceedings of EACL'99*.

E. F. Tjong Kim Sang. 2002. Introduction to the conll-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158.

Segment-based Hidden Markov Models for Information Extraction

Zhenmei Gu
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2l 3G1
z2gu@uwaterloo.ca

Nick Cercone
Faculty of Computer Science
Dalhousie University
Halifax, Nova Scotia, Canada B3H 1W5
nick@cs.dal.ca

Abstract

Hidden Markov models (HMMs) are powerful statistical models that have found successful applications in Information Extraction (IE). In current approaches to applying HMMs to IE, an HMM is used to model text at the document level. This modelling might cause undesired redundancy in extraction in the sense that more than one filler is identified and extracted. We propose to use HMMs to model text at the segment level, in which the extraction process consists of two steps: a segment retrieval step followed by an extraction step. In order to retrieve extraction-relevant segments from documents, we introduce a method to use HMMs to model and retrieve segments. Our experimental results show that the resulting segment HMM IE system not only achieves near zero extraction redundancy, but also has better overall extraction performance than traditional document HMM IE systems.

1 Introduction

A Hidden Markov Model (HMM) is a finite state automaton with stochastic state transitions and symbol emissions (Rabiner, 1989). The automaton models a random process that can produce a sequence of symbols by starting from some state, transferring from one state to another state with a symbol being emitted at each state, until a final state is reached. Formally, a hidden Markov model (HMM) is specified by a five-tuple (S, K, Π, A, B), where S is a set of states; K is the alphabet of observation symbols; Π is the initial state distribution; A is the probability distribution of state transitions; and B is the probability distribution of symbol emissions. When the structure of an HMM is determined, the complete model parameters can be represented as $\lambda = (A, B, \Pi)$.

HMMs are particularly useful in modelling sequential data. They have been applied in several areas within natural language processing (NLP), with one of the most successful efforts in speech recognition. HMMs have also been applied in information extraction. An early work of using HMMs for IE is (Leek, 1997) in which HMMs are trained to extract gene name-location facts from a collection of scientific abstracts. Another related work is (Bikel et al., 1997) which used HMMs as part of its modelling for the name finding problem in information extraction.

A more recent work on applying HMMs to IE is (Freitag and McCallum, 1999), in which a separate HMM is built for extracting fillers for each slot. To train an HMM for extracting fillers for a specific slot, maximum likelihood estimation is used to determine the probabilities (i.e., the initial state probabilities, the state transition probabilities, and the symbol emission probabilities) associated with each HMM from labelled texts.

One characteristic of current HMM-based IE systems is that an HMM models the entire document. Each document is viewed as a long sequence of tokens (i.e., words, punctuation marks etc.), which is the observation generated from the given HMM. The extraction is performed by finding the best state sequence for this observed long token sequence constituting the whole document, and the subsequences of tokens that pass through the target filler state are extracted as fillers. We call such approaches to applying HMMs to IE at the document level as document-based HMM IE or *document HMM IE* for brevity.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 481–488,
Sydney, July 2006.

In addition to HMMs, there are other Markovian sequence models that have been applied to IE. Examples of these models include maximum entropy Markov models (McCallum et al., 2000), Bayesian information extraction network (Peshkin and Pfeffer, 2003), and conditional random fields (McCallum, 2003) (Peng and McCallum, 2004). In the IE systems using these models, extraction is performed by sequential tag labelling. Similar to HMM IE, each document is considered to be a single steam of tokens in these IE models as well.

In this paper, we introduce the concept of extraction redundancy, and show that current document HMM IE systems often produce undesired redundant extractions. In order to address this extraction redundancy issue, we propose a segment-based two-step extraction approach in which a segment retrieval step is imposed before the extraction step. Our experimental results show that the resulting segment-based HMM IE system not only achieves near-zero extraction redundancy but also improves the overall extraction performance.

This paper is organized as follows. In section 2, we describe our document HMM IE system in which the Simple Good-Turning (SGT) smoothing is applied for probability estimation. We also evaluate our document HMM IE system, and compare it to the related work. In Section 3, we point out the extraction redundancy issue in a document HMM IE system. The definition of the extraction redundancy is introduced for better evaluation of an IE system with possible redundant extraction. In order to address this extraction redundancy issue, we propose our segment-based HMM IE method in Section 4, in which a segment retrieval step is applied before the extraction is performed. Section 5 presents a segment retrieval algorithm by using HMMs to model and retrieve segments. We compare the performance between the segment HMM IE system and the document HMM IE system in Section 6. Finally, conclusions are made and some future work is mentioned in Section 7.

2 Document-based HMM IE with the SGT smoothing

2.1 HMM structure

We use a similar HMM structure (named as HMM_Context) as in (Freitag and McCallum, 1999) for our document HMM IE system. An example of such an HMM is shown in Figure 1, in which the number of pre-context states, post-context states, and the number of parallel filler paths are all set to 4, the default model parameter setting in our system.

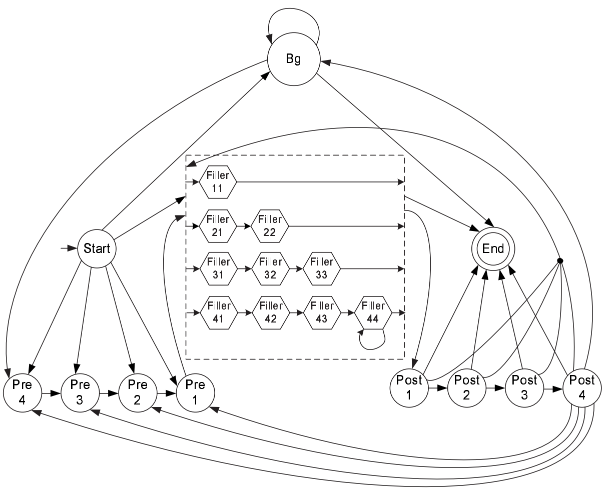


Figure 1: An example of HMM_Context structure

HMM_Context consists of the following four kinds of states in addition to the special *start* and *end* states.

Filler states $Filler_{mn}$, $m = 1, 2, 3, 4$ and $n = 1, \cdots, m$ states, correspond to the occurrences of filler tokens.

Background state This state corresponds to the occurrences of the tokens that are not related to fillers or their contexts.

Pre context states $Pre_4, Pre_3, Pre_2, Pre_1$ states correspond to the events present when context tokens occur before the fillers at the specific positions relative to the fillers, respectively.

Post context states $Post_1, Post_2, Post_3, Post_4$ states correspond to the events present when context tokens occur after the fillers at the specific positions relative to the fillers, respectively.

Our HMM structure differs from the one used in (Freitag and McCallum, 1999) in that we have added the transitions from the last post context state to every pre context state as well as every first filler state. This handles the situation where two filler occurrences in the document are so close to each other that the text segment between these two fillers is shorter than the sum of the pre context and the post context sizes.

2.2 Smoothing in HMM IE

There are many probabilities that need to be estimated to train an HMM for information extraction from a limited number of labelled documents. The data sparseness problem commonly occurring in probabilistic learning would also be an issue in the training for an HMM IE system, especially when more advanced HMM_Context models are used. Since the emission vocabulary is usually large with respect to the number of training examples, maximum likelihood estimation of emission probabilities will lead to inappropriate zero probabilities for many words in the alphabet.

The Simple Good-Turning (SGT) smoothing (Gale and Sampson, 1995) is a simple version of Good-Turning approach, which is a population frequency estimator used to adjust the observed term frequencies to estimate the real population term frequencies. The observed frequency distribution from the sample can be represented as a vector of (r, n_r) pairs, $r = 1, 2, \cdots$. r values are the observed term frequencies from the training data, and n_r refers to the number of different terms that occur with frequency r in the sample.

For each r observed in the sample, the Good-Turning method gives an estimation for its real population frequency as $r^* = (r+1)\frac{E(n_{r+1})}{E(n_r)}$, where $E(n_r)$ is the expected number of terms with frequency r. For unseen events, an amount of probability P_0 is assigned to all these unseen events, $P_0 = \frac{E(n_1)}{N} \approx \frac{n_1}{N}$, where N is the total number of term occurrences in the sample.

The SGT smoothing has been successfully applied to naive Bayes IE systems in (Gu and Cercone, 2006) for more robust probability estimation. We apply the SGT smoothing method to our HMM IE systems to alleviate the data sparseness problem in HMM training. In particular, the emission probability distribution for each state is smoothed using the SGT method. The number of unseen emission terms is estimated, as the observed alphabet size difference between the specific state emission term distribution and the all term distribution, for each state before assigning the total unseen probability obtained from the SGT smoothing among all these unseen terms.

The data sparseness problem in probability estimation for HMMs has been addressed to some extent in previous HMM based IE systems (e.g., (Leek, 1997) and (Freitag and McCallum, 1999)). Smoothing methods such as absolute discounting have been used for this purpose. Moreover, (Freitag and McCallum, 1999) uses a *shrinkage* technique for estimating word emission probabilities of HMMs in the face of sparse training data. It first defines a shrinkage topology over HMM states, then learns the mixture weights for producing interpolated emission probabilities by using a separate data set that is "held-out" from the labelled data. This technique is called *deleted interpolation* in speech recognition (Jelinek and Mercer, 1980).

2.3 Experimental results on document HMM IE and comparison to related work

We evaluated our document HMM IE system on the seminar announcements IE domain using tenfold cross validation evaluation. The data set consists of 485 annotated seminar announcements, with the fillers for the following four slots specified for each seminar: *location* (the location of a seminar), *speaker* (the speaker of a seminar), *stime* (the starting time of a seminar) and *etime* (the ending time of a seminar). In our HMM IE experiments, the structure parameters are set to system default values, i.e., 4 for both pre-context and post-context size, and 4 for the number of parallel filler paths.

Table 1 shows F1 scores (95% confidence intervals) of our Document HMM IE system (Doc_HMM). The performance numbers from other HMM IE systems (Freitag and McCallum, 1999) are also listed in Table 1 for comparison, where HMM_None is their HMM IE system that uses absolute discounting but with no shrinkage, and HMM_Global is the representative version of their HMM IE system with shrinkage.

By using the same structure parameters (i.e., the same context size) as in (Freitag and McCallum, 1999), our Doc_HMM system performs consistently better on all slots than their HMM IE system using absolute discounting. Even compared to their much more complex version of HMM IE with shrinkage, our system has achieved comparable results on *location*, *speaker* and *stime*, but obtained significantly better performance on the *etime* slot. It is noted that our smoothing method is much simpler to apply, and does not require any extra effort such as specifying shrinkage topology or any extra labelled data for a held-out set.

Table 1: F1 of Document HMM IE systems on seminar announcements

Learner	location	speaker	stime	etime
Doc_HMM	0.8220±0.022	0.7135±0.025	1.0000±0.0	0.9488±0.012
HMM_None	0.735	0.513	0.991	0.814
HMM_Global	0.839	0.711	0.991	0.595

3 Document extraction redundancy in HMM IE

3.1 Issue with document-based HMM IE

In existing HMM based IE systems, an HMM is used to model the entire document as one long observation sequence emitted from the HMM. The extracted fillers are identified by any part of the sequence in which tokens in it are labelled as one of the filler states. The commonly used structure of the hidden Markov models in IE allows multiple passes through the paths of the filler states. So it is possible for the labelled state sequences to present multiple filler extractions.

It is not known from the performance reports from previous works (e.g., (Freitag and McCallum, 1999)) that how exactly a correct extraction for one document is defined in HMM IE evaluation. One way to define a correct extraction for a document is to require that at least one of the text segments that pass the filler states is the same as a labelled filler. Alternatively, we can define the correctness by requiring that all the text segments that pass the filler states are same as the labelled fillers. In this case, it is actually required an exact match between the HMM state sequence determined by the system and the originally labelled one for that document. Very likely, the former correctness criterion was used in evaluating these document-based HMM IE systems. We used the same criterion for evaluating our document HMM IE systems in Section 2.

Although it might be reasonable to define that a document is correctly extracted if any one of the identified fillers from the state sequence labelled by the system is a correct filler, certain issues exist when a document HMM IE system returns multiple extractions for the same slot for one document. For example, it is possible that some of the fillers found by the system are not correct extractions. In this situation, such document-wise extraction evaluation alone would not be sufficient to measure the performance of an HMM IE system.

Document HMM IE modelling does provide any guidelines for selecting one mostly likely filler from the ones identified by the state sequence matching over the whole document. For the template filling IE problem that is of our interest in this paper, the ideal extraction result is one slot filler per document. Otherwise, some further post-processing would be required to choose only one extraction, from the multiple fillers possibly extracted by a document HMM IE system, for filling in the slot template for that document.

3.2 Concept of document extraction redundancy in HMM IE

In order to make a more complete extraction performance evaluation in an HMM-based IE system, we introduce another performance measure, *document extraction redundancy* as defined in Definition 1, to be used with the document-wise extraction correctness measure .

Definition 1. Document extraction redundancy *is defined over the documents that contain correct extraction(s), as the ratio of the* **incorrectly** *extracted fillers to all returned fillers from the document HMM IE system.*

For example, when the document HMM IE system issues more than one slot extraction for a document, if all the issued extractions are correct ones, then the extraction redundancy for that document is 0. Among all the issued extractions, the larger of the number of incorrect extractions is, the closer the extraction redundancy for that document is to 1. However, the extraction redundancy can never be 1 according to our definition, since this measure is only defined over the documents that contain at lease one correct extraction.

Now let us have a look at the extraction redundancy in the document HMM IE system from Section 2. We calculate the average document extraction redundancy over all the documents that are judged as correctly extracted. The evaluation results for the document extraction redundancy (shown in column R) are listed in Table 2, paired with their corresponding F1 scores from the document-wise extraction evaluation.

Table 2: F1 / redundancy in document HMM IE on SA domain

Slot	F1	R
location	0.8220	0.0543
speaker	0.7135	0.0952
stime	1.0000	0.1312
etime	0.9488	0.0630

Generally speaking, the HMM IE systems based on document modelling has exhibited a certain extraction redundancy for any slot in this IE domain, and in some cases such as for *speaker* and *stime*, the average extraction redundancy is by all means not negligible.

4 Segment-based HMM IE Modelling

In order to make the IE system capable of producing the ideal extraction result that issues only one slot filler for each document, we propose a segment-based HMM IE framework in the following sections of this paper. We expect this framework can dramatically reduce the document extraction redundancy and make the resulting IE system output extraction results to the template filling IE task with the least post-processing requirement.

The basic idea of our approach is to use HMMs to extract fillers from only *extraction-relevant* part of text instead of the entire document. We refer to this modelling as segment-based HMM IE, or *segment HMM IE* for brevity. The unit of the extraction-relevant text segments is definable according to the nature of the texts. For most texts, one sentence in the text can be regarded as a text segment. For some texts that are not written in a grammatical style and sentence boundaries are hard to identify, we can define a *extraction-relevant* text segment be the part of text that includes a filler occurrence and its contexts.

4.1 Segment-based HMM IE modelling: the procedure

By imposing an extraction-relevant text segment retrieval in the segment HMM IE modelling, we perform an extraction on a document by completing the following two successive sub-tasks.

Step 1: Identify from the entire documents the text segments that are relevant to a specific slot extraction. In other words, the document is filtered by locating text segments that might contain a filler.

Step 2: Extraction is performed by applying the segment HMM only on the extraction-relevant text segments that are obtained from the first step. Each retrieved segment is labelled with the most probable state sequence by the HMM, and all these segments are sorted according to their normalized likelihoods of their best state sequences. The filler(s) identified by the segment having the largest likelihood is/are returned as the extraction result.

4.2 Extraction from relevant segments

Since it is usual that more than one segment have been retrieved at Step 1, these segments need to compete at step 2 for issuing extraction(s) from their best state sequences found with regard to the HMM λ used for extraction. For each segment s with token length of n, its normalized best state sequence likelihood is defined as follows.

$$l(s) = \log\bigl(\max_{all\ Q} P(Q, s|\lambda)\bigr) \times \frac{1}{n}, \qquad (1)$$

where λ is the HMM and Q is any possible state sequence associated with s. All the retrieved segments are then ranked according to their $l(s)$, and the segment with the highest $l(s)$ number is selected and the extraction is identified from its labelled state sequence by the segment HMM.

This proposed two-step HMM based extraction procedure requires that the training of the IE models follows the same style. First, we need to learn an extraction-relevance segment retrieval system from the labelled texts which will be described in detail in Section 5. Then, an HMM is trained for each slot extraction by only using the extraction-relevant text segments instead of the whole documents.

By limiting the HMM training to a much smaller part of the texts, basically including the fillers and their surrounding contexts, the alphabet size of all emission symbols associated with the HMM would be significantly reduced. Compared to the common document-based HMM IE modelling, our proposed segment-based HMM IE modelling would also ease the HMM training difficulty caused by the data sparseness problem since we are working on a smaller alphabet.

5 Extraction-relevant segment retrieval using HMMs

We propose a segment retrieval approach for performing the first subtask by also using HMMs. In particular, it trains an HMM from labelled segments in texts, and then use the learned HMM to determine whether a segment is relevant or not with regard to a specific extraction task. In order to distinguish the HMM used for segment retrieval in the first step from the HMM used for the extraction in the second step, we call the former one as the *retrieval HMM* and the later one as the *extractor HMM*.

5.1 Training HMMs for segment retrieval

To train a retrieval HMM, it requires each training segment to be labelled in the same way as in the annotated training document. After the training texts are segmented into sentences (we are using sentence as the segment unit), the obtained segments that carry the original slot filler tags are used directly as the training examples for the retrieval HMM.

An HMM with the same IE specific structure is trained from the prepared training segments in exactly the same way as we train an HMM in the document HMM IE system from a set of training documents. The difference is that much shorter labelled observation sequences are used.

5.2 Segment retrieval using HMMs

After a retrieval HMM is trained from the labelled segments, we use this HMM to determine whether an unseen segment is relevant or not to a specific extraction task. This is done by estimating, from the HMM, how likely the associated state sequence of the given segment passes the target filler states. The HMM λ trained from labelled segments has the structure as shown in Figure 1. So for a segment s, all the possible state sequences can be categorized into two kinds: the state sequences passing through one of the target filler path, and the state sequences not passing through any target filler states.

Because of the structure constraints of the specified HMM in IE, we can see that the second kind of state sequences actually have only one possible path, denoted as Q_{bg} in which the whole observation sequence of s starts at the background state q_{bg} and continues staying in the background state until the end. Let $s = O_1O_2 \cdots O_T$, where T is the length of s in tokens. The probability of s following this particular background state path Q_{bg} can be easily calculated with respect to the HMM λ as follows:

$$\begin{aligned} P(s, Q_{bg}|\lambda) =& \pi_{q_{bg}} b_{q_{bg}}(O_1) a_{q_{bg}q_{bg}} b_{q_{bg}}(O_2) \\ & \cdots a_{q_{bg}q_{bg}} b_{q_{bg}}(O_T), \end{aligned}$$

where π_i is the initial state probability for state i, $b_i(O_t)$ is the emission probability of symbol O_t at state i, and a_{ij} is the state transition probability from state i to state j.

We know that the probability of observing s given the HMM λ actually sums over the probabilities of observing s on all the possible state sequences given the HMM, i.e.,

$$P(s|\lambda) = \sum_{all\ Q} P(s, Q|\lambda)$$

Let Q_{filler} denote the set of state sequences that pass through any filler states. We have $\{all\ Q\} = Q_{bg} \cup Q_{filler}$. $P(s|\lambda)$ can be calculated efficiently using the forward-backward procedure which makes the estimate for the total probability of all state paths that go through filler states straightforward to be:

$$\begin{aligned} P(s, Q_{filler}|\lambda) \triangleq & \sum_{allQ \in Q_{filler}} P(s, Q|\lambda) \\ & = P(s|\lambda) - P(s, Q_{bg}|\lambda). \end{aligned}$$

Now it is clear to see that, if the calculated $P(s, Q_{filler}|\lambda) > P(s, Q_{bg}|\lambda)$, then segment s is considered more likely to have filler occurrence(s). Therefore in this case we classify s as an extraction relevant segment and it will be retrieved.

5.3 Document-wise retrieval performance

Since the purpose of our segment retrieval is to identify relevant segments from each document, we need to define how to determine whether a document is correctly filtered (i.e., with extraction relevant segments retrieved) by a given segment retrieval system. We consider two criteria, first a loose correctness definition as follows:

Definition 2. *A document is **least correctly filtered** by the segment retrieval system when **at least one** of the extraction relevant segments in that document has been retrieved by the system; otherwise, we say the system fails on that document.*

Then we define a stricter correctness measure as follows:

Definition 3. *A document is* ***most correctly filtered*** *by the segment retrieval system only when* ***all*** *the extraction relevant segments in that document have been retrieved by the system; otherwise, we say the system fails on that document.*

The overall segment retrieval performance is measured by *retrieval precision* (i.e., ratio of the number of correctly filtered documents to the number of documents from which the system has retrieved at least one segments) and *retrieval recall* (i.e., ratio of the number of correctly filtered documents to the number of documents that contain relevant segments). According to the just defined two correctness measures, the overall retrieval performance for the all testing documents can be evaluated under both the *least correctly filtered* and the *least correctly filtered* measures.

We also evaluate average *document-wise segment retrieval redundancy*, as defined in Definition 4 to measure the segment retrieval accuracy.

Definition 4. ***Document-wise segment retrieval redundancy*** *is defined over the documents which are least correctly filtered by the segment retrieval system, as the ratio of the retrieved* ***irrelevant*** *segments to all retrieved segments for that document.*

5.4 Experimental results on segment retrieval

Table 3 shows the document-wise segment retrieval performance evaluation results under both *least correctly filtered* and *most correctly filtered* measures, as well as the related average number of retrieved segments for each document (as in Column **nSeg**) and the average retrieval redundancy.

Shown from Table 3, the segment retrieval results have achieved high recall especially with the *least correctly filtered* correctness criterion. In addition, the system has produced the retrieval results with relatively small redundancy which means most of the segments that are fed to the segment HMM extractor from the retrieval step are actually extraction-related segments.

6 Segment vs. document HMM IE

We conducted experiments to evaluate our segment-based HMM IE model, using the proposed segment retrieval approach, and comparing their final extraction performance to the document-based HMM IE model. Table 4 shows the overall performance comparison between the document HMM IE system (Doc_HMM) and the segment HMM IE system (Seg_HMM).

Compared to the document-based HMM IE modelling, the extraction performance on *location* is significantly improved by our segment HMM IE system. The important improvement from the segment HMM IE system that it has achieved zero extraction redundancy for all the slots in this experiment.

7 Conclusions and future work

In current HMM based IE systems, an HMM is used to model at the document level which causes certain redundancy in the extraction. We propose a segment-based HMM IE modelling method in order to achieve near-zero redundancy extraction. In our segment HMM IE approach, a segment retrieval step is first applied so that the HMM extractor identifies fillers from a smaller set of extraction-relevant segments. The resulting segment HMM IE system using the segment retrieval method has not only achieved nearly zero extraction redundancy, but also improved the overall extraction performance. The effect of the segment-based HMM extraction goes beyond applying a post-processing step to the document-based HMM extraction, since the latter can only reduce the redundancy but not improve the F1 scores.

For the template-filling style IE problems, it is more reasonable to perform extraction by HMM state labelling on segments, instead of on the entire document. When the observation sequence to be labelled becomes longer, finding the best single state sequence for it would become a more difficult task. Since the effect of changing a small part in a very long state sequence would not be as obvious, with regard to the state path probability calculation, as changing the same subsequence in a much shorter state sequence. In fact, this perspective not only applies in HMM IE modelling, but also applies in any IE modelling in which extraction is performed by sequential state labelling. We are working on extending this segment-based framework to other Markovian sequence models used for IE.

Segment retrieval for extraction is an important step in segment HMM IE, since it filters out irrelevant segments from the document. The HMM for extraction is supposed to model extraction-relevant segments, so the irrelevant segments that are fed to the second step would make the extraction more difficult by adding noise to the competition among relevant segments. We have

Table 3: Segment retrieval results

Slot	*least correctly*		*most correctly*			
	Precision	**Recall**	**Precision**	**Recall**	**nSeg**	**Redundancy**
location	0.8948	0.9177	0.8758	0.8982	2.6064	0.4569
speaker	0.8791	0.7633	0.6969	0.6042	1.6082	0.1664
stime	1.0000	1.0000	0.9464	0.9464	2.6576	0.1961
etime	0.4717	0.9952	0.4570	0.9609	1.7896	0.1050

Table 4: F1 comparison on seminar announcements (document HMM IE vs. segment HMM IE)

Learner	**location**		**speaker**		**stime**		**etime**	
	F1	R	F1	R	F1	R	F1	R
Doc_HMM	0.822±0.022	0.0543	0.7135±0.025	0.0952	1.0000±0.0	0.131	0.9488±0.012	0.063
Seg_HMM	0.8798±0.018	0	0.7162±0.025	0	0.998±0.003	0	0.9611±0.011	0

presented and evaluated our segment retrieval method. Document-wise retrieval performance can give us more insights on the goodness of a particular segment retrieval method for our purpose: the document-wise retrieval recall using the *least correctly filtered* measure provides an upper bound on the final extraction performance.

Our current segment retrieval method requires the training documents to be segmented in advance. Although sentence segmentation is a relatively easy task in NLP, some segmentation errors are still unavoidable especially for ungrammatical online texts. For example, an improper segmentation could set a segment boundary in the middle of a filler, which would definitely affect the final extraction performance of the segment HMM IE system. In the future, we intend to design segment retrieval methods that do not require documents to be segmented before retrieval, hence avoiding the possibility of early-stage errors introduced from the text segmentation step. A very promising idea is to adapt a naive Bayes IE to perform redundant extractions directly on an entire document to retrieve filler-containing text segments for a segment HMM IE system.

References

[Bikel et al.1997] D. M. Bikel, S. Miller, R. Schwartz, and R. Weischedel. 1997. Nymble: a high-performance learning name-finder. In *Proceedings of ANLP-97*, pages 194–201.

[Freitag and McCallum1999] D. Freitag and A. McCallum. 1999. Information extraction with HMMs and shrinkage. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*.

[Gale and Sampson1995] W. Gale and G. Sampson. 1995. Good-turning smoothing without tears. *Journal of Quantitative Linguistics*, 2:217–37.

[Gu and Cercone2006] Z. Gu and N. Cercone. 2006. Naive bayes modeling with proper smoothing for information extraction. In *Proceedings of the 2006 IEEE International Conference on Fuzzy Systems*.

[Jelinek and Mercer1980] F. Jelinek and R. L. Mercer. 1980. Intepolated estimation of markov source parameters from sparse data. In E. S. Gelesma and L. N. Kanal, editors, *Proceedings of the Wrokshop on Pattern Recognition in Practice*, pages 381–397, Amsterdam, The Netherlands: North-Holland, May.

[Leek1997] T. R. Leek. 1997. Information extraction using hidden markov models. Master's thesis, UC San Diego.

[McCallum et al.2000] A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum entropy Markov models for informaion extraction and segmentation. In *Proceedings of ICML-2000*.

[McCallum2003] Andrew McCallum. 2003. Efficiently inducing features of conditional random fields. In *Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03)*.

[Peng and McCallum2004] F. Peng and A. McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics*.

[Peshkin and Pfeffer2003] L. Peshkin and A. Pfeffer. 2003. Bayesian information extraction network. In *Proceedings of the Eighteenth International Joint Conf. on Artificial Intelligence*.

[Rabiner1989] L. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77(2).

A DOM Tree Alignment Model for Mining Parallel Data from the Web

Lei Shi[1], Cheng Niu[1], Ming Zhou[1], and Jianfeng Gao[2]
[1]Microsoft Research Asia, 5F Sigma Center, 49 Zhichun Road, Beijing 10080, P. R. China
[2]Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA
{leishi,chengniu,mingzhou,jfgao}@microsoft.com

Abstract

This paper presents a new web mining scheme for parallel data acquisition. Based on the Document Object Model (DOM), a web page is represented as a DOM tree. Then a DOM tree alignment model is proposed to identify the translationally equivalent texts and hyperlinks between two parallel DOM trees. By tracing the identified parallel hyperlinks, parallel web documents are recursively mined. Compared with previous mining schemes, the benchmarks show that this new mining scheme improves the mining coverage, reduces mining bandwidth, and enhances the quality of mined parallel sentences.

1 Introduction

Parallel bilingual corpora are critical resources for statistical machine translation (Brown 1993), and cross-lingual information retrieval (Nie 1999). Additionally, parallel corpora have been exploited for various monolingual natural language processing (NLP) tasks, such as word-sense disambiguation (Ng 2003) and paraphrase acquisition (Callison 2005).

However, large scale parallel corpora are not readily available for most language pairs. Even where resources are available, such as for English-French, the data are usually restricted to government documents (*e.g.,* the Hansard corpus, which consists of French-English translations of debates in the Canadian parliament) or newswire texts. The "governmentese" that characterizes these document collections cannot be used on its own to train data-driven machine translation systems for a range of domains and language pairs.

With a sharply increasing number of bilingual web sites, web mining for parallel data becomes a promising solution to this knowledge acquisition problem. In an effort to estimate the amount of bilingual data on the web, (Ma and Liberman 1999) surveyed web pages in the de (German web site) domain, showing that of 150,000 websites in the .de domain, 10% are German-English bilingual. Based on such observations, some web mining systems have been developed to automatically obtain parallel corpora from the web (Nie *et al* 1999; Ma and Liberman 1999; Chen, Chau and Yeh 2004; Resnik and Smith 2003； Zhang *et al* 2006). These systems mine parallel web documents within bilingual web sites, exploiting the fact that URLs of many parallel web pages are named with apparent patterns to facilitate website maintenance. Hence given a bilingual website, the mining systems use pre-defined URL patterns to discover candidate parallel documents within the site. Then content-based features will be used to verify the translational equivalence of the candidate pairs.

However, due to the diversity of web page styles and website maintenance mechanisms, bilingual websites use varied naming schemes for parallel documents. For example, the United Nation's website, which contains thousands of parallel pages, simply names the majority of its web pages with some computer generated ad-hoc URLs. Such a website then cannot be mined by the URL pattern-based mining scheme. To further improve the coverage of web mining, other patterns associated with translational parallelism are called for.

Besides, URL pattern-based mining may raise concerns on high bandwidth cost and slow download speed. Based on descriptions of (Nie *et al* 1999; Ma and Liberman 1999; Chen, Chau and Yeh 2004), the mining process requires a full host crawling to collect URLs before using URL patterns to discover the parallel documents. Since in many bilingual web sites, parallel documents are much sparser than comparable documents, a significant portion of internet bandwidth is wasted on downloading web pages without translational counterparts.

Furthermore, there is a lack of discussion on the quality of mined data. To support machine translation, parallel sentences should be extracted from the mined parallel documents. However, current sentence alignment models, (Brown *et al* 1991; Gale & Church 1991; Wu 1994; Chen

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 489–496,
Sydney, July 2006. ©2006 Association for Computational Linguistics

1993; Zhao and Vogel, 2002; *etc.*) are targeted on traditional textual documents. Due to the noisy nature of the web documents, parallel web pages may consist of non-translational content and many out-of-vocabulary words, both of which reduce sentence alignment accuracy. To improve sentence alignment performance on the web data, the similarity of the HTML tag structures between the parallel web documents should be leveraged properly in the sentence alignment model.

In order to improve the quality of mined data and increase the mining coverage and speed, this paper proposes a new web parallel data mining scheme. Given a pair of parallel web pages as seeds, the Document Object Model[1] (DOM) is used to represent the web pages as a pair of DOM trees. Then a stochastic DOM tree alignment model is used to align translationally equivalent content, including both textual chunks and hyperlinks, between the DOM tree pairs. The parallel hyperlinks discovered are regarded as anchors to new parallel data. This makes the mining scheme an iterative process.

The new mining scheme has three advantages: (i) Mining coverage is increased. *Parallel hyperlinks referring to parallel web page* is a general and reliable pattern for parallel data mining. Many bilingual websites not supporting URL pattern-based mining scheme support this new mining scheme. Our mining experiment shows that, using the new web mining scheme, the web mining throughput is increased by 32%; (ii) The quality of the mined data is improved. By leveraging the web pages' HTML structures, the sentence aligner supported by the DOM tree alignment model outperforms conventional ones by 7% in both precision and recall; (iii) The bandwidth cost is reduced by restricting web page downloads to the links that are very likely to be parallel.

The rest of the paper is organized as follows: In the next section, we introduce the related work. In Section 3, a new web parallel data mining scheme is presented. Three component technologies, the DOM tree alignment model, the sentence aligner, and the candidate parallel page verification model are presented in Section 4, 5, and 6. Section 7 presents experiments and benchmarks. The paper is finally concluded in Section 8.

2 Related Work

The parallel data available on the web have been an important knowledge source for machine translation. For example, *Hong Kong Laws*, an English-Chinese Parallel corpus released by Linguistic Data Consortium (LDC) is downloaded from the *Department of Justice of the Hong Kong Special Administrative Region* website.

Recently, web mining systems have been built to automatically acquire parallel data from the web. Exemplary systems include *PTMiner* (Nie et al 1999), STRAND (Resnik and Smith, 2003), BITS (Ma and Liberman, 1999), and PTI (Chen, Chau and Yeh, 2004). Given a bilingual website, these systems identify candidate parallel documents using pre-defined URL patterns. Then content-based features are employed for candidate verification. Particularly, HTML tag similarities have been exploited to verify parallelism between pages. But it is done by simplifying HTML tags as a string sequence instead of a hierarchical DOM tree. Tens of thousands parallel documents have been acquired with accuracy over 90%.

To support machine translation, parallel sentence pairs should be extracted from the parallel web documents. A number of techniques for aligning sentences in parallel corpora have been proposed. (Gale & Church 1991; Brown *et al.* 1991; Wu 1994) used sentence length as the basic feature for alignment. (Kay & Roscheisen 1993; and Chen 1993) used lexical information for sentence alignment. Models combining length and lexicon information were proposed in (Zhao and Vogel, 2002; Moore 2002). Signal processing techniques is also employed in sentence alignment by (Church 1993; Fung & McKeown 1994). Recently, much research attention has been paid to aligning sentences in comparable documents (Utiyama et al 2003, Munteanu et al 2004).

The DOM tree alignment model is the key technique of our mining approach. Although, to our knowledge, this is the first work discussing DOM tree alignments, there is substantial research focusing on syntactic tree alignment model for machine translation. For example, (Wu 1997; Alshawi, Bangalore, and Douglas, 2000; Yamada and Knight, 2001) have studied synchronous context free grammar. This formalism requires isomorphic syntax trees for the source sentence and its translation. (Shieber and Schabes 1990) presents a synchronous tree adjoining grammar (STAG) which is able to align two syn-

[1] See http://www.w3.org/DOM/

tactic trees at the linguistic minimal units. The synchronous tree substitution grammar (STSG) presented in (Hajic etc. 2004) is a simplified version of STAG which allows tree substitution operation, but prohibits the operation of tree adjunction.

3 A New Parallel Data Mining Scheme Supported by DOM Tree Alignment

Our new web parallel data mining scheme consists of the following steps:

(1) Given a web site, the root page and web pages directly linked from the root page are downloaded. Then for each of the downloaded web page, all of its anchor texts (*i.e.* the hyperlinked words on a web page) are compared with a list of predefined strings known to reflect translational equivalence among web pages (Nie *et al* 1999). Examples of such predefined trigger strings include: (i) trigger words for English translation {*English, English Version, 英文, 英文版, etc.*}; and (ii) trigger words for Chinese translation {*Chinese, Chinese Version, Simplified Chinese, Traditional Chinese, 中文, 中文版，, etc.*}. If both categories of trigger words are found, the web site is considered bilingual, and every web page pair are sent to Step 2 for parallelism verification.

(2) Given a pair of the plausible parallel web pages, a verification module is called to determine if the page pair is truly translationally equivalent.

(3) For each verified pair of parallel web pages, a DOM tree alignment model is called to extract parallel text chunks and hyperlinks.

(4) Sentence alignment is performed on each pair of the parallel text chunks, and the resulting parallel sentences are saved in an output file.

(5) For each pair of parallel hyperlinks, the corresponding pair of web pages is downloaded, and then goes to Step 2 for parallelism verification. If no more parallel hyperlinks are found, stop the mining process.

Our new mining scheme is iterative in nature. It fully exploits the information contained in the parallel data and effectively uses it to pinpoint the location holding more parallel data. This approach is based on our observation that parallel pages share similar structures holding parallel content, and parallel hyperlinks refer to new parallel pages.

By exploiting both the HTML tag similarity and the content-based translational equivalences, the DOM tree alignment model extracts parallel text chunks. Working on the parallel text chunks instead of the text of the whole web page, the sentence alignment accuracy can be improved by a large margin.

In the next three sections, three component techniques, the DOM tree alignment model, sentence alignment model, and candidate web page pair verification model are introduced.

4 DOM Tree Alignment Model

The Document Object Model (DOM) is an application programming interface for valid HTML documents. Using DOM, the logical structure of a HTML document is represented as a tree where each node belongs to some pre-defined node types (*e.g. Document, DocumentType, Element, Text, Comment, ProcessingInstruction etc.*). Among all these types of nodes, the nodes most relevant to our purpose are *Element* nodes (corresponding to the HTML tags) and *Text* nodes (corresponding to the texts). To simplify the description of the alignment model, minor modifications of the standard DOM tree are made: (i) Only the *Element* nodes and *Text* nodes are kept in our document tree model. (ii) The *ALT* attribute is represented as *Text* node in our document tree model. The *ALT* text are textual alternative when images cannot be displayed, hence is helpful to align images and hyperlinks. (iii) the *Text* node (which must be a leaf) and its parent *Element* node are combined into one node in order to concise the representation of the alignment model. The above three modifications are exemplified in Fig. 1.

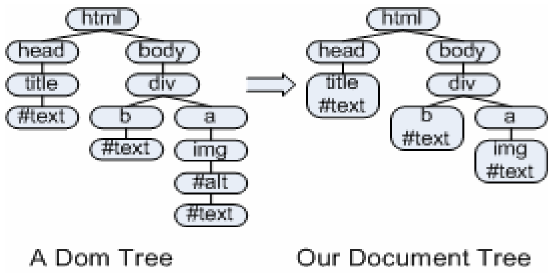


Fig. 1 Difference between Standard DOM and Our Document Tree

Despite these minor differences, our document tree is still referred as DOM tree throughout this paper.

4.1 DOM Tree Alignment

Similar to STSG, our DOM tree alignment model supports node deletion, insertion and substitution. Besides, both STSG and our DOM tree alignment model define the alignment as a tree hierarchical invariance process, *i.e.* if node *A* is aligned with node *B*, then the children of *A* are either deleted or aligned with the children of *B*.

But two major differences exist between STSG and our DOM tree alignment model: (i) Our DOM tree alignment model requires the alignment a sequential order invariant process, *i.e.* if node *A* is aligned with node *B*, then the sibling nodes following *A* have to be either deleted or aligned with the sibling nodes following *B*. (ii) (Hajic etc. 2004) presents STSG in the context of language generation, while we search for the best alignment on the condition that both trees are given.

To facilitate the presentation of the tree alignment model, the following symbols are introduced: given a HTML document D, T^{D} refers to the corresponding DOM tree; $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}$ refers to the i^{th} node of T^{D} (here the index of the node is in the breadth-first order), and $\mathrm{T}_{\mathrm{i}}^{\mathrm{D}}$ refers to the sub-tree rooted at $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}$, so $\mathrm{N}_{1}^{\mathrm{D}}$ refers to the root of T^{D}, and $\mathrm{T}_{1}^{\mathrm{D}} = T^{D}$; $\mathrm{T}_{[\mathrm{i},\mathrm{j}]}^{\mathrm{D}}$ refers to the forest consisting of the sub-trees rooted at nodes from $\mathrm{T}_{\mathrm{i}}^{\mathrm{D}}$ to $\mathrm{T}_{\mathrm{j}}^{\mathrm{D}}$. $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}.t$ refers to the text of node $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}$; $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}.l$ refers to the HTML tag of the node $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}$; $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}.C_j$ refers to the j^{th} child of the node $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}$; $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}.C_{[m,n]}$ refers to the consecutive sequence of $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}$'s children nodes from $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}.C_m$ to $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}.C_n$; the sub-tree rooted at $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}.C_j$ is represented as $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}.TC_j$ and the forest rooted at $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}.C_{[m,n]}$ is represented as $\mathrm{N}_{\mathrm{i}}^{\mathrm{D}}.TC_{[m,n]}$. Finally *NULL* refers to the empty node introduced for node deletion.

To accommodate the hierarchical structure of the DOM tree, two different translation probabilities are defined:

$\Pr\left(T_m^F \middle| T_i^E\right)$: probability of translating sub-tree T_i^E into sub-tree T_m^F ;

$\Pr\left(N_m^F \middle| N_i^E\right)$: probability of translating node N_i^E into N_m^F .

Besides, $\Pr\left(T_{[m,n]}^F \middle| T_{[i,j]}^E, A\right)$ represents the probability of translating the forest $T_{[i,j]}^E$ into $T_{[m,n]}^F$ based on the alignment *A*. The tree alignment *A* is defined as a mapping from target nodes onto source nodes or the null node.

Given two HTML documents F (in French) and E (in English), the tree alignment task is defined as searching for *A* which maximizes the following probability:

$$\Pr\left(A \middle| T^F, T^E\right) \propto \Pr\left(T^F \middle| T^E, A\right) \Pr\left(A \middle| T^E\right) \qquad (1)$$

where $\Pr\left(A \middle| T^E\right)$ represents the prior knowledge of the alignment configurations.

By introducing p_d which refers to the probability of a source or target node deletion occurring in an alignment configuration, the alignment prior $\Pr\left(A \middle| T^E\right)$ is assumed as the following binominal distribution:

$$\Pr\left(A \middle| T^E\right) \propto \left(1 - p_d\right)^L p_d{}^M$$

where *L* is the count of non-empty alignments in A, and *M* is the count of source and target node deletions in *A*.

As to $\Pr\left(T^F \middle| T^E, A\right)$, we can estimate as $\Pr\left(T^F \middle| T^E, A\right) = \Pr\left(T_1^F \middle| T_1^E, A\right)$, and $Pr\left(T_l^F \middle| T_i^E, A\right)$ can be calculated recursively depending on the alignment configuration of A:

(1) If N_l^F is aligned with N_i^E, and the children of N_l^F are aligned with the children of N_i^E, then we have

$$\Pr\left(T_l^F \middle| T_i^E, A\right)$$
$$= \Pr\left(N_l^F \middle| N_i^E\right) \Pr\left(N_l^F.TC_{[1,K]} \middle| N_i^E.TC_{\left[1,K'\right]}, A\right)$$

where K and K' are degree of N_l^F and N_i^E.

(2) If N_l^F is deleted, and the children of N_l^F is aligned with T_i^E, then we have

$$\Pr\left(T_l^F \middle| T_i^E, A\right) = \Pr\left(N_l^F \middle| NULL\right) \Pr\left(N_l^F.TC_{[1,K]} \middle| T_i^E, A\right)$$

where K is the degree of N_l^F

(3) If N_i^E is deleted, and N_l^F is aligned with the children of N_i^E, then

$$\Pr\left(T_l^F \middle| T_i^E, A\right) = \Pr\left(T_l^F \middle| T_i^E.TC_{[1,K]}, A\right)$$

where K is the degree of N_i^E .

To complete the alignment model, $Pr\left(T_{[m,n]}^F \middle| T_{[i,j]}^E, A\right)$ is to be estimated. As mentioned before, only the alignment configurations with unchanged node sequential order are considered as valid. So, $Pr\left(T_{[m,n]}^F \middle| T_{[i,j]}^E, A\right)$ is estimated recursively according to the following five alignment configurations of A:

(4) If T_m^F is aligned with T_i^E, and $T_{[m+1,n]}^F$ is

aligned with $T^E_{[i+1,j]}$, then

$$\Pr\left(T^F_{[m,n]} \middle| T^E_{[i,j]}, A\right) = \Pr\left(N^F_m \middle| N^E_i\right) \Pr\left(T^F_{[m+1,n]} \middle| T^E_{[i+1,j]}, A\right)$$

(5) If T^F_m is deleted, and $T^F_{[m+1,n]}$ is aligned with $T^E_{[i,j]}$, then

$$\Pr\left(T^F_{[m,n]} \middle| T^E_{[i,j]}, A\right) = \Pr\left(N^F_m \middle| NULL\right) \Pr\left(T^F_{[m+1,n]} \middle| T^E_{[i,j]}, A\right)$$

(6) If T^E_i is deleted, and $T^F_{[m,n]}$ is aligned with $T^E_{[i+1,j]}$, then

$$\Pr\left(T^F_{[m,n]} \middle| T^E_{[i,j]}, A\right) = \Pr\left(T^F_{[m,n]} \middle| T^E_{[i+1,j]}, A\right)$$

(7) If N^F_m is deleted, and N^F_m's children $N^F_m.C_{[1,K]}$ is combined with $T^F_{[m+1,n]}$ to aligned with $T^E_{[i,j]}$, then

$$\Pr\left(T^F_{[m,n]} \middle| T^E_{[i,j]}, A\right)$$
$$= \Pr\left(N^F_m \middle| NULL\right) \Pr\left(N^F_m.TC_{[1,K]} T^F_{[m+1,n]} \middle| T^E_{[i,j]}, A\right)$$

where K is the degree of N^F_m.

(8) N^E_i is deleted, and N^E_i's children $N^E_i.C_{[1,K]}$ is combined with $T^E_{[i+1,j]}$ to be aligned with $T^F_{[m,n]}$, then

$$\Pr\left(T^F_{[m,n]} \middle| T^E_{[i,j]}, A\right) = \Pr\left(T^F_{[m,n]} \middle| N^E_i.TC_{[1,K]} T^E_{[i+1,j]}, A\right)$$

where K is the degree of N^E_i.

Finally, the node translation probability is modeled as $\Pr\left(N^F_l \middle| N^E_j\right) \approx \Pr\left(N^F_l.l \middle| N^E_i.l\right) \Pr\left(N^F_l.t \middle| N^E_i.t\right)$. And the text translation probability $\Pr\left(t^F \middle| t^E\right)$ is model using IBM model I (Brown *et al* 1993).

4.2 Parameter Estimation Using Expectation-Maximization

Our tree alignment model involves three categories of parameters: the text translation probability $\Pr\left(t^F \middle| t^E\right)$, tag mapping probability $\Pr\left(l \middle| l'\right)$, and node deletion probability p_d.

Conventional parallel data released by LDC are used to train IBM model I for estimating the text translation probability $\Pr\left(t^F \middle| t^E\right)$.

One way to estimate $\Pr\left(l \middle| l'\right)$ and p_d is to manually align nodes between parallel DOM trees, and use them as training corpora for maximum likelihood estimation. However, this is a very time-consuming and error-prone procedure. In this paper, the inside outside algorithm presented in (Lari and Young, 1990) is extended to train parameters $\Pr\left(l \middle| l'\right)$ and p_d by optimally fitting the existing parallel DOM trees.

4.3 Dynamic Programming for Decoding

It is observed that if two trees are optimally aligned, the alignment of their sub-trees must be optimal as well. In the decoding process, dynamic programming techniques can be applied to find the optimal tree alignment using that of the sub-trees in a bottom up manner. The following is the pseudo-code of the decoding algorithm:

```
For i=$|T^F|$ to 1  (bottom-up) {
    For j=$|T^E|$ to 1 (bottom-up) {
        derive the best alignments among $T^F_i.TC_{[1,K_i]}$ and $T^E_j.TC_{[1,K_j]}$, and then compute the best alignment between $N^F_i$ and $N^E_j$.
```

where $|T^F|$ and $|T^E|$ are number of nodes in T^F and T^E; K_i and K_j are the degrees of N^F_i and N^E_j. The time complexity of the decoding algorithm is $\mathrm{O}(|\mathrm{T}_\mathrm{F}| \times |T_E| \times (\mathrm{degree}(T^F) + \mathrm{degree}(T^E))^2)$, where the degree of a tree is defined as the largest degree of its nodes.

5 Aligning Sentences Using Tree Alignment Model

To exploit the HTML structure similarities between parallel web documents, a cascaded approach is used in our sentence aligner implementation.

First, text chunks associated with DOM tree nodes are aligned using the DOM tree alignment model. Then for each pair of parallel text chunks, the sentence aligner described in (Zhao et al 2002), which combines IBM model I and the length model of (Gale & Church 1991) under a maximum likelihood criterion, is used to align parallel sentences.

6 Web Document Pair Verification Model

To verify whether a candidate web document pair is truly parallel, a binary maximum entropy based classifier is used.

Following (Nie *et al* 1999) and (Resnik and Smith, 2003), three features are used: (i) file length ratio; (ii) HTML tag similarity; (iii) sentence alignment score.

The HTML tag similarity feature is computed as follows: all of the HTML tags of a given web page are extracted, and concatenated as a string. Then, a minimum edit distance between the two tag strings associated with the candidate pair is computed, and the HMTL tag similarity score is defined as the ratio of match operation number to the total operation number.

The sentence alignment score is defined as the ratio of the number of aligned sentences and the total number of sentences in both files.

Using these three features, the maximum entropy model is trained on 1,000 pairs of web pages manually labeled as parallel or non-parallel. The Iterative Scaling algorithm (Pietra, Pietra and Lafferty 1995) is used for the training.

7 Experimental Results

The DOM tree alignment based mining system is used to acquire English-Chinese parallel data from the web. The mining procedure is initiated by acquiring Chinese website list.

We have downloaded about 300,000 URLs of Chinese websites from the web directories at *cn.yahoo.com, hk.yahoo.com and tw.yahoo.com.* And each website is sent to the mining system for English-Chinese parallel data acquisition. To ensure that the whole mining experiment to be finished in schedule, we stipulate that it takes at most 10 hours on mining each website. Totally 11,000 English-Chinese websites are discovered, from which 63,214 pairs of English-Chinese parallel web documents are mined. After sentence alignment, totally 1,069,423 pairs of English-Chinese parallel sentences are extracted.

In order to compare the system performance, 100 English-Chinese bilingual websites are also mined using the URL pattern based mining scheme. Following (Nie *et al* 1999; Ma and Liberman 1999; Chen, Chau and Yeh 2004), the URL pattern-based mining consists of three steps: (i) host crawling for URL collection; (ii) candidate pair identification by pre-defined URL pattern matching; (iii) candidate pair verification.

Based on these mining results, the quality of the mined data, the mining coverage and mining efficiency are measured.

First, we benchmarked the precision of the mined parallel documents. 3,000 pairs of English-Chinese candidate documents are randomly selected from the output of each mining system, and are reviewed by human annotators. The document level precision is shown in Table 1.

	URL pattern	DOM Tree Alignment
Precision	93.5%	97.2%

Table 1: Precision of Mined Parallel Documents

The document-level mining precision solely depends on the candidate document pair verification module. The verification modules of both mining systems use the same features, and the only difference is that in the new mining system the sentence alignment score is computed with DOM tree alignment support. So the 3.7% improvement in document-level precision indirectly confirms the enhancement of sentence alignment.

Secondly, the accuracy of sentence alignment model is benchmarked as follows: 150 English-Chinese parallel document pairs are randomly taken from our mining results. All parallel sentence pairs in these document pairs are manually annotated by two annotators with cross-validation. We have compared sentence alignment accuracy with and without DOM tree alignment support. In case of no tree alignment support, all the texts in the web pages are extracted and sent to sentence aligner for alignment. The benchmarks are shown in Table 2.

Alignment Method	**Number Right**	**Number Wrong**	**Number Missed**	**Precision**	**Recall**
Eng-Chi (no DOM tree)	2172	285	563	86.9%	79.4%
Eng-Chi (with DOM tree)	2369	156	366	93.4%	86.6%

Table 2: sentence alignment accuracy

Table 2 shows that with DOM tree alignment support, the sentence alignment accuracy is greatly improved by 7% in both precision and recall. We also observed that the recall is lower than precision. This is because web pages tend to contain many short sentences (one or two words only) whose alignment is hard to identify due to the lack of content information.

Although Table 2 benchmarks the accuracy of sentence aligner, but the quality of the final sentence pair outputs depend on many other modules as well, *e.g.* the document level parallelism verification, sentence breaker, Chinese word breaker, etc. To further measure the quality of the mined data, 2,000 sentence pairs are randomly picked from the final output, and are manually classified into three categories: (i) exact parallel, (ii) roughly parallel: two parallel sentences involving missing words or erroneous additions; (iii) not parallel. Two annotators are assigned for this task with cross-validation. As is shown in Table 3, 93.5% of output sentence pairs are either exact or roughly parallel.

Corpus	Exact Parallel	Roughly Parallel	Not Parallel
Mined	1703	167	130

Table 3 Quality of Mined Parallel Sentences

As we know, the absolute value of mining system recall is hard to estimate because it is impractical to evaluate all the parallel data held by a bilingual website. Instead, we compare mining coverage and efficiency between the two systems. 100 English-Chinese bilingual website are mined by both of the system. And the mining efficiency comparison is reported in Table 4.

Mining System	Parallel Page Pairs found & verified	# of page downloads	# of downloads per pair
URL pattern-based Mining	4383	84942	19.38
DOM Tree Alignment-based Mining	5785	13074	2.26

Table 4. Mining Efficiency Comparison on 100 Bilingual Websites

Although it downloads less data, the DOM tree based mining scheme increases the parallel data acquisition throughput by 32%. Furthermore, the ratio of downloaded page count per parallel pair is 2.26, which means the bandwidth usage is almost optimal.

Another interesting topic is the complementarities between both mining systems. As reported in Table (5), 1797 pairs of parallel documents mined by the new scheme is not covered by the URL pattern-based scheme. So if both systems are used, the throughput can be further increased by 41%.

# of Parallel Page Pairs Mined by Both Systems	# of Parallel Page Pairs Mined by URL Patterns only	# of Parallel Page Pairs Mined by Tree Alignment only
3988	395	1797

Table 5. Mining Results Complementarities on 100 Bilingual Website

8 Discussion and Conclusion

Mining parallel data from web is a promising method to overcome the *knowledge bottleneck* faced by machine translation. To build a practical mining system, three research issues should be fully studied: (i) the quality of mined data, (ii) the mining coverage, and (iii) the mining speed. Exploiting DOM tree similarities helps in all the three issues.

Motivated by this observation, this paper presents a new web mining scheme for parallel data acquisition. A DOM tree alignment model is proposed to identify translationally equivalent text chunks and hyperlinks between two HTML documents. Parallel hyperlinks are used to pinpoint new parallel data, and make parallel data mining a recursive process. Parallel text chunks are fed into sentence aligner to extract parallel sentences.

Benchmarks show that sentence aligner supported by DOM tree alignment achieves performance enhancement by 7% in both precision and recall. Besides, the new mining scheme reduce the bandwidth cost by 8~9 times on average compared with the URL pattern-based mining scheme. In addition, the new mining scheme is more general and reliable, and is able to mine more data. Using the new mining scheme alone, the mining throughput is increased by 32%, and when combined with URL pattern-based scheme, the mining throughput is increased by 41%.

References

Alshawi, H., S. Bangalore, and S. Douglas. 2000. Learning Dependency Translation Models as Collections of Finite State Head Transducers. *Computational Linguistics*, 26(1).

Brown, P. F., J. C. Lai and R. L. Mercer. 1991. Aligning Sentences in Parallel Corpora. In *Proceedings of 29th Annual Meeting of the Association for Computational Linguistics.*

Brown, P. E., S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics,* V19(2).

Callison-Burch, C. and C. Bannard. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of 43th Annual Meeting of the Association for Computational Linguistics.*

Chen, J., R. Chau, and C.-H. Yeh. 1991. Discovering Parallel Text from the World Wide Web. In *Proceedings of the second workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalization.*

Chen, S. 1993. Aligning Sentences in Bilingual Corpora Using Lexical Information. In *Proceedings of 31st Annual Meeting of the Association for Computational Linguistics.*

Church, K. W. 1993. Char_align: A Program for Aligning Parallel Texts at the Character Level. In

Proceedings of 31st Annual Meeting of the Association for Computational Linguistics.

Fung, P. and K. Mckeown. 1994. Aligning Noisy Parallel Corpora across Language Groups: Word Pair Feature Matching by Dynamic Time Warping. In *Proceedings of the First Conference of the Association for Machine Translation in the Americas.*

Gale W. A. and K. Church. 1991. A Program for Aligning Sentences in Parallel Corpora. In *Proceedings of 29th Annual Meeting of the Association for Computational Linguistics.*

Hajic J., *et al.* 2004. Final Report: Natural Language Generation in the Context of Machine Translation.

Kay M. and M. Roscheisen. 1993. Text-Translation Alignment. *Computational Linguistics*, 19(1).

Lari K. and S. J. Young. 1990. The Estimation of Stochastic Context Free Grammars using the Inside-Outside Algorithm. *Computer Speech and Language,* 4:35—56, 1990.

Ma, X. and M. Liberman. 1999. Bits: A Method for Bilingual Text Search over the Web. In *Proceedings of Machine Translation Summit VII.*

Ng, H. T., B. Wang, and Y. S. Chan. 2003. Exploiting Parallel Texts for Word Sense Disambiguation: An Empirical Study. In *Proceedings of 41st Annual Meeting of the Association for Computational Linguistics.*

Nie, J. Y., M. S. P. Isabelle, and R. Durand. 1999. Cross-language Information Retrieval based on Parallel Texts and Automatic Mining of Parallel Texts from the Web. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development.*

Moore, R. C. 2002. Fast and Accurate Sentence Alignment of Bilingual Corpora. In *Proceedings of 5th Conference of the Association for Machine Translation in the Americas*.

Munteanu D. S, A. Fraser, and D. Marcu. D., 2002. Improved Machine Translation Performance via Parallel Sentence Extraction from Comparable Corpora. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004.*

Pietra, S. D., V. D. Pietra, and J. Lafferty. 1995. Inducing Features Of Random Fields. In *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Resnik, P. and N. A. Smith. 2003. The Web as a Parallel Corpus. *Computational Linguistics*, 29(3)

Shieber, S. M. and Y. Schabes. 1990. Synchronous tree-adjoining grammars. In *Proceedings of the 13th International Conference on Computational linguistics.*

Utiyama, M. and H. Isahara 2003. Reliable Measures for Aligning Japanese-English News Articles and Sentences. In *Proceedings of 41st Annual Meeting of the Association for Computational Linguistics.*ACL 2003.

Wu, D. 1994. Aligning a parallel English-Chinese corpus statistically with lexical criterias. In *Proceedings of of 32nd Annual Meeting of the Association for Computational Linguistics.*

Wu, D. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3).

Yamada K. and K. Knight. 2001. A Syntax Based Statistical Translation Model. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*.

Zhao B. and S. Vogel. 2002. Adaptive Parallel Sentences Mining From Web Bilingual News Collection. In *2002 IEEE International Conference on Data Mining.*

Zhang, Y., K. Wu, J. Gao, and Phil Vines. 2006. Automatic Acquisition of Chinese-English Parallel Corpus from the Web. In *Proceedings of 28th European Conference on Information Retrieval.*

QuestionBank: Creating a Corpus of Parse-Annotated Questions

John Judge[1], **Aoife Cahill**[1], **and Josef van Genabith**[1,2]
[1]National Centre for Language Technology and School of Computing,
Dublin City University, Dublin, Ireland
[2]IBM Dublin Center for Advanced Studies,
IBM Dublin, Ireland
{jjudge,acahill,josef}@computing.dcu.ie

Abstract

This paper describes the development of QuestionBank, a corpus of 4000 parse-annotated questions for (i) use in training parsers employed in QA, and (ii) evaluation of question parsing. We present a series of experiments to investigate the effectiveness of QuestionBank as both an exclusive and supplementary training resource for a state-of-the-art parser in parsing both question and non-question test sets. We introduce a new method for recovering empty nodes and their antecedents (capturing long distance dependencies) from parser output in CFG trees using LFG f-structure reentrancies. Our main findings are (i) using QuestionBank training data improves parser performance to 89.75% labelled bracketing f-score, an increase of almost 11% over the baseline; (ii) back-testing experiments on non-question data (Penn-II WSJ Section 23) shows that the retrained parser does not suffer a performance drop on non-question material; (iii) ablation experiments show that the size of training material provided by QuestionBank is sufficient to achieve optimal results; (iv) our method for recovering empty nodes captures long distance dependencies in questions from the ATIS corpus with high precision (96.82%) and low recall (39.38%). In summary, QuestionBank provides a useful new resource in parser-based QA research.

1 Introduction

Parse-annotated corpora (treebanks) are crucial for developing machine learning and statistics-based parsing resources for a given language or task. Large treebanks are available for major languages, however these are often based on a specific text type or genre, e.g. financial newspaper text (the Penn-II Treebank (Marcus et al., 1993)). This can limit the applicability of grammatical resources induced from treebanks in that such resources underperform when used on a different type of text or for a specific task.

In this paper we present work on creating QuestionBank, a treebank of parse-annotated questions, which can be used as a supplementary training resource to allow parsers to accurately parse questions (as well as other text). Alternatively, the resource can be used as a stand-alone training corpus to train a parser specifically for questions. Either scenario will be useful in training parsers for use in question answering (QA) tasks, and it also provides a suitable resource to evaluate the accuracy of these parsers on questions.

We use a semi-automatic "bootstrapping" method to create the question treebank from raw text. We show that a parser trained on the question treebank alone can accurately parse questions. Training on a combined corpus consisting of the question treebank and an established training set (Sections 02-21 of the Penn-II Treebank), the parser gives state-of-the-art performance on both questions and a non-question test set (Section 23 of the Penn-II Treebank).

Section 2 describes background work and motivation for the research presented in this paper. Section 3 describes the data we used to create the corpus. In Section 4 we describe the semi-automatic method to "bootstrap" the question corpus, discuss some interesting and problematic phenomena, and show how the manual vs. automatic workload distribution changed as work progressed. Two sets of experiments using our new question corpus are presented in Section 5. In Section 6 we introduce a new method for recovering empty nodes and their antecedents using Lexical Functional Grammar (LFG) f-structure reen-

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 497–504,
Sydney, July 2006. ©2006 Association for Computational Linguistics

trancies. Section 7 concludes and outlines future work.

2 Background and Motivation

High quality probabilistic, treebank-based parsing resources can be rapidly induced from appropriate treebank material. However, treebank- and machine learning-based grammatical resources reflect the characteristics of the training data. They generally underperform on test data substantially different from the training data.

Previous work on parser performance and domain variation by Gildea (2001) showed that by training a parser on the Penn-II Treebank and testing on the Brown corpus, parser accuracy drops by 5.7% compared to parsing the Wall Street Journal (WSJ) based Penn-II Treebank Section 23. This shows a negative effect on parser performance even when the test data is *not radically* different from the training data (both the Penn II and Brown corpora consist primarily of written texts of American English, the main difference is the considerably more varied nature of the text in the Brown corpus). Gildea also shows how to resolve this problem by adding appropriate data to the training corpus, but notes that a large amount of additional data has little impact if it is not matched to the test material.

Work on more *radical* domain variance and on adapting treebank-induced LFG resources to analyse ATIS (Hemphill et al., 1990) question material is described in Judge et al. (2005). The research established that even a small amount of additional training data can give a substantial improvement in question analysis in terms of both CFG parse accuracy and LFG grammatical functional analysis, with no significant negative effects on non-question analysis. Judge et al. (2005) suggest, however, that further improvements are possible given a larger question training corpus.

Clark et al. (2004) worked specifically with question parsing to generate dependencies for QA with Penn-II treebank-based Combinatory Categorial Grammars (CCG's). They use "what" questions taken from the TREC QA datasets as the basis for a What-Question corpus with CCG annotation.

3 Data Sources

The raw question data for QuestionBank comes from two sources, the TREC 8-11 QA track test sets[1], and a question classifier training set produced by the Cognitive Computation Group (CCG[2]) at the University of Illinois at Urbana-Champaign.[3] We use equal amounts of data from each source so as not to bias the corpus to either data source.

3.1 TREC Questions

The TREC evaluations have become the standard evaluation for QA systems. Their test sets consist primarily of fact seeking questions with some imperative statements which request information, e.g. "List the names of cell phone manufacturers." We included 2000 TREC questions in the raw data from which we created the question treebank. These 2000 questions consist of the test questions for the first three years of the TREC QA track (1893 questions) and 107 questions from the 2003 TREC test set.

3.2 CCG Group Questions

The CCG provide a number of resources for developing QA systems. One of these resources is a set of 5500 questions and their answer types for use in training question classifiers. The 5500 questions were stripped of answer type annotation, duplicated TREC questions were removed and 2000 questions were used for the question treebank.

The CCG 5500 questions come from a number of sources (Li and Roth, 2002) and some of these questions contain minor grammatical mistakes so that, in essence, this corpus is more representative of genuine questions that would be put to a working QA system. A number of changes in tokenisation were corrected (eg. separating contractions), but the minor grammatical errors were left unchanged because we believe that it is necessary for a parser for question analysis to be able to cope with this sort of data if it is to be used in a working QA system.

4 Creating the Treebank

4.1 Bootstrapping a Question Treebank

The algorithm used to generate the question treebank is an iterative process of parsing, manual correction, retraining, and parsing.

[1] http://trec.nist.gov/data/qa.html

[2] Note that the acronym CCG here refers to Cognitive Computation Group, rather than Combinatory Categorial Grammar mentioned in Section 2.

[3] http://l2r.cs.uiuc.edu/ cogcomp/tools.php

Algorithm 1 Induce a parse-annotated treebank from raw data

```
repeat
    Parse a new section of raw data
    Manually correct errors in the parser output
    Add the corrected data to the training set
    Extract a new grammar for the parser
until All the data has been processed
```

Algorithm 1 summarises the bootstrapping algorithm. A section of raw data is parsed. The parser output is then manually corrected, and added to the parser's training corpus. A new grammar is then extracted, and the next section of raw data is parsed. This process continues until all the data has been parsed and hand corrected.

4.2 Parser

The parser used to process the raw questions prior to manual correction was that of Bikel (2002)[4], a retrainable emulation of Collins (1999) model 2 parser. Bikel's parser is a history-based parser which uses a lexicalised generative model to parse sentences. We used WSJ Sections 02-21 of the Penn-II Treebank to train the parser for the first iteration of the algorithm. The training corpus for subsequent iterations consisted of the WSJ material and increasing amounts of processed questions.

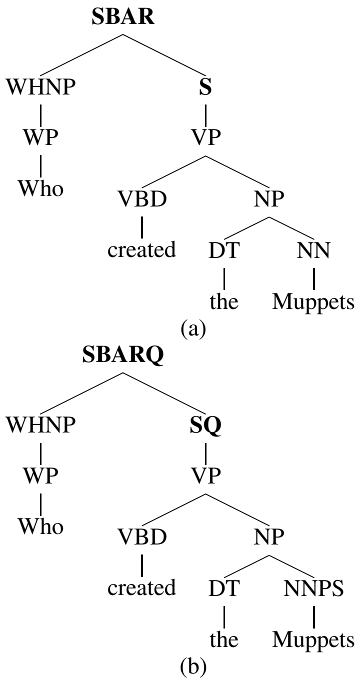
4.3 Basic Corpus Development Statistics

Our question treebank was created over a period of three months at an average annotation speed of about 60 questions per day. This is quite rapid for treebank development. The speed of the process was helped by two main factors: the questions are generally quite short (typically about 10 words long), and, due to retraining on the continually increasing training set, the quality of the parses output by the parser improved dramatically during the development of the treebank, with the effect that corrections during the later stages were generally quite small and not as time consuming as during the initial phases of the bootstrapping process.

For example, in the first week of the project the trees from the parser were of relatively poor quality and over 78% of the trees needed to be corrected manually. This slowed the annotation process considerably and parse-annotated questions were being produced at an average rate of 40 trees per day. During the later stages of the project this had changed dramatically. The quality of trees from the parser was much improved with less than 20% of the trees requiring manual correction. At this stage parse-annotated questions were being produced at an average rate of 90 trees per day.

4.4 Corpus Development Error Analysis

Some of the more frequent errors in the parser output pertain to the syntactic analysis of WH-phrases (WHNP, WHPP, etc). In Sections 02-21 of the Penn-II Treebank, these are used more often in relative clause constructions than in questions. As a result many of the corpus questions were given syntactic analyses corresponding to relative clauses (SBAR with an embedded S) instead of as questions (SBARQ with an embedded SQ). Figure 1 provides an example.



(a)

(b)

Figure 1: Example tree before (a) and after correction (b)

Because the questions are typically short, an error like this has quite a large effect on the accuracy for the overall tree; in this case the f-score for the parser output (Figure 1(a)) would be only 60%. Errors of this nature were quite frequent in the first section of questions analysed by the parser, but with increased training material becoming available during successive iterations, this error became less frequent and towards the end of

[4] Downloaded from http://www.cis.upenn.edu/~dbikel /software.html#stat-parser

the project it was only seen in rare cases.

WH-XP marking was the source of a number of consistent (though infrequent) errors during annotation. This occurred mostly in PP constructions containing WHNPs. The parser would output a structure like Figure 2(a), where the PP mother of the WHNP is not correctly labelled as a WHPP as in Figure 2(b).

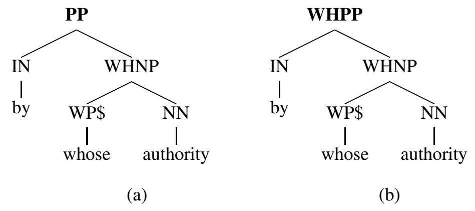


Figure 2: WH-XP assignment

The parser output often had to be rearranged structurally to varying degrees. This was common in the longer questions. A recurring error in the parser output was failing to identify VPs in SQs with a single object NP. In these cases the verb and the object NP were left as daughters of the SQ node. Figure 3(a) illustrates this, and Figure 3(b) shows the corrected tree with the VP node inserted.

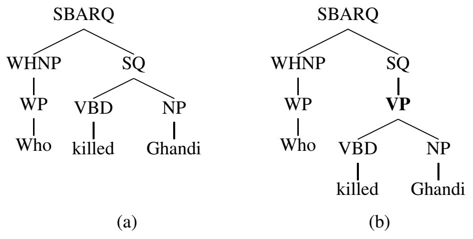


Figure 3: VP missing inside SQ with a single NP

On inspection, we found that the problem was caused by copular constructions, which, according to the Penn-II annotation guidelines, do not feature VP constituents. Since almost half of the question data contain copular constructions, the parser trained on this data would sometimes misanalyse non-copular constructions or, conversely, incorrectly bracket copular constructions using a VP constituent (Figure 4(a)).

The predictable nature of these errors meant that they were simple to correct. This is due to the particular context in which they occur and the finite number of forms of the copular verb.

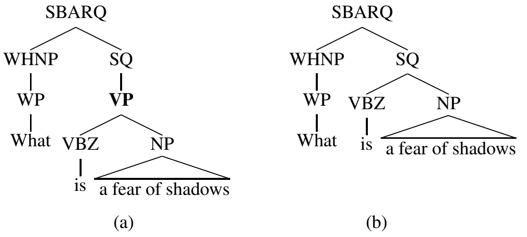


Figure 4: Erroneous VP in copular constructions

5 Experiments with QuestionBank

In order to test the effect training on the question corpus has on parser performance, we carried out a number of experiments. In cross-validation experiments with 90%/10% splits we use all 4000 trees in the completed QuestionBank as the test set. We performed ablation experiments to investigate the effect of varying the amount of question and non-question training data on the parser's performance. For these experiments we divided the 4000 questions into two sets. We randomly selected 400 trees to be held out as a gold standard test set against which to evaluate, the remaining 3600 trees were then used as a training corpus.

5.1 Establishing the Baseline

The baseline we use for our experiments is provided by Bikel's parser trained on WSJ Sections 02-21 of the Penn-II Treebank. We test on all 4000 questions in our question treebank, and also Section 23 of the Penn-II Treebank.

QuestionBank		WSJ Section 23	
Coverage	100	Coverage	100
F-Score	78.77	F-Score	82.97

Table 1: Baseline parsing results

Table 1 shows the results for our baseline evaluations on question and non-question test sets. While the coverage for both tests is high, the parser underperforms significantly on the question test set with a labelled bracketing f-score of 78.77 compared to 82.97 on Section 23 of the Penn-II Treebank. Note that unlike the published results for Bikel's parser in our evaluations we test on Section 23 and *include punctuation*.

5.2 Cross-Validation Experiments

We carried out two cross-validation experiments. In the first experiment we perform a 10-fold cross-validation experiment using our 4000 question treebank. In each case a randomly selected set of 10% of the questions in QuestionBank was held out during training and used as a test set. In this way parses from unseen data were generated for all 4000 questions and evaluated against the QuestionBank trees.

The second cross-validation experiment was similar to the first, but in each of the 10 folds we train on 90% of the 4000 questions in QuestionBank and on all of Sections 02-21 of the Penn-II Treebank.

In both experiments we also backtest each of the ten grammars on Section 23 of the Penn-II Treebank and report the average scores.

QuestionBank		Backtest on Sect 23	
Coverage	100	Coverage	98.79
F-Score	88.82	F-Score	59.79

Table 2: Cross-validation experiment using the 4000 question treebank

Table 2 shows the results for the first cross-validation experiment, using only the 4000 sentence QuestionBank. Compared to Table 1, the results show a significant improvement of over 10% on the baseline f-score for questions. However, the tests on the non-question Section 23 data show not only a significant drop in accuracy but also a drop in coverage.

Questions		Backtest on Sect 23	
Coverage	100	Coverage	100
F-Score	89.75	F-Score	82.39

Table 3: Cross-validation experiment using Penn-II Treebank Sections 02-21 and 4000 questions

Table 3 shows the results for the second cross-validation experiment using Sections 02-21 of the Penn-II Treebank and the 4000 questions in QuestionBank. The results show an even greater increase on the baseline f-score than the experiments using only the question training set (Table 2). The non-question results are also better and are comparable to the baseline (Table 1).

5.3 Ablation Runs

In a further set of experiments we investigated the effect of varying the amount of data in the parser's training corpus. We experiment with varying both the amount of QuestionBank and Penn-II Treebank data that the parser is trained on. In each experiment we use the 400 question test set and Section 23 of the Penn-II Treebank to evaluate against, and the 3600 question training set described above and Sections 02-21 of the Penn-II Treebank as the basis for the parser's training corpus. We report on three experiments:

In the first experiment we train the parser using only the 3600 question training set. We performed ten training and parsing runs in this experiment, incrementally reducing the size of the QuestionBank training corpus by 10% of the whole on each run.

The second experiment is similar to the first but in each run we add Sections 02-21 of the Penn-II Treebank to the (shrinking) training set of questions.

The third experiment is the converse of the second, the amount of questions in the training set remains fixed (all 3600) and the amount of Penn-II Treebank material is incrementally reduced by 10% on each run.

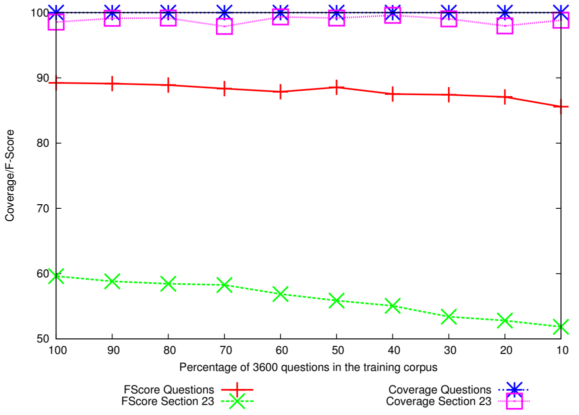


Figure 5: Results for ablation experiment reducing 3600 training questions in steps of 10%

Figure 5 graphs the coverage and f-score for the parser in tests on the 400 question test set, and Section 23 of the Penn-II Treebank in ten parsing runs with the amount of data in the 3600 question training corpus reducing incrementally on each run. The results show that training on only a small amount of questions, the parser can parse questions with high accuracy. For example when trained on only 10% of the 3600 questions used in this experiment, the parser successfully parses all of the 400 question test set and achieves an f-score of 85.59. However the results for the tests on WSJ Section 23 are considerably worse. The parser never manages to parse the full test set, and the best score at 59.61 is very low.

Figure 6 graphs the results for the second abla-

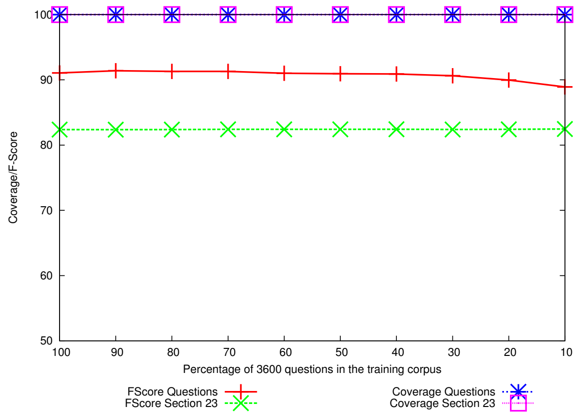


Figure 6: Results for ablation experiment using PTB Sections 02-21 (fixed) and reducing 3600 questions in steps of 10%

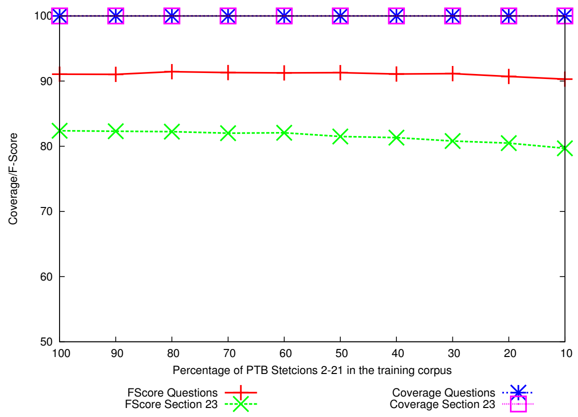


Figure 7: Results for ablation experiment using 3600 questions (fixed) and reducing PTB Sections 02-21 in steps of 10%

tion experiment. The training set for the parser consists of a fixed amount of Penn-II Treebank data (Sections 02-21) and a reducing amount of question data from the 3600 question training set. Each grammar is tested on both the 400 question test set, and WSJ Section 23. The results here are significantly better than in the previous experiment. In all of the runs the coverage for both test sets is 100%, f-scores for the question test set decrease as the amount of question data in the training set is reduced (though they are still quite high.) There is little change in the f-scores for the tests on Section 23, the results all fall in the range 82.36 to 82.46, which is comparable to the baseline score.

Figure 7 graphs the results for the third ablation experiment. In this case the training set is a fixed amount of the question training set described above (all 3600 questions) and a reducing amount of data from Sections 02-21 of the Penn Treebank. The graph shows that the parser performs consistently well on the question test set in terms of both coverage and accuracy. The tests on Section 23, however, show that as the amount of Penn-II Treebank material in the training set decreases, the f-score also decreases.

6 Long Distance Dependencies

Long distance dependencies are crucial in the proper analysis of question material. In English wh-questions, the fronted wh-constituent refers to an argument position of a verb inside the interrogative construction. Compare the superficially similar

1. Who$_1$ $[t_1]$ killed Harvey Oswald?

2. Who$_1$ did Harvey Oswald kill $[t_1]$?

(1) queries the agent (syntactic subject) of the described eventuality, while (2) queries the patient (syntactic object). In the Penn-II and ATIS treebanks, dependencies such as these are represented in terms of empty productions, traces and coindexation in CFG tree representations (Figure 8).

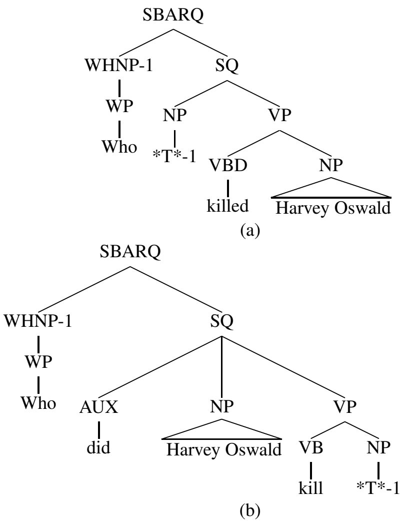


Figure 8: LDD resolved treebank style trees

With few exceptions[5] the trees produced by current treebank-based probabilistic parsers do not represent long distance dependencies (Figure 9).

Johnson (2002) presents a tree-based method for reconstructing LDD dependencies in Penn-II trained parser output trees. Cahill et al. (2004) present a method for resolving LDDs

[5] Collins' Model 3 computes a limited number of wh-dependencies in relative clause constructions.

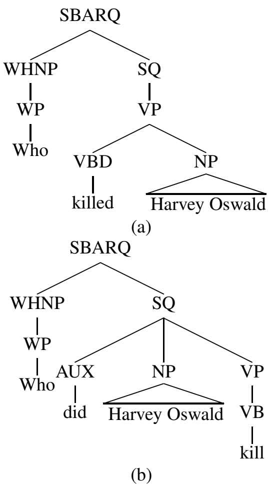


Figure 9: Parser output trees

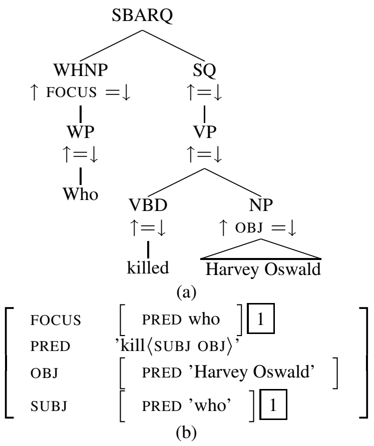


Figure 10: Annotated tree and f-structure

at the level of Lexical-Functional Grammar f-structure (attribute-value structure encodings of basic predicate-argument structure or dependency relations) without the need for empty productions and coindexation in parse trees. Their method is based on learning finite approximations of functional uncertainty equations (regular expressions over paths in f-structure) from an automatically f-structure annotated version of the Penn-II treebank and resolves LDDs at f-structure. In our work we use the f-structure-based method of Cahill et al. (2004) to "reverse engineer" empty productions, traces and coindexation in parser output trees. We explain the process by way of a worked example.

We use the parser output tree in Figure 9(a) (without empty productions and coindexation) and automatically annotate the tree with f-structure information and compute LDD-resolution at the level of f-structure using the resources of Cahill et al. (2004). This generates the f-structure annotated tree[6] and the LDD resolved f-structure in Figure 10.

Note that the LDD is indicated in terms of a reentrancy $\boxed{1}$ between the question FOCUS and the SUBJ function in the resolved f-structure. Given the correspondence between the f-structure and f-structure annotated nodes in the parse tree, we compute that the SUBJ function newly introduced and reentrant with the FOCUS function is an argument of the PRED 'kill' and the verb form 'killed' in the tree. In order to reconstruct the corresponding empty subject NP node in the parser output tree, we need to determine candidate anchor sites for the empty node. These anchor sites can only be realised along the path up to the maximal projection of the governing verb indicated by ↑=↓ annotations in LFG. This establishes three anchor sites: VP, SQ and the top level SBARQ. From the automatically f-structure annotated Penn-II treebank, we extract f-structure annotated PCFG rules for each of the three anchor sites whose RHSs contain exactly the information (daughter categories plus LFG annotations) in the tree in Figure 10 (in the same order) plus an additional node (of whatever CFG category) annotated ↑SUBJ=↓, located anywhere within the RHSs. This will retrieve rules of the form

VP $\rightarrow NP[\uparrow \textsc{subj} =\downarrow]\ VBD[\uparrow=\downarrow]\ NP[\uparrow \textsc{obj} =\downarrow]$
$VP \rightarrow \ldots$
$\ldots$
$SQ \rightarrow NP[\uparrow \textsc{subj} =\downarrow]\ VP[\uparrow=\downarrow]$
$SQ \rightarrow \ldots$
$\ldots$
$SBARQ \rightarrow \ldots$
$\ldots$

each with their associated probabilities. We select the rule with the highest probability and cut the rule into the tree in Figure 10 at the appropriate anchor site (as determined by the rule LHS). In our case this selects $SQ \rightarrow NP[\uparrow \textsc{subj}=\downarrow]VP[\uparrow=\downarrow]$ and the resulting tree is given in Figure 11. From this tree, it is now easy to compute the tree with the coindexed trace in Figure 8 (a).

In order to evaluate our empty node and coindexation recovery method, we conducted two experiments, one using 146 gold-standard ATIS question trees and one using parser output on the corresponding strings for the 146 ATIS question trees.

[6] Lexical annotations are suppressed to aid readability.

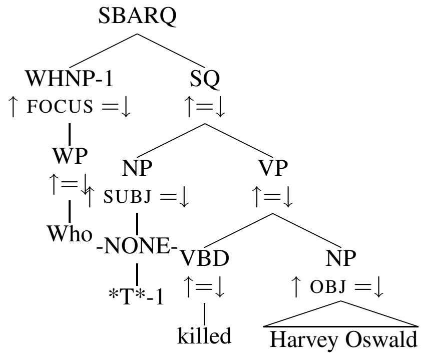


Figure 11: Resolved tree

In the first experiment, we delete empty nodes and coindexation from the ATIS gold standard trees and and reconstruct them using our method and the preprocessed ATIS trees. In the second experiment, we parse the strings corresponding to the ATIS trees with Bikel's parser and reconstruct the empty productions and coindexation. In both cases we evaluate against the original (unreduced) ATIS trees and score if and only if all of insertion site, inserted CFG category and coindexation match.

	Parser Output	Gold Standard Trees
Precision	96.77	96.82
Recall	38.75	39.38

Table 4: Scores for LDD recovery (empty nodes and antecedents)

Table 4 shows that currently the recall of our method is quite low at 39.38% while the accuracy is very high with precision at 96.82% on the ATIS trees. Encouragingly, evaluating parser output for the same sentences shows little change in the scores with recall at 38.75% and precision at 96.77%.

7 Conclusions

The data represented in Figure 5 show that training a parser on 50% of QuestionBank achieves an f-score of 88.56% as against 89.24% for training on all of QuestionBank. This implies that while we have not reached an absolute upper bound, the question corpus is sufficiently large that the gain in accuracy from adding more data is so small that it does not justify the effort.

We will evaluate grammars learned from QuestionBank as part of a working QA system. A beta-release of the non-LDD-resolved QuestionBank is available for download at `http://www.computing.dcu.ie/~jjudge/qtreebank/4000qs.txt`. The final, hand-corrected, LDD-resolved version will be available in October 2006.

Acknowledgments

We are grateful to the anonymous reviewers for their comments and suggestions. This research was supported by Science Foundation Ireland (SFI) grant 04/BR/CS0370 and an Irish Research Council for Science Engineering and Technology (IRCSET) PhD scholarship 2002-05.

References

Daniel M. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of HLT 2002*, pages 24–27, San Diego, CA.

Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2004. Long-Distance Dependency Resolution in Automatically Acquired Wide-Coverage PCFG-Based LFG Approximations. In *Proceedings of ACL-04*, pages 320–327, Barcelona, Spain.

Stephen Clark, Mark Steedman, and James R. Curran. 2004. Object-extraction and question-parsing using ccg. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP-04*, pages 111–118, Barcelona, Spain.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA.

Daniel Gildea. 2001. Corpus variation and parser performance. In Lillian Lee and Donna Harman, editors, *Proceedings of EMNLP*, pages 167–202, Pittsburgh, PA.

Charles T. Hemphill, John J. Godfrey, and George R. Doddington. 1990. The ATIS Spoken Language Systems pilot corpus. In *Proceedings of DARPA Speech and Natural Language Workshop*, pages 96–101, Hidden Valley, PA.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings ACL-02*, University of Pennsylvania, Philadelphia, PA.

John Judge, Aoife Cahill, Michael Burke, Ruth O'Donovan, Josef van Genabith, and Andy Way. 2005. Strong Domain Variation and Treebank-Induced LFG Resources. In *Proceedings LFG-05*, pages 186–204, Bergen, Norway, July.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of COLING-02*, pages 556–562, Taipei, Taiwan.

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.

Creating a CCGbank and a wide-coverage CCG lexicon for German

Julia Hockenmaier
Institute for Research in Cognitive Science
University of Pennsylvania
Philadelphia, PA 19104, USA
juliahr@cis.upenn.edu

Abstract

We present an algorithm which creates a German CCGbank by translating the syntax graphs in the German Tiger corpus into CCG derivation trees. The resulting corpus contains 46,628 derivations, covering 95% of all complete sentences in Tiger. Lexicons extracted from this corpus contain correct lexical entries for 94% of all known tokens in unseen text.

1 Introduction

A number of wide-coverage TAG, CCG, LFG and HPSG grammars (Xia, 1999; Chen et al., 2005; Hockenmaier and Steedman, 2002a; O'Donovan et al., 2005; Miyao et al., 2004) have been extracted from the Penn Treebank (Marcus et al., 1993), and have enabled the creation of wide-coverage parsers for English which recover local and non-local dependencies that approximate the underlying predicate-argument structure (Hockenmaier and Steedman, 2002b; Clark and Curran, 2004; Miyao and Tsujii, 2005; Shen and Joshi, 2005). However, many corpora (Böhomvá et al., 2003; Skut et al., 1997; Brants et al., 2002) use dependency graphs or other representations, and the extraction algorithms that have been developed for Penn Treebank style corpora may not be immediately applicable to this representation. As a consequence, research on statistical parsing with "deep" grammars has largely been confined to English. Free-word order languages typically pose greater challenges for syntactic theories (Rambow, 1994), and the richer inflectional morphology of these languages creates additional problems both for the coverage of lexicalized formalisms such as CCG or TAG, and for the usefulness of dependency counts extracted from the training data. On the other hand, formalisms such as CCG and TAG are particularly suited to capture the crossing dependencies that arise in languages such as Dutch or German, and by choosing an appropriate linguistic representation, some of these problems may be mitigated.

Here, we present an algorithm which translates the German Tiger corpus (Brants et al., 2002) into CCG derivations. Similar algorithms have been developed by Hockenmaier and Steedman (2002a) to create CCGbank, a corpus of CCG derivations (Hockenmaier and Steedman, 2005) from the Penn Treebank, by Çakıcı (2005) to extract a CCG lexicon from a Turkish dependency corpus, and by Moortgat and Moot (2002) to induce a type-logical grammar for Dutch.

The annotation scheme used in Tiger is an extension of that used in the earlier, and smaller, German Negra corpus (Skut et al., 1997). Tiger is better suited for the extraction of subcategorization information (and thus the translation into "deep" grammars of any kind), since it distinguishes between PP complements and modifiers, and includes "secondary" edges to indicate shared arguments in coordinate constructions. Tiger also includes morphology and lemma information.

Negra is also provided with a "Penn Treebank"-style representation, which uses flat phrase structure trees instead of the crossing dependency structures in the original corpus. This version has been used by Cahill et al. (2005) to extract a German LFG. However, Dubey and Keller (2003) have demonstrated that lexicalization does not help a Collins-style parser that is trained on this corpus, and Levy and Manning (2004) have shown that its context-free representation is a poor approximation to the underlying dependency structure. The resource presented here will enable future research to address the question whether "deep" grammars such as CCG, which capture the underlying dependencies directly, are better suited to parsing German than linguistically inadequate context-free approximations.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 505–512,
Sydney, July 2006. ©2006 Association for Computational Linguistics

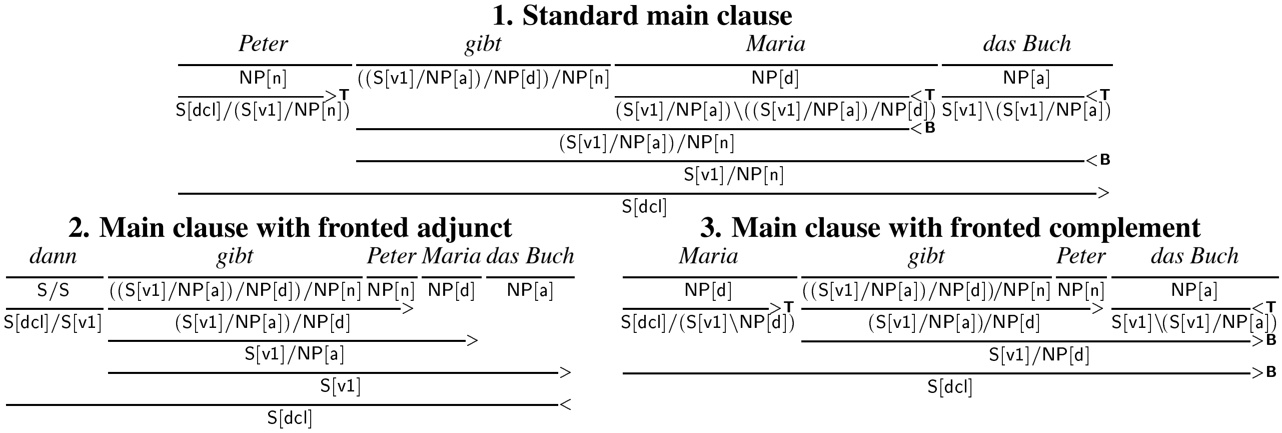


Figure 1: CCG uses topicalization (1.), a type-changing rule (2.), and type-raising (3.) to capture the different variants of German main clause order with the same lexical category for the verb.

2 German syntax and morphology

Morphology German verbs are inflected for person, number, tense and mood. German nouns and adjectives are inflected for number, case and gender, and noun compounding is very productive.

Word order German has three different word orders that depend on the clause type. Main clauses (1) are verb-second. Imperatives and questions are verb-initial (2). If a modifier or one of the objects is moved to the front, the word order becomes verb-initial (2). Subordinate and relative clauses are verb-final (3):

(1) a. *Peter gibt Maria das Buch.*
Peter gives Mary the book.
b. *ein Buch gibt Peter Maria.*
c. *dann gibt Peter Maria das Buch.*
(2) a. *Gibt Peter Maria das Buch?*
b. *Gib Maria das Buch!*
(3) a. *dass Peter Maria das Buch gibt.*
b. *das Buch, das Peter Maria gibt.*

Local Scrambling In the so-called "Mittelfeld" all orders of arguments and adjuncts are potentially possible. In the following example, all 5! permutations are grammatical (Rambow, 1994):

(4) *dass [eine Firma] [meinem Onkel] [die Möbel] [vor drei Tagen] [ohne Voranmeldung] zugestellt hat.*
that [a company] [to my uncle] [the furniture] [three days ago] [without notice] delivered has.

Long-distance scrambling Objects of embedded verbs can also be extraposed unboundedly within the same sentence (Rambow, 1994):

(5) *dass [den Schrank] [niemand] [zu reparieren] versprochen hat.*
that [the wardrobe] [nobody] [to repair] promised has.

3 A CCG for German

3.1 Combinatory Categorial Grammar

CCG (Steedman (1996; 2000)) is a lexicalized grammar formalism with a completely transparent syntax-semantics interface. Since CCG is mildly context-sensitive, it can capture the crossing dependencies that arise in Dutch or German, yet is efficiently parseable.

In categorial grammar, words are associated with syntactic categories, such as S\NP or (S\NP)/NP for English intransitive and transitive verbs. Categories of the form X/Y or X\Y are functors, which take an argument Y to their left or right (depending on the the direction of the slash) and yield a result X. Every syntactic category is paired with a semantic interpretation (usually a λ-term).

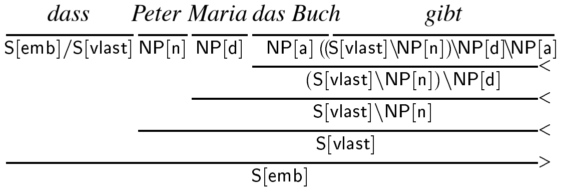
Like all variants of categorial grammar, CCG uses function application to combine constituents, but it also uses a set of combinatory rules such as composition (**B**) and type-raising (**T**). Non-order-preserving type-raising is used for topicalization:

Application:	X/Y	Y	$\Rightarrow$	X
	Y	X\Y	$\Rightarrow$	X
Composition:	X/Y	Y/Z	$\Rightarrow_{\mathbf{B}}$	X/Z
	X/Y	Y\Z	$\Rightarrow_{\mathbf{B}}$	X\Z
	Y\Z	X\Y	$\Rightarrow_{\mathbf{B}}$	X\Z
	Y/Z	X\Y	$\Rightarrow_{\mathbf{B}}$	X/Z
Type-raising:	X		$\Rightarrow_{\mathbf{T}}$	T\(T/X)
Topicalization:	X		$\Rightarrow_{\mathbf{T}}$	T/(T/X)

Hockenmaier and Steedman (2005) advocate the use of additional "type-changing" rules to deal with complex adjunct categories (e.g. (NP\NP) $\Rightarrow$ S[ng]\NP for *ing*-VPs that act as noun phrase modifiers). Here, we also use a small number of such rules to deal with similar adjunct cases.

3.2 Capturing German word order

We follow Steedman (2000) in assuming that the underlying word order in main clauses is always verb-initial, and that the sententce-initial subject is in fact topicalized. This enables us to capture different word orders with the same lexical category (Figure 1). We use the features S[v1] and S[vlast] to distinguish verbs in main and subordinate clauses. Main clauses have the feature S[dcl], requiring either a sentential modifier with category S[dcl]/S[v1], a topicalized subject (S[dcl]/(S[v1]/NP[nom])), or a type-raised argument (S[dcl]/(S[v1]\X)), where X can be any argument category, such as a noun phrase, prepositional phrase, or a non-finite VP. Here is the CCG derivation for the subordinate clause (S[emb]) example:



For simplicity's sake our extraction algorithm ignores the issues that arise through local scrambling, and assumes that there are different lexical category for each permutation.[1]

Type-raising and composition are also used to deal with wh-extraction and with long-distance scrambling (Figure 2).

4 Translating Tiger graphs into CCG

4.1 The Tiger corpus

The Tiger corpus (Brants et al., 2002) is a publicly available[2] corpus of ca. 50,000 sentences (almost 900,000 tokens) taken from the *Frankfurter Rundschau* newspaper. The annotation is based on a hybrid framework which contains features of phrase-structure and dependency grammar. Each sentence is represented as a graph whose nodes are labeled with syntactic categories (NP, VP, S, PP, etc.) and POS tags. Edges are directed and labeled with syntactic functions (e.g. head, subject, accusative object, conjunct, appositive). The edge labels are similar to the Penn Treebank function tags, but provide richer and more explicit information. Only 72.5% of the graphs have no crossing edges; the remaining 27.5% are marked as discontinuous. 7.3% of the sentences have one or more "secondary" edges, which are used to indicate double dependencies that arise in coordinated structures which are difficult to bracket, such as right node raising, argument cluster coordination or gapping. There are no traces or null elements to indicate non-local dependencies or wh-movement.

Figure 2 shows the Tiger graph for a PP whose NP argument is modified by a relative clause. There is no NP level inside PPs (and no noun level inside NPs). Punctuation marks are often attached at the so-called "virtual" root (VROOT) of the entire graph. The relative pronoun is a dative object (edge label DA) of the embedded infinitive, and is therefore attached at the VP level. The relative clause itself has the category S; the incoming edge is labeled RC (relative clause).

4.2 The translation algorithm

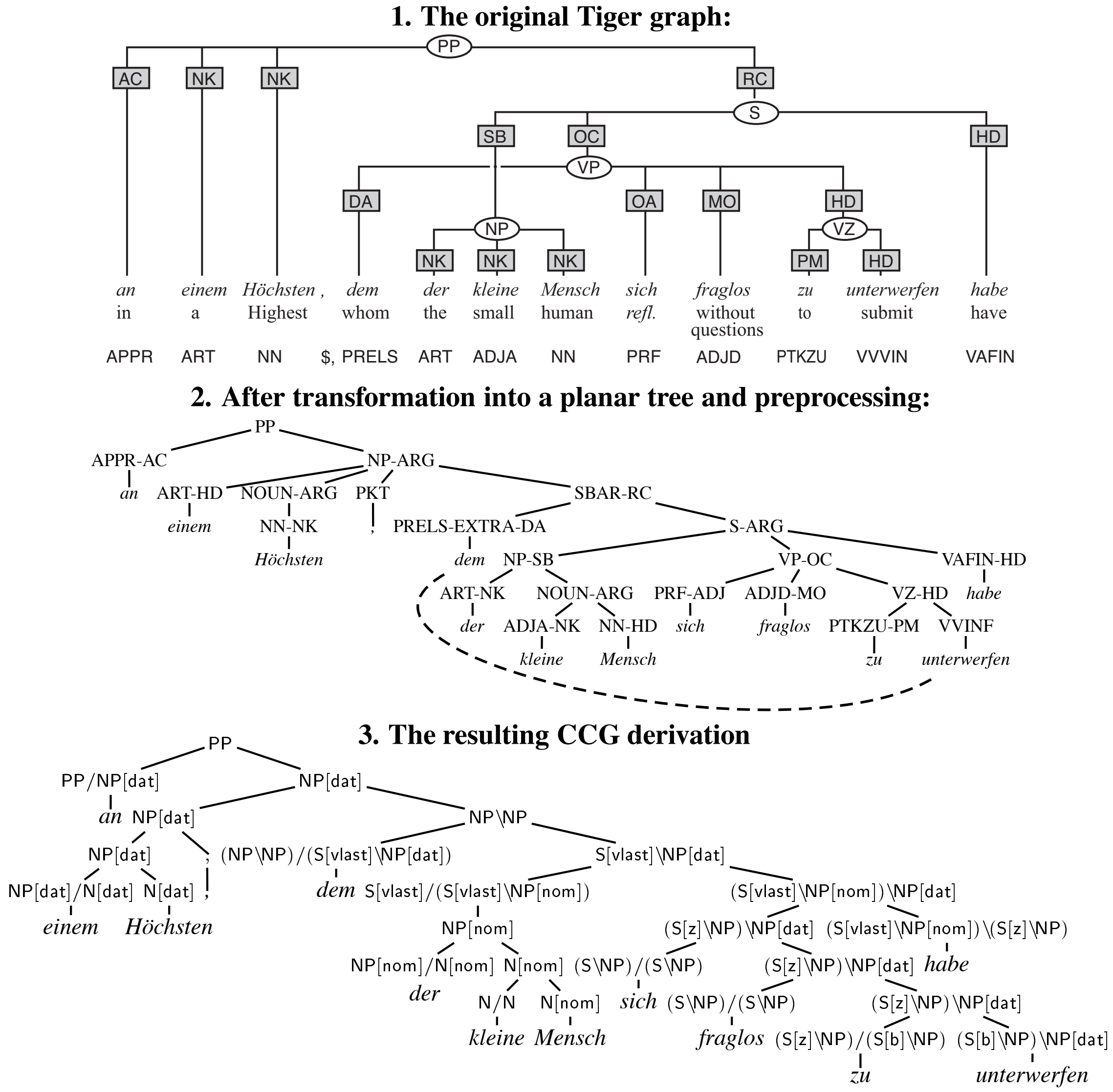
Our translation algorithm has the following steps:

```
translate(TigerGraph g):
    TigerTree t = createTree(g);
    preprocess(t);
    if ($t \neq$ null)
        CCGderiv d = translateToCCG(t);
        if ($d \neq$ null);
            if (isCCGderivation(d))
                return d;
            else fail;
        else fail;
    else fail;
```

1. Creating a planar tree: After an initial preprocessing step which inserts punctuation that is attached to the "virtual" root (VROOT) of the graph in the appropriate locations, discontinuous graphs are transformed into planar trees. Starting at the lowest nonterminal nodes, this step turns the Tiger graph into a planar tree without crossing edges, where every node spans a contiguous substring. This is required as input to the actual translation step, since CCG derivations are planar binary trees. If the first to the ith child of a node X span a contiguous substring that ends in the jth word, and the $(i+1)$th child spans a substring starting at $k > j+1$, we attempt to move the first i children of X to its parent P (if the head position of P is greater than i). Punctuation marks and adjuncts are simply moved up the tree and treated as if they were originally attached to P. This changes the syntactic scope of adjuncts, but typically only VP modifiers are affected which could also be attached at a higher VP or S node without a change in meaning. The main exception

[1] Variants of CCG, such as Set-CCG (Hoffman, 1995) and Multimodal-CCG (Baldridge, 2002), allow a more compact lexicon for free word order languages.

[2] http://www.ims.uni-stuttgart.de/projekte/TIGER


1. The original Tiger graph:

PP
AC
NK
NK
RC
S
SB
OC
HD
VP
DA
OA
MO
HD
NP
VZ
NK
NK
NK
PM
HD
an einem Höchsten , dem der kleine Mensch sich fraglos zu unterwerfen habe
in a Highest whom the small human refl. without questions to submit have
APPR ART NN $, PRELS ART ADJA NN PRF ADJD PTKZU VVVIN VAFIN

2. After transformation into a planar tree and preprocessing:

3. The resulting CCG derivation

Figure 2: From Tiger graphs to CCG derivations

are extraposed relative clauses, which CCG treats as sentential modifiers with an anaphoric dependency. Arguments that are moved up are marked as extracted, and an additional "extraction" edge (explained below) from the original head is introduced to capture the correct dependencies in the CCG derivation. Discontinuous dependencies between resumptive pronouns ("place holders", PH) and their antecedents ("repeated elements", RE) are also dissolved.

2. Additional preprocessing: In order to obtain the desired CCG analysis, a certain amount of preprocessing is required. We insert NPs into PPs, nouns into NPs[3], and change sentences whose first element is a complementizer (*dass*, *ob*, etc.) into an SBAR (a category which does not exist in the original Tiger annotation) with S argument. This is necessary to obtain the desired CCG derivations where complementizers and prepositions take a sentential or nominal argument to their right, whereas they appear at the same level as their arguments in the Tiger corpus. Further preprocessing is required to create the required structures for wh-extraction and certain coordination phenomena (see below).

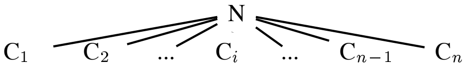
In figure 2, preprocessing of the original Tiger graph (top) yields the tree shown in the middle (edge labels are shown as Penn Treebank-style function tags).[4]

We will first present the basic translation algorithm before we explain how we obtain a derivation which captures the dependency between the relative pronoun and the embedded verb.

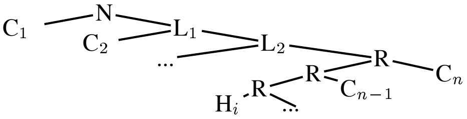
[3] The span of nouns is given by the NK edge label.

[4] We treat reflexive pronouns as modifiers.

3. The basic translation step Our basic translation algorithm is very similar to Hockenmaier and Steedman (2005). It requires a planar tree without crossing edges, where each node is marked as head, complement or adjunct. The latter information is represented in the Tiger edge labels, and only a small number of additional head rules is required. Each individual translation step operates on local trees, which are typically flat.



Assuming the CCG category of N is X, and its head position is i, the algorithm traverses first the left nodes $C_1...C_{i-1}$ from left to right to create a right-branching derivation tree, and then the right nodes $(C_n...C_{i+1})$ from right to left to create a left-branching tree. The algorithm starts at the root category and recursively traverses the tree.

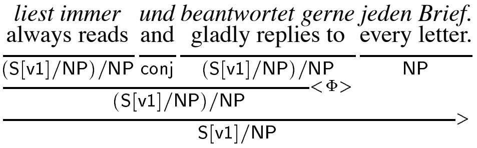


The CCG category of complements and of the root of the graph is determined from their Tiger label. VPs are S[.]\NP, where the feature [.] distinguishes bare infinitives, *zu*-infinitives, passives, and (active) past participles. With the exception of passives, these features can be determined from the POS tags alone.[5] Embedded sentences (under an SBAR-node) are always S[vlast]. NPs and nouns (NP and N) have a case feature, e.g. [nom].[6] Like the English CCGbank, our grammar ignores number and person agreement.

Special cases: Wh-extraction and extraposition In Tiger, wh-extraction is not explicitly marked. Relative clauses, wh-questions and free relatives are all annotated as S-nodes,and the wh-word is a normal argument of the verb. After turning the graph into a planar tree, we can identify these constructions by searching for a relative pronoun in the leftmost child of an S node (which may be marked as extraposed in the case of extraction from an embedded verb). As shown in figure 2, we turn this S into an SBAR (a category which does not exist in Tiger) with the first edge as complementizer and move the remaining children under a new S node which becomes the second daughter of the SBAR. The relative pronoun is the head of this SBAR and takes the S-node as argument. Its category is S[vlast], since all clauses with a complementizer are verb-final. In order to capture the long-range dependency, a "trace" is introduced, and percolated down the tree, much like in the algorithm of Hockenmaier and Steedman (2005), and similar to GPSG's slash-passing (Gazdar et al., 1985). These trace categories are appended to the category of the head node (and other arguments are type-raised as necessary). In our case, the trace is also associated with the verb whose argument it is. If the span of this verb is within the span of a complement, the trace is percolated down this complement. When the VP that is headed by this verb is reached, we assume a canonical order of arguments in order to "discharge" the trace.

If a complement node is marked as extraposed, it is also percolated down the head tree until the constituent whose argument it is is found. When another complement is found whose span includes the span of the constituent whose argument the extraposed edge is, the extraposed category is percolated down this tree (we assume extraction out of adjuncts is impossible).[7] In order to capture the topicalization analysis, main clause subjects also introduce a trace. Fronted complements or subjects, and the first adjunct in main clauses are analyzed as described in figure 1.

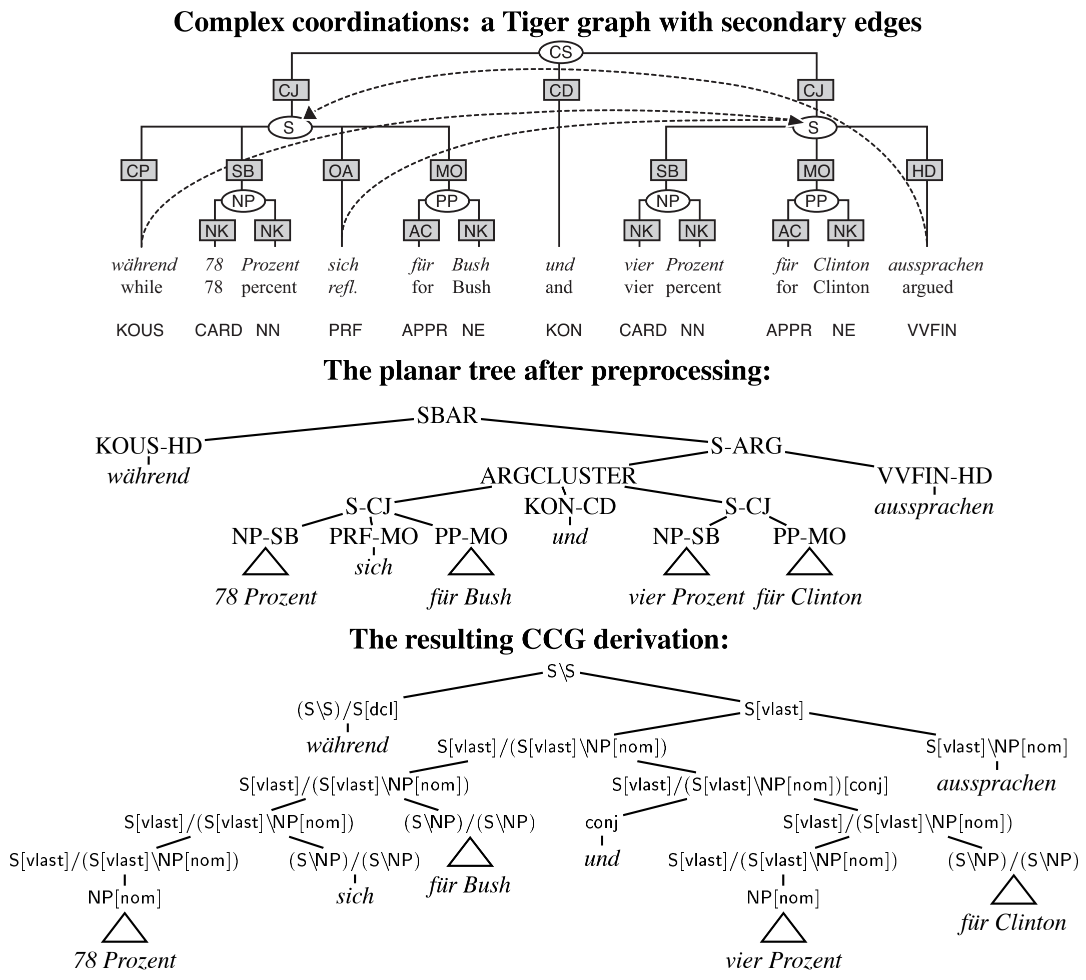
Special case: coordination – secondary edges Tiger uses "secondary edges" to represent the dependencies that arise in coordinate constructions such as gapping, argument cluster coordination and right (or left) node raising (Figure 3). In right (left) node raising, the shared elements are arguments or adjuncts that appear on the right periphery of the last, (or left periphery of the first) conjunct. CCG uses type-raising and composition to combine the incomplete conjuncts into one constituent which combines with the shared element:



[5] Eventive (*"werden"*) passive is easily identified by context; however, we found that not all stative (*"sein"*) passives seem to be annotated as such.

[6] In some contexts, measure nouns (e.g. *Mark*, *Kilometer*) lack case annotation.

[7] In our current implementation, each node cannot have more than one forward and one backward extraposed element and one forward and one backward trace. It may be preferable to use list structures instead, especially for extraposition.


Complex coordinations: a Tiger graph with secondary edges

The planar tree after preprocessing:

The resulting CCG derivation:

CS
CJ
CD
CJ
S
S
CP
SB
OA
MO
SB
MO
HD
NP
PP
NP
PP
NK NK
AC NK
NK NK
AC NK
während 78 Prozent sich für Bush und vier Prozent für Clinton aussprachen
while 78 percent refl. for Bush and vier percent for Clinton argued
KOUS CARD NN PRF APPR NE KON CARD NN APPR NE VVFIN
SBAR
KOUS-HD
während
S-ARG
ARGCLUSTER
VVFIN-HD
aussprachen
S-CJ
KON-CD
und
S-CJ
NP-SB
PRF-MO
sich
PP-MO
NP-SB
PP-MO
78 Prozent
für Bush
vier Prozent
für Clinton
S\S
(S\S)/S[dcl]
während
S[vlast]
S[vlast]/(S[vlast]\NP[nom])
S[vlast]\NP[nom]
aussprachen
S[vlast]/(S[vlast]\NP[nom])
S[vlast]/(S[vlast]\NP[nom])[conj]
S[vlast]/(S[vlast]\NP[nom])
(S\NP)/(S\NP)
conj
und
S[vlast]/(S[vlast]\NP[nom])
S[vlast]/(S[vlast]\NP[nom])
(S\NP)/(S\NP)
S[vlast]/(S[vlast]\NP[nom])
(S\NP)/(S\NP)
NP[nom]
für Bush
sich
NP[nom]
für Clinton
78 Prozent
vier Prozent

Figure 3: Processing secondary edges in Tiger

In order to obtain this analysis, we lift such shared peripheral constituents inside the conjuncts of conjoined sentences CS (or verb phrases, CVP) to new S (VP) level that we insert in between the CS and its parent.

In argument cluster coordination (Figure 3), the shared peripheral element (*aussprachen*) is the head.[8] In CCG, the remaining arguments and adjuncts combine via composition and typeraising into a functor category which takes the category of the head as argument (e.g. a ditransitive verb), and returns the same category that would result from a non-coordinated structure (e.g. a VP). The result category of the furthest element in each conjunct is equal to the category of the entire VP (or sentence), and all other elements are type-raised and composed with this to yield a category which takes as argument a verb with the required subcat frame and returns a verb phrase (sentence). Tiger assumes instead that there are two conjuncts (one of which is headless), and uses secondary edges to indicate the dependencies between the head and the elements in the distant conjunct. Coordinated sentences and VPs (CS and CVP) that have this annotation are rebracketed to obtain the CCG constituent structure, and the conjuncts are marked as argument clusters. Since the edges in the argument cluster are labeled with their correct syntactic functions, we are able to mimic the derivation during category assignment.

In sentential gapping, the main verb is shared and appears in the middle of the first conjunct:

(6) *Er trinkt Bier und sie Wein.*
He drinks beer and she wine.

As in the English CCGbank, we ignore this construction, which requires a non-combinatory "decomposition" rule (Steedman, 1990).

5 Evaluation

Translation coverage The algorithm can fail at several stages. If the graph cannot be turned into a tree, it cannot be translated. This happens in 1.3% (647) of all sentences. In many cases, this is due

[8] *Während* has scope over the entire coordinated structure.

to coordinated NPs or PPs where one or more conjuncts are extraposed. We believe that these are anaphoric, and further preprocessing could take care of this. In other cases, this is due to verb topicalization (*gegeben hat Peter Maria das Buch*), which our algorithm cannot currently deal with.[9] For 1.9% of the sentences, the algorithm cannot obtain a correct CCG derivation. Mostly this is the case because some traces and extraposed elements cannot be discharged properly. Typically this happens either in local scrambling, where an object of the main verb appears between the auxiliary and the subject (*hat das Buch Peter...*)[10], or when an argument of a noun that appears in a relative clause is extraposed to the right. There are also a small number of constituents whose head is not annotated. We ignore any gapping construction or argument cluster coordination that we cannot get into the right shape (1.5%), 732 sentences).

There are also a number of other constructions that we do not currently deal with. We do not process sentences if the root of the graph is a "virtual root" that does not expand into a sentence (1.7%, 869). This is mostly the case for strings such as *Frankfurt (Reuters)*), or if we cannot identify a head child of the root node (1.3%, 648; mostly fragments or elliptical constructions).

Overall, we obtain CCG derivations for 92.4% (46,628) of all 54,0474 sentences, including 88.4% (12,122) of those whose Tiger graphs are marked as discontinuous (13,717), and 95.2% of all 48,957 full sentences (excluding headless roots, and fragments, but counting coordinate structures such as gapping).

Lexicon size There are 2,506 lexical category types, but 1,018 of these appear only once. 933 category types appear more than 5 times.

Lexical coverage In order to evaluate coverage of the extracted lexicon on unseen data, we split the corpus into segments of 5,000 sentences (ignoring the last 474), and perform 10-fold cross-validation, using 9 segments to extract a lexicon and the 10th to test its coverage. Average coverage is 86.7% (by token) of all lexical categories. Coverage varies between 84.4% and 87.6%. On average, 92% (90.3%-92.6%) of the lexical tokens that appear in the held-out data also appear in the training data. On these seen tokens, coverage is 94.2% (93.5%-92.6%). More than half of all missing lexical entries are nouns.

In the English CCGbank, a lexicon extracted from section 02-21 (930,000 tokens) has 94% coverage on all tokens in section 00, and 97.7% coverage on all seen tokens (Hockenmaier and Steedman, 2005). In the English data set, the proportion of seen tokens (96.2%) is much higher, most likely because of the relative lack of derivational and inflectional morphology. The better lexical coverage on seen tokens is also to be expected, given that the flexible word order of German requires case markings on all nouns as well as at least two different categories for each tensed verb, and more in order to account for local scrambling.

6 Conclusion and future work

We have presented an algorithm which converts the syntax graphs in the German Tiger corpus (Brants et al., 2002) into Combinatory Categorial Grammar derivation trees. This algorithm is currently able to translate 92.4% of all graphs in Tiger, or 95.2% of all full sentences. Lexicons extracted from this corpus contain the correct entries for 86.7% of all and 94.2% of all seen tokens. Good lexical coverage is essential for the performance of statistical CCG parsers (Hockenmaier and Steedman, 2002a). Since the Tiger corpus contains complete morphological and lemma information for all words, future work will address the question of how to identify and apply a set of (non-recursive) lexical rules (Carpenter, 1992) to the extracted CCG lexicon to create a much larger lexicon. The number of lexical category types is almost twice as large as that of the English CCGbank. This is to be expected, since our grammar includes case features, and German verbs require different categories for main and subordinate clauses. We currently perform only the most essential preprocessing steps, although there are a number of constructions that might benefit from additional changes (e.g. comparatives, parentheticals, or fragments), both to increase coverage and accuracy of the extracted grammar.

Since Tiger corpus is of comparable size to the Penn Treebank, we hope that the work presented here will stimulate research into statistical wide-coverage parsing of free word order languages such as German with deep grammars like CCG.

[9] The corresponding CCG derivation combines the remnant complements as in argument cluster coordination.

[10] This problem arises because Tiger annotates subjects as arguments of the auxiliary. We believe this problem could be avoided if they were instead arguments of the non-finite verb.

Acknowledgments

I would like to thank Mark Steedman and Aravind Joshi for many helpful discussions. This research is supported by NSF ITR grant 0205456.

References

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Alena Böhomvá, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The Prague Dependency Treebank: Three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Syntactially Annotated Corpora*. Kluwer.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lexius, and George Smith. 2002. The TIGER treebank. In *Workshop on Treebanks and Linguistic Theories*, Sozpol.

Aoife Cahill, Martin Forst, Mairead McCarthy, Ruth O'Donovan, Christian Rohrer, Josef van Genabith, and Andy Way. 2005. Treebank-based acquisition of multilingual unification-grammar resources. *Journal of Research on Language and Computation.*

Ruken Çakıcı. 2005. Automatic induction of a CCG grammar for Turkish. In *ACL Student Research Workshop*, pages 73–78, Ann Arbor, MI, June.

Bob Carpenter. 1992. Categorial grammars, lexical rules, and the English predicative. In Robert Levine, editor, *Formal Grammar: Theory and Implementation*, chapter 3. Oxford University Press.

John Chen, Srinivas Bangalore, and K. Vijay-Shanker. 2005. Automated extraction of Tree-Adjoining Grammars from treebanks. *Natural Language Engineering*.

Stephen Clark and James R. Curran. 2004. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain.

Amit Dubey and Frank Keller. 2003. Probabilistic parsing for German using Sister-Head dependencies. In Erhard Hinrichs and Dan Roth, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 96–103, Sapporo, Japan.

Gerald Gazdar, Ewan Klein, Geoffrey K. Pullum, and Ivan A. Sag. 1985. *Generalised Phrase Structure Grammar*. Blackwell, Oxford.

Julia Hockenmaier and Mark Steedman. 2002a. Acquiring compact lexicalized grammars from a cleaner Treebank. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*, pages 1974–1981, Las Palmas, Spain, May.

Julia Hockenmaier and Mark Steedman. 2002b. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 335–342, Philadelphia, PA.

Julia Hockenmaier and Mark Steedman. 2005. CCGbank: Users' manual. Technical Report MS-CIS-05-09, Computer and Information Science, University of Pennsylvania.

Beryl Hoffman. 1995. *Computational Analysis of the Syntax and Interpretation of 'Free' Word-order in Turkish.* Ph.D. thesis, University of Pennsylvania. IRCS Report 95-17.

Roger Levy and Christopher Manning. 2004. Deep dependencies from context-free statistical parsers: correcting the surface dependency approximation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

Yusuke Miyao and Jun'ichi Tsujii. 2005. Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 83–90, Ann Arbor, MI.

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. In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP-04)*.

Michael Moortgat and Richard Moot. 2002. Using the Spoken Dutch Corpus for type-logical grammar induction. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC)*.

Ruth O'Donovan, Michael Burke, Aoife Cahill, Josef van Genabith, and Andy Way. 2005. Large-scale induction and evaluation of lexical resources from the Penn-II and Penn-III Treebanks. *Computational Linguistics*, 31(3):329 – 365, September.

Owen Rambow. 1994. *Formal and Computational Aspects of Natural Language Syntax*. Ph.D. thesis, University of Pennsylvania, Philadelphia PA.

Libin Shen and Aravind K. Joshi. 2005. Incremental LTAG parsing. In *Proceedings of the Human Language Technology Conference / Conference of Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Fifth Conference on Applied Natural Language Processing*.

Mark Steedman. 1990. Gapping as constituent coordination. *Linguistics and Philosophy*, 13:207–263.

Mark Steedman. 1996. *Surface Structure and Interpretation.* MIT Press, Cambridge, MA. Linguistic Inquiry Monograph, 30.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Fei Xia. 1999. Extracting Tree Adjoining Grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99).*

Improved Discriminative Bilingual Word Alignment

Robert C. Moore **Wen-tau Yih** **Andreas Bode**
Microsoft Research
Redmond, WA 98052, USA
{bobmoore,scottyhi,abode}@microsoft.com

Abstract

For many years, statistical machine translation relied on generative models to provide bilingual word alignments. In 2005, several independent efforts showed that discriminative models could be used to enhance or replace the standard generative approach. Building on this work, we demonstrate substantial improvement in word-alignment accuracy, partly though improved training methods, but predominantly through selection of more and better features. Our best model produces the lowest alignment error rate yet reported on Canadian Hansards bilingual data.

1 Introduction

Until recently, almost all work in statistical machine translation was based on word alignments obtained from combinations of generative probabalistic models developed at IBM by Brown et al. (1993), sometimes augmented by an HMM-based model or Och and Ney's "Model 6" (Och and Ney, 2003). In 2005, however, several independent efforts (Liu et al., 2005; Fraser and Marcu, 2005; Ayan et al., 2005; Taskar et al., 2005; Moore, 2005; Ittycheriah and Roukos, 2005) demonstrated that discriminatively trained models can equal or surpass the alignment accuracy of the standard models, if the usual unlabeled bilingual training corpus is supplemented with human-annotated word alignments for only a small subset of the training data.

The work cited above makes use of various training procedures and a wide variety of features. Indeed, whereas it can be difficult to design a factorization of a generative model that incorporates all the desired information, it is relatively easy to add arbitrary features to a discriminative model. We take advantage of this, building on our existing framework (Moore, 2005), to substantially reduce the alignment error rate (AER) we previously reported, given the same training and test data. Through a careful choice of features, and modest improvements in training procedures, we obtain the lowest error rate yet reported for word alignment of Canadian Hansards data.

2 Overall Approach

As in our previous work (Moore, 2005), we train two models we call stage 1 and stage 2, both in the form of a weighted linear combination of feature values extracted from a pair of sentences and a proposed word alignment of them. The possible alignment having the highest overall score is selected for each sentence pair. Thus, for a sentence pair (e, f) we seek the alignment $\hat{a}$ such that

$$\hat{a} = \text{argmax}_a \sum_{i=1}^{n} \lambda_i f_i(a, e, f)$$

where the f_i are features and the λ_i are weights. The models are trained on a large number of bilingual sentence pairs, a small number of which have hand-created word alignments provided to the training procedure. A set of hand alignments of a different subset of the overall training corpus is used to evaluate the models.

In the stage 1 model, all the features are based on surface statistics of the training data, plus the hypothesized alignment. The entire training corpus is then automatically aligned using this model. The stage 2 model uses features based not only on the parallel sentences themselves but also on statistics of the alignments produced by the stage

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 513–520,
Sydney, July 2006. ©2006 Association for Computational Linguistics

1 model. The stage 1 model is discussed in Section 3, and the stage 2 model, in Section 4. After experimenting with many features and combinations of features, we made the final selection based on minimizing training set AER.

For alignment search, we use a method nearly identical to our previous beam search procedure, which we do not discuss in detail. We made two minor modifications to handle the possiblity that more than one alignment may have the same score, which we previously did not take into account. First, we modified the beam search so that the beam size dynamically expands if needed to accomodate all the possible alignments that have the same score. Second we implemented a structural tie breaker, so that the same alignment will always be chosen as the one-best from a set of alignments having the same score. Neither of these changes significantly affected the alignment results.

The principal training method is an adaptation of averaged perceptron learning as described by Collins (2002). The differences between our current and earlier training methods mainly address the observation that perceptron training is very sensitive to the order in which data is presented to the learner. We also investigated the large-margin training technique described by Tsochantaridis et al. (2004). The training procedures are described in Sections 5 and 6.

3 Stage 1 Model

In our previous stage 1 model, we used five features. The most informative feature was the sum of bilingual word-association scores for all linked word pairs, computed as a log likelihood ratio. We used two features to measure the degree of nonmonotonicity of alignments, based on traversing the alignment in the order of the source sentence tokens, and noting the instances where the corresponding target sentence tokens were not in left-to-right order. One feature counted the number of times there was a backwards jump in the order of the target sentence tokens, and the other summed the magnitudes of these jumps. In order to model the trade-off between one-to-one and many-to-one alignments, we included a feature that counted the number of alignment links such that one of the linked words participated in another link. Our fifth feature was the count of the number of words in the sentence pair left unaligned.

In addition to these five features, we employed two hard constraints. One constraint was that the only alignment patterns allowed were 1–1, 1–2, 1–3, 2–1, and 3–1. Thus, many-to-many link patterns were disallowed, and a single word could be linked to at most three other words. The second constraint was that a possible link was considered only if it involved the strongest degree of association within the sentence pair for at least one of the words to be linked. If both words had stronger associations with other words in the sentence pair, then the link was disallowed.

Our new stage 1 model includes all the features we used previously, plus the constraint on alignment patterns. The constraint involving strongest association is not used. In addition, our new stage 1 model employs the following features:

association score rank features We define the rank of an association with respect to a word in a sentence pair to be the number of association types (word-type to word-type) for that word that have higher association scores, such that words of both types occur in the sentence pair. The contraint on strength of association we previously used can be stated as a requirement that no link be considered unless the corresponding association is of rank 0 for at least one of the words. We replace this hard constraint with two features based on association rank. One feature totals the sum of the association ranks with respect to both words involved in each link. The second feature sums the minimum of association ranks with respect to both words involved in each link. For alignments that obey the previous hard constraint, the value of this second feature would always be 0.

jump distance difference feature In our original models, the only features relating to word order were those measuring nonmonotonicity. The likelihoods of various forward jump distances were not modeled. If alignments are dense enough, measuring nonmonotonicity gets at this indirectly; if every word is aligned, it is impossible to have large forward jumps without correspondingly large backwards jumps, because something has to link to the words that are jumped over. If word alignments are sparse, however, due to free translation, it is possible to have alignments with very different forward jumps, but the same backwards jumps. To differentiate such alignments, we introduce a feature that sums the differences between the distance between consecutive aligned source words and the distance between the closest target words they are aligned to.

many-to-one jump distance features It seems intuitive that the likelihood of a large forward jump on either the source or target side of an alignment is much less if the jump is between words that are both linked to the same word of the other language. This motivates the distinction between the d_1 and $d_{>1}$ parameters in IBM Models 4 and 5. We model this by including two features. One feature sums, for each word w, the number of words not linked to w that fall between the first and last words linked to w. The other features counts only such words that are linked to some word other than w. The intuition here is that it is not so bad to have a function word not linked to anything, between two words linked to the same word.

exact match feature We have a feature that sums the number of words linked to identical words. This is motivated by the fact that proper names or specialized terms are often the same in both languages, and we want to take advantage of this to link such words even when they are too rare to have a high association score.

lexical features Taskar et al. (2005) gain considerable benefit by including features counting the links between particular high frequency words. They use 25 such features, covering all pairs of the five most frequent non-punctuation words in each language. We adopt this type of feature but do so more agressively. We include features for all bilingual word pairs that have at least two co-occurrences in the labeled training data. In addition, we include features counting the number of unlinked occurrences of each word having at least two occurrences in the labeled training data.

In training our new stage 1 model, we were concerned that using so many lexical features might result in overfitting to the training data. To try to prevent this, we train the stage 1 model by first optimizing the weights for all other features, then optimizing the weights for the lexical features, with the other weights held fixed to their optimium values without lexical features.

4 Stage 2 Model

In our original stage 2 model, we replaced the log-likelihood-based word association statistic with the logarithm of the estimated conditional probability of a cluster of words being linked by the stage 1 model, given that they co-occur in a pair of aligned sentences, computed over the full (500,000 sentence pairs) training data. We estimated these probabilities using a discounted maximum likelihood estimate, in which a small fixed amount was subtracted from each link count:

$$LP_d(w_1, \ldots, w_k) = \frac{links_1(w_1, \ldots, w_k) - d}{cooc(w_1, \ldots, w_k)}$$

$LP_d(w_1, \ldots, w_k)$ represents the estimated conditional link probability for the cluster of words $w_1, \ldots, w_k$; $links_1(w_1, \ldots, w_k)$ is the number of times they are linked by the stage 1 model, d is the discount; and $cooc(w_1, \ldots, w_k)$ is the number of times they co-occur. We found that $d = 0.4$ seemed to minimize training set AER.

An important difference between our stage 1 and stage 2 models is that the stage 1 model considers each word-to-word link separately, but allows multiple links per word, as long as they lead to an alignment consisting only of one-to-one and one-to-many links (in either direction). The stage 2 model, however, uses conditional probabilities for both one-to-one and one-to-many clusters, but requires all clusters to be disjoint. Our original stage 2 model incorporated the same addtional features as our original stage 1 model, except that the feature that counts the number of links involved in non-one-to-one link clusters was omitted.

Our new stage 2 model differs in a number of ways from the original version. First we replace the estimated conditional probability of a cluster of words being linked with the estimated conditional odds of a cluster of words being linked:

$$LO(w_1, \ldots, w_k) = \frac{links_1(w_1, \ldots, w_k) + 1}{(cooc(w_1, \ldots, w_k) - links_1(w_1, \ldots, w_k)) + 1}$$

$LO(w_1, \ldots, w_k)$ represents the estimated conditional link odds for the cluster of words $w_1, \ldots, w_k$. Note that we use “add-one” smoothing in place of a discount.

Additional features in our new stage 2 model include the unaligned word feature used previously, plus the following features:

symmetrized nonmonotonicity feature We symmetrize the previous nonmonontonicity feature that sums the magnitude of backwards jumps, by averaging the sum of of backwards jumps in the target sentence order relative to the source sentence order, with the sum of the backwards jumps in the source sentence order relative to the target sentence order. We omit the feature that counts the number of backwards jumps.

multi-link feature This feature counts the number of link clusters that are not one-to-one. This enables us to model whether the link scores for these clusters are more or less reliable than the link scores for one-to-one clusters.

empirically parameterized jump distance feature We take advantage of the stage 1 alignment to incorporate a feature measuring the jump distances between alignment links that are more sophisticated than simply measuring the difference in source and target distances, as in our stage 1 model. We measure the (signed) source and target distances between all pairs of links in the stage 1 alignment of the full training data. From this, we estimate the odds of each possible target distance given the corresponding source distance:

$$JO(d_t|d_s) = \frac{C(target\_dist = d_t \wedge source\_dist = d_s) + 1}{C(target\_dist \neq d_t \wedge source\_dist = d_s) + 1}$$

We similarly estimate the odds of each possible source distance given the corresponding target distance. The feature values consist of the sum of the scaled log odds of the jumps between consecutive links in a hypothesized alignment, computed in both source sentence and target sentence order. This feature is applied only when both the source and target jump distances are non-zero, so that it applies only to jumps between clusters, not to jumps on the "many" side of a many-to-one cluster. We found it necessary to linearly scale these feature values in order to get good results (in terms of training set AER) when using perceptron training.[1] We found empirically that we could get good results in terms of training set AER by dividing each log odds estimate by the largest absolute value of any such estimate computed.

5 Perceptron Training

We optimize feature weights using a modification of averaged perceptron learning as described by Collins (2002). Given an initial set of feature weight values, the algorithm iterates through the labeled training data multiple times, comparing, for each sentence pair, the best alignment a_{hyp} according to the current model with the reference alignment a_{ref}. At each sentence pair, the weight for each feature is is incremented by a multiple of the difference between the value of the feature for the best alignment according to the model and the value of the feature for the reference alignment:

$$\lambda_i \leftarrow \lambda_i + \eta(f_i(a_{ref}, e, f) - f_i(a_{hyp}, e, f))$$

The updated feature weights are used to compute a_{hyp} for the next sentence pair. The multiplier η is called the learning rate. In the averaged perceptron, the feature weights for the final model are the average of the weight values over all the data rather than simply the values after the final sentence pair of the final iteration.

Differences between our approach and Collins's include averaging feature weights over each pass through the data, rather than over all passes; randomizing the order of the data for each learning pass; and performing an evaluation pass after each learning pass, with feature weights fixed to their average values for the preceding learning pass, during which training set AER is measured. This procedure is iterated until a local minimum on training set AER is found.

We initialize the weight of the anticipated most-informative feature (word-association scores in stage 1; conditional link probabilities or odds in stage 2) to 1.0, with other feature weights intialized to 0. The weight for the most informative feature is not updated. Allowing all weights to vary allows many equivalent sets of weights that differ only by a constant scale factor. Fixing one weight eliminates a spurious apparent degree of freedom.

Previously, we set the learning rate η differently in training his stage 1 and stage 2 models. For the stage 2 model, we used a single learning rate of 0.01. For the stage 1 model, we used a sequence of learning rates: 1000, 100, 10, and 1.0. At each transition between learning rates, we re-initialized the feature weights to the optimum values found with the previous learning rate.

In our current work, we make a number of modifications to this procedure. We reset the feature weights to the best averaged values we have yet seen at the begining of each learning pass through the data. Anecdotally, this seems to result in faster convergence to a local AER minimum. We also use multiple learning rates for both the stage 1 and

[1] Note that this is purely for effective training, since after training, one could adjust the feature weights according to the scale factor, and use the original feature values.

stage 2 models, setting the learning rates automatically. The initial learning rate is the maximum absolute value (for one word pair/cluster) of the word association, link probability, or link odds feature, divided by the number of labeled training sentence pairs. Since many of the feature values are simple counts, this allows a minimal difference of 1 in the feature value, if repeated in every training example, to permit a count feature to have as large a weighted value as the most informative feature, after a single pass through the data.

After the learning search terminates for a given learning rate, we reduce the learning rate by a factor of 10, and iterate until we judge that we are at a local minimum for this learning rate. We continue with progressively smaller learning rates until an entire pass through the data produces feature weights that differ so little from their values at the beginning of the pass that the training set AER does not change.

Two final modifications are inspired by the realization that the results of perceptron training are very sensitive to the order in which the data is presented. Since we randomize the order of the data on every pass, if we make a pass through the training data, and the training set AER increases, it may be that we simply encountered an unfortunate ordering of the data. Therefore, when training set AER increases, we retry two additional times with the same initial weights, but different random orderings of the data, before giving up and trying a smaller learning rate. Finally, we repeat the entire training process multiple times, and average the feature weights resulting from each of these runs. We currently use 10 runs of each model. This final averaging is inspired by the idea of “Bayes-point machines” (Herbrich and Graepel, 2001).

6 SVM Training

After extensive experiments with perceptron training, we wanted to see if we could improve the results obtained with our best stage 2 model by using a more sophisticated training method. Perceptron training has been shown to obtain good results for some problems, but occasionally very poor results are reported, notably by Taskar et al. (2005) for the word-alignment problem. We adopted the support vector machine (SVM) method for structured output spaces of Tsochantaridis et al. (2005), using Joachims’ SVM^{struct} package.

Like standard SVM learning, this method tries to find the hyperplane that separates the training examples with the largest margin. Despite a very large number of possible output labels (e.g., all possible alignments of a given pair of sentences), the optimal hyperplane can be efficiently approximated given the desired error rate, using a cutting plane algorithm. In each iteration of the algorithm, it adds the “best” incorrect predictions given the current model as constraints, and optimizes the weight vector subject only to them.

The main advantage of this algorithm is that it does not pose special restrictions on the output structure, as long as “decoding” can be done efficiently. This is crucial to us because several features we found very effective in this task are difficult to incorporate into structured learning methods that require decomposable features. This method also allows a variety of loss functions, but we use only simple 0-1 loss, which in our case means whether or not the alignment of a sentence pair is completely correct, since this worked as well as anything else we tried.

Our SVM method has a number of free parameters, which we tried tuning in two different ways. One way is minimizing training set AER, which is how we chose the stopping points in perceptron training. The other is five-fold cross validation. In this method, we train five times on 80% of the training data and test on the other 20%, with five disjoint subsets used for testing. The parameter values yielding the best averaged AER on the five test subsets of the training set are used to train the final model on the entire training set.

7 Evaluation

We used the same training and test data as in our previous work, a subset of the Canadian Hansards bilingual corpus supplied for the bilingual word alignment workshop held at HLT-NAACL 2003 (Mihalcea and Pedersen, 2003). This subset comprised 500,000 English-French sentences pairs, including 224 manually word-aligned sentence pairs for labeled training data, and 223 labeled sentences pairs as test data. Automatic sentence alignment of the training data was provided by Ulrich Germann, and the hand alignments of the labeled data were created by Franz Och and Hermann Ney (Och and Ney, 2003).

For baselines, Table 1 shows the test set results we previously reported, along with results for IBM Model 4, trained with Och’s Giza++ software

Alignment	Recall	Precision	AER
Prev LLR	0.829	0.848	0.160
CLP$_1$	0.889	0.934	0.086
CLP$_2$	0.898	0.947	0.075
Giza E → F	0.870	0.890	0.118
Giza F → E	0.876	0.907	0.106
Giza union	0.929	0.845	0.124
Giza intersection	0.817	0.981	0.097
Giza refined	0.908	0.929	0.079

Table 1: Baseline Results.

package, using the default configuration file (Och and Ney, 2003).[2] "Prev LLR" is our earlier stage 1 model, and CLP$_1$ and CLP$_2$ are two versions of our earlier stage 2 model. For CLP$_1$, conditional link probabilities were estimated from the alignments produced by our "Prev LLR" model, and for CLP$_2$, they were obtained from a yet earlier, heuristic alignment model. Results for IBM Model 4 are reported for models trained in both directions, English-to-French and French-to-English, and for the union, intersection, and what Och and Ney (2003) call the "refined" combination of the those two alignments.

Results for our new stage 1 model are presented in Table 2. The first line is for the model described in Section 3, optimizing non-lexical features before lexical features. The second line gives results for optimizing all features simultaneously. The next line omits lexical features entirely. The last line is for our original stage 1 model, but trained using our improved perceptron training method.

As we can see, our best stage 1 model reduces the error rate of previous stage 1 model by almost half. Comparing the first two lines shows that two-phase training of non-lexical and lexical features produces a 0.7% reduction in test set error. Although the purpose of the two-phase training was to mitigate overfitting to the training data, we also found training set AER was reduced (7.3% vs. 8.8%). Taken all together, the results show a 7.9% total reduction in error rate: 4.0% from new non-lexical features, 3.3% from lexical features with two-phase training, and 0.6% from other improvements in perceptron training.

Table 3 presents results for perceptron training of our new stage 2 model. The first line is for the model as described in Section 4. Since the use of log odds is somewhat unusual, in the second line we show results for a similiar model, but using log probabilities instead of log odds for both the link model and the jump model. This result is 0.2% worse than the log-odds-based model, but the difference is small enough to warrant testing its significance. Comparing the errors on each test sentence pair with a 2-tailed paired t test, the results were suggestive, but not significant ($p = 0.28$)

Alignment	Recall	Precision	AER
Two-phase train	0.907	0.928	0.081
One-phase train	0.911	0.912	0.088
No lex feats	0.889	0.885	0.114
Prev LLR (new train)	0.834	0.855	0.154

Table 2: Stage 1 Model Results.

Alignment	Recall	Precision	AER
Log odds	0.935	0.964	0.049
Log probs	0.934	0.962	0.051
CLP$_1$ (new A & T)	0.925	0.952	0.060
CLP$_1$ (new A)	0.917	0.955	0.063

Table 3: Stage 2 Model Results.

The third line of Table 3 shows results for our earlier CLP$_1$ model with probabilities estimated from our new stage 1 model alignments ("new A"), using our recent modifications to perceptron training ("new T"). These results are significantly worse than either of the two preceding models ($p < 0.0008$). The fourth line is for the same model and stage 1 alignments, but with our earlier perceptron training method. While the results are 0.3% worse than with our new training method, the difference is not significant ($p = 0.62$).

Table 4 shows the results of SVM training of the model that was best under perceptron training, tuning free parameters either by minimizing error on the entire training set or by 5-fold cross validation on the training set. The cross-validation method produced slightly lower test-set AER, but both results rounded to 4.7%. While these results are somewhat better than with perceptron training, the differences are not significant ($p \geq 0.47$).

8 Comparisons to Other Work

At the time we carried out the experiments described above, our sub-5% AER results were the best we were aware of for word alignment of Canadian Hansards bilingual data, although direct comparisons are problematic due to differences in

[2] Thanks to Chris Quirk for providing Giza++ alignments.

Alignment	Recall	Precision	AER
Min train err	0.941	0.962	0.047
5 × CV	0.942	0.962	0.047

Table 4: SVM Training Results.

total training data, labeled training data, and test data. The best previously reported result was by Och and Ney (2003), who obtained 5.2% AER for a combination including all the IBM models except Model 2, plus the HMM model and their Model 6, together with a bilingual dictionary, for the refined alignment combination, trained on three times as much data as we used.

Cherry and Lin's (2003) method obtained an AER of 5.7% as reported by Mihalcea and Pedersen (2003), the previous lowest reported error rate for a method that makes no use of the IBM models. Cherry and Lin's method is similar to ours in using explicit estimates of the probability of a link given the co-occurence of the linked words; but it is generative rather than discriminative, it requires a parser for the English side of the corpus, and it does not model many-to-one links. Taskar et al. (2005) reported 5.4% AER for a discriminative model that includes predictions from the intersection of IBM Model 4 alignments as a feature. Their best result without using information from the IBM models was 10.7% AER.

After completing the experiments described in Section 7, we became aware further developments in the line of research reported by Taskar et al. (Lacoste-Julien et al., 2006). By modifying their previous approach to allow many-to-one alignments and first-order interactions between alignments, Lacoste-Julien et al. have improved their best AER without using information from the more complex IBM models to 6.2%. Their best result, however, is obtained from a model that includes both a feature recording intersected IBM Model 4 predictions, plus a feature whose values are the alignment probabilities obtained from a pair of HMM alignment models trained in both directions in such a way that they agree on the alignment probabilities (Liang et al., 2006). With this model, they obtained a much lower 3.8% AER.

Lacoste-Julien very graciously provided both the IBM Model 4 predictions and the probabilities estimated by the bidirectional HMM models that they had used to compute these additional feature values. We then added features based on this information to see how much we could improve our best model. We also eliminated one other difference between our results and those of Lacoste-Julien et al., by training on all 1.1 million English-French sentence pairs from the 2003 word alignment workshop, rather than the 500,000 sentence pairs we had been using.

Since all our other feature values derived from probabilities are expressed as log odds, we also converted the HMM probabilities estimated by Liang et al. to log odds. To make this well defined in all cases, we thresholded high probabilities (including 1.0) at 0.999999, and low probabilities (including 0.0) at 0.1 (which we found produced lower training set error than using a very small non-zero probability, although we have not searched systematically for the optimal value).

In our latest experiments, we first established that simply increasing the unlabled training data to 1.1 million sentence pairs made very little difference, reducing the test-set AER of our stage 2 model under perceptron training only from 4.9% to 4.8%. Combining our stage 2 model features with the HMM log odds feature using SVM training with 5-fold cross validation yielded a substantial reduction in test-set AER to 3.9% (96.9% precision, 95.1% recall). We found it somewhat difficult to improve these results further by including IBM Model 4 intersection feature. We finally obtained our best results, however, for both training-set and test-set AER, by holding the stage 2 model feature weights at the values obtained by SVM training with the HMM log odds feature, and optimizing the HMM log odds feature weight and IBM Model 4 intersection feature weight with perceptron training.[3] This produced a test-set AER of 3.7% (96.9% precision, 95.5% recall).

9 Conclusions

For Canadian Hansards data, the test-set AER of 4.7% for our stage 2 model is one of the lowest yet reported for an aligner that makes no use of the expensive IBM models, and our test-set AER of 3.7% for the stage 2 model in combination with the HMM log odds and Model 4 intersection features is the lowest yet reported for any aligner.[4]

Perhaps if any general conclusion is to be drawn from our results, it is that in creating a discrim-

[3] At this writing we have not yet had time to try this with SVM training.

[4] However, the difference between our result and the 3.8% of Lacoste-Julien et al. is almost certainly not significant.

inative word alignment model, the model structure and features matter the most, with the discriminative training method of secondary importance. While we obtained a small improvements by varying the training method, few of the differences were statistically significant. Having better features was much more important.

References

Necip Fazil Ayan, Bonnie J. Dorr, and Christof Monz. 2005. NeurAlign: Combining Word Alignments Using Neural Networks. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pp. 65–72, Vancouver, British Columbia.

Peter F. Brown, 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.

Colin Cherry and Dekang Lin. 2003. A Probability Model to Improve Word Alignment. In *Proceedings of the 41st Annual Meeting of the ACL*, pp. 88–95, Sapporo, Japan.

Michael Collins. 2002. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1–8, Philadelphia, Pennsylvania.

Alexander Fraser and Daniel Marcu. 2005. ISI's Participation in the Romanian-English Alignment Task. In *Proceedings of the ACL Workshop on Building and Using Parallel Texts*, pp. 91–94, Ann Arbor, Michigan.

Ralf Herbrich and Thore Graepel. 2001. Large Scale Bayes Point Machines Advances. In *Neural Information Processing Systems 13*, pp. 528–534.

Abraham Ittycheriah and Salim Roukos. 2005. A Maximum Entropy Word Aligner for Arabic-English Machine Translation. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pp. 89–96, Vancouver, British Columbia.

Simon Lacoste-Julien, Ben Taskar, Dan Klein, and Michael Jordan. 2006. Word Alignment via Quadratic Assignment. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 112–119, New York City.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by Agreement. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 104–111, New York City.

Yang Liu, Qun Liu, and Shouxun Lin. 2005. Log-linear Models for Word Alignment. In *Proceedings of the 43rd Annual Meeting of the ACL*, pp. 459–466, Ann Arbor, Michigan.

Rada Mihalcea and Ted Pedersen. 2003. An Evaluation Exercise for Word Alignment. In *Proceedings of the HLT-NAACL 2003 Workshop, Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pp. 1–6, Edmonton, Alberta.

Robert C. Moore. 2005. A Discriminative Framework for Bilingual Word Alignment. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pp. 81–88, Vancouver, British Columbia.

Franz Joseph Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Ben Taskar, Simon Lacoste-Julien, and Dan Klein. 2005. A Discriminative Matching Approach to Word Alignment. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pp. 73–80, Vancouver, British Columbia.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2005. Large Margin Methods for Structured and Interdependent Output Variables. *Journal of Machine Learning Research (JMLR)*, pp. 1453–1484.

Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation

Deyi Xiong
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China, 100080
Graduate School of Chinese Academy of Sciences
dyxiong@ict.ac.cn

Qun Liu and Shouxun Lin
Institute of Computing Technology
Chinese Academy of Sciences
Beijing, China, 100080
{liuqun, sxlin}@ict.ac.cn

Abstract

We propose a novel reordering model for phrase-based statistical machine translation (SMT) that uses a maximum entropy (MaxEnt) model to predicate reorderings of neighbor blocks (phrase pairs). The model provides content-dependent, hierarchical phrasal reordering with generalization based on features automatically learned from a real-world bitext. We present an algorithm to extract all reordering events of neighbor blocks from bilingual data. In our experiments on Chinese-to-English translation, this MaxEnt-based reordering model obtains significant improvements in BLEU score on the NIST MT-05 and IWSLT-04 tasks.

1 Introduction

Phrase reordering is of great importance for phrase-based SMT systems and becoming an active area of research recently. Compared with word-based SMT systems, phrase-based systems can easily address reorderings of words within phrases. However, at the phrase level, reordering is still a computationally expensive problem just like reordering at the word level (Knight, 1999).

Many systems use very simple models to reorder phrases [1]. One is distortion model (Och and Ney, 2004; Koehn et al., 2003) which penalizes translations according to their jump distance instead of their content. For example, if N words are skipped, a penalty of N will be paid regardless of which words are reordered. This model takes the risk of penalizing long distance jumps which are common between two languages with very different orders. Another simple model is flat reordering model (Wu, 1996; Zens et al., 2004; Kumar et al., 2005) which is not content dependent either. Flat model assigns constant probabilities for monotone order and non-monotone order. The two probabilities can be set to prefer monotone or non-monotone orientations depending on the language pairs.

In view of content-independency of the distortion and flat reordering models, several researchers (Och et al., 2004; Tillmann, 2004; Kumar et al., 2005; Koehn et al., 2005) proposed a more powerful model called lexicalized reordering model that is phrase dependent. Lexicalized reordering model learns local orientations (monotone or non-monotone) with probabilities for each bilingual phrase from training data. During decoding, the model attempts to finding a Viterbi local orientation sequence. Performance gains have been reported for systems with lexicalized reordering model. However, since reorderings are related to concrete phrases, researchers have to design their systems carefully in order not to cause other problems, e.g. the data sparseness problem.

Another smart reordering model was proposed by Chiang (2005). In his approach, phrases are reorganized into hierarchical ones by reducing subphrases to variables. This template-based scheme not only captures the reorderings of phrases, but also integrates some phrasal generalizations into the global model.

In this paper, we propose a novel solution for phrasal reordering. Here, under the ITG constraint (Wu, 1997; Zens et al., 2004), we need to consider just two kinds of reorderings, *straight* and *inverted* between two consecutive blocks. Therefore reordering can be modelled as a problem of

[1] In this paper, we focus our discussions on phrases that are not necessarily aligned to syntactic constituent boundary.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 521–528,
Sydney, July 2006. ©2006 Association for Computational Linguistics

classification with only two labels, *straight* and *inverted*. In this paper, we build a maximum entropy based classification model as the reordering model. Different from lexicalized reordering, we do not use the whole block as reordering evidence, but only features extracted from blocks. This is more flexible. It makes our model reorder any blocks, observed in training or not. The whole maximum entropy based reordering model is embedded inside a log-linear phrase-based model of translation. Following the Bracketing Transduction Grammar (BTG) (Wu, 1996), we built a CKY-style decoder for our system, which makes it possible to reorder phrases hierarchically.

To create a maximum entropy based reordering model, the first step is learning reordering examples from training data, similar to the lexicalized reordering model. But in our way, any evidences of reorderings will be extracted, not limited to reorderings of bilingual phrases of length less than a predefined number of words. Secondly, features will be extracted from reordering examples according to feature templates. Finally, a maximum entropy classifier will be trained on the features.

In this paper we describe our system and the MaxEnt-based reordering model with the associated algorithm. We also present experiments that indicate that the MaxEnt-based reordering model improves translation significantly compared with other reordering approaches and a state-of-the-art distortion-based system (Koehn, 2004).

2 System Overview

2.1 Model

Under the BTG scheme, translation is more like monolingual parsing through derivations. Throughout the translation procedure, three rules are used to derive the translation

$$A \xrightarrow{[\,]} (A^1, A^2) \tag{1}$$

$$A \xrightarrow{\langle\,\rangle} (A^1, A^2) \tag{2}$$

$$A \rightarrow (x, y) \tag{3}$$

During decoding, the source sentence is segmented into a sequence of phrases as in a standard phrase-based model. Then the lexical rule (3) [2] is used to translate source phrase y into target phrase x and generate a block A. Later, the *straight* rule (1) merges two consecutive blocks into a single larger block in the straight order; while the *inverted* rule (2) merges them in the inverted order. These two merging rules will be used continuously until the whole source sentence is covered. When the translation is finished, a tree indicating the hierarchical segmentation of the source sentence is also produced.

In the following, we will define the model in a straight way, not in the dynamic programming recursion way used by (Wu, 1996; Zens et al., 2004). We focus on defining the probabilities of different rules by separating different features (including the language model) out from the rule probabilities and organizing them in a log-linear form. This straight way makes it clear how rules are used and what they depend on.

For the two merging rules *straight* and *inverted*, applying them on two consecutive blocks A^1 and A^2 is assigned a probability $Pr^m(A)$

$$Pr^m(A) = \Omega^{\lambda_\Omega} \cdot \triangle^{\lambda_{LM}}_{p_{LM}(A^1,A^2)} \tag{4}$$

where the Ω is the reordering score of block A^1 and A^2, λ_Ω is its weight, and $\triangle_{p_{LM}(A^1,A^2)}$ is the increment of the language model score of the two blocks according to their final order, λ_{LM} is its weight.

For the lexical rule, applying it is assigned a probability $Pr^l(A)$

$$\begin{aligned} Pr^l(A) \quad = \quad & p(x|y)^{\lambda_1} \cdot p(y|x)^{\lambda_2} \cdot p_{lex}(x|y)^{\lambda_3} \\ & \cdot p_{lex}(y|x)^{\lambda_4} \cdot exp(1)^{\lambda_5} \cdot exp(|x|)^{\lambda_6} \\ & \cdot p_{LM}^{\lambda_{LM}}(x) \end{aligned} \tag{5}$$

where $p(\cdot)$ are the phrase translation probabilities in both directions, $p_{lex}(\cdot)$ are the lexical translation probabilities in both directions, and $exp(1)$ and $exp(|x|)$ are the phrase penalty and word penalty, respectively. These features are very common in state-of-the-art systems (Koehn et al., 2005; Chiang, 2005) and λs are weights of features.

For the reordering model Ω, we define it on the two consecutive blocks A^1 and A^2 and their order $o \in \{straight, inverted\}$

$$\Omega = f(o, A^1, A^2) \tag{6}$$

Under this framework, different reordering models can be designed. In fact, we defined four reordering models in our experiments. The first one

[2] Currently, we restrict phrases x and y not to be null. Therefore neither deletion nor insertion is carried out during decoding. However, these operations are to be considered in our future version of model.

is *NONE*, meaning no explicit reordering features at all. We set Ω to 1 for all different pairs of blocks and their orders. So the phrasal reordering is totally dependent on the language model. This model is obviously different from the monotone search, which does not use the *inverted* rule at all. The second one is a distortion style reordering model, which is formulated as

$$\Omega = \begin{cases} exp(0), & o = straight \\ exp(|A^1|) + (|A^2|), & o = inverted \end{cases}$$

where $|A^i|$ denotes the number of words on the source side of blocks. When $\lambda_\Omega < 0$, this design will penalize those non-monotone translations. The third one is a flat reordering model, which assigns probabilities for the straight and inverted order. It is formulated as

$$\Omega = \begin{cases} p_m, & o = straight \\ 1 - p_m, & o = inverted \end{cases}$$

In our experiments on Chinese-English tasks, the probability for the straight order is set at $p_m = 0.95$. This is because word order in Chinese and English is usually similar. The last one is the maximum entropy based reordering model proposed by us, which will be described in the next section.

We define a derivation D as a sequence of applications of rules $(1) - (3)$, and let $c(D)$ and $e(D)$ be the Chinese and English yields of D. The probability of a derivation D is

$$Pr(D) = \prod_i Pr(i) \qquad (7)$$

where $Pr(i)$ is the probability of the ith application of rules. Given an input sentence c, the final translation e^* is derived from the best derivation D^*

$$\begin{aligned} D^* &= \operatorname*{argmax}_{c(D)=c} Pr(D) \\ e^* &= e(D^*) \end{aligned} \qquad (8)$$

2.2 Decoder

We developed a CKY style decoder that employs a beam search algorithm, similar to the one by Chiang (2005). The decoder finds the best derivation that generates the input sentence and its translation. From the best derivation, the best English e^* is produced.

Given a source sentence c, firstly we initiate the chart with phrases from phrase translation table by applying the lexical rule. Then for each cell that spans from i to j on the source side, all possible derivations spanning from i to j are generated. Our algorithm guarantees that any sub-cells within (i, j) have been expanded before cell (i, j) is expanded. Therefore the way to generate derivations in cell (i, j) is to merge derivations from any two neighbor sub-cells. This combination is done by applying the *straight* and *inverted* rules. Each application of these two rules will generate a new derivation covering cell (i, j). The score of the new generated derivation is derived from the scores of its two sub-derivations, reordering model score and the increment of the language model score according to the Equation (4). When the whole input sentence is covered, the decoding is over.

Pruning of the search space is very important for the decoder. We use three pruning ways. The first one is recombination. When two derivations in the same cell have the same w leftmost/rightmost words on the English yields, where w depends on the order of the language model, they will be recombined by discarding the derivation with lower score. The second one is the threshold pruning which discards derivations that have a score worse than α times the best score in the same cell. The last one is the histogram pruning which only keeps the top n best derivations for each cell. In all our experiments, we set $n = 40, \alpha = 0.5$ to get a tradeoff between speed and performance in the development set.

Another feature of our decoder is the k-best list generation. The k-best list is very important for the minimum error rate training (Och, 2003a) which is used for tuning the weights λ for our model. We use a very lazy algorithm for the k-best list generation, which runs two phases similarly to the one by Huang et al. (2005). In the first phase, the decoder runs as usual except that it keeps some information of weaker derivations which are to be discarded during recombination. This will generate not only the first-best of final derivation but also a shared forest. In the second phase, the lazy algorithm runs recursively on the shared forest. It finds the second-best of the final derivation, which makes its children to find their second-best, and children's children's second-best, until the leaf node's second-best. Then it finds the third-best, forth-best, and so on. In all our experiments, we set $k = 200$.

The decoder is implemented in C++. Using the pruning settings described above, without the k-best list generation, it takes about 6 seconds to translate a sentence of average length 28.3 words on a 2GHz Linux system with 4G RAM memory.

3 Maximum Entropy Based Reordering Model

In this section, we discuss how to create a maximum entropy based reordering model. As described above, we defined the reordering model Ω on the three factors: order o, block A^1 and block A^2. The central problem is, given two neighbor blocks A^1 and A^2, how to predicate their order $o \in \{straight, inverted\}$. This is a typical problem of two-class classification. To be consistent with the whole model, the conditional probability $p(o|A^1, A^2)$ is calculated. A simple way to compute this probability is to take counts from the training data and then to use the maximum likelihood estimate (MLE)

$$p(o|A^1, A^2) = \frac{Count(o, A^1, A^2)}{Count(A^1, A^2)} \quad (9)$$

The similar way is used by lexicalized reordering model. However, in our model this way can't work because blocks become larger and larger due to using the merging rules, and finally unseen in the training data. This means we can not use blocks as direct reordering evidences.

A good way to this problem is to use features of blocks as reordering evidences. Good features can not only capture reorderings, avoid sparseness, but also integrate generalizations. It is very straight to use maximum entropy model to integrate features to predicate reorderings of blocks. Under the MaxEnt model, we have

$$\Omega = p_\theta(o|A^1, A^2) = \frac{exp(\sum_i \theta_i h_i(o, A^1, A^2))}{\sum_o exp(\sum_i \theta_i h_i(o, A^1, A^2))} \quad (10)$$

where the functions $h_i \in \{0, 1\}$ are model features and the θ_i are weights of the model features which can be trained by different algorithms (Malouf, 2002).

3.1 Reordering Example Extraction Algorithm

The input for the algorithm is a bilingual corpus with high-precision word alignments. We obtain the word alignments using the way of Koehn et al. (2005). After running GIZA++ (Och and Ney, 2000) in both directions, we apply the "grow-diag-final" refinement rule on the intersection alignments for each sentence pair.

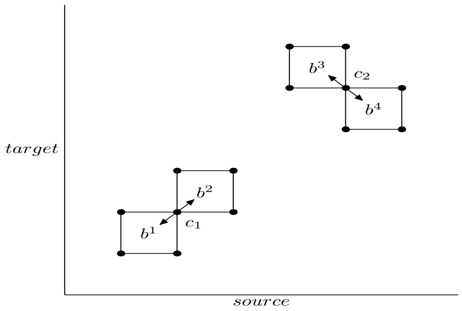


Figure 1: The bold dots are corners. The arrows from the corners are their links. Corner c_1 is shared by block b^1 and b^2, which in turn are linked by the STRAIGHT links, *bottomleft* and *topright* of c_1. Similarly, block b^3 and b^4 are linked by the INVERTED links, *topleft* and *bottomright* of c_2.

Before we introduce this algorithm, we introduce some formal definitions. The first one is *block* which is a pair of source and target contiguous sequences of words

$$b = (s_{i_1}^{i_2}, t_{j_1}^{j_2})$$

b must be consistent with the word alignment M

$$\forall(i, j) \in M, i_1 \leq i \leq i_2 \leftrightarrow j_1 \leq j \leq j_2$$

This definition is similar to that of bilingual phrase except that there is no length limitation over block. A *reordering example* is a triple of (o, b^1, b^2) where b^1 and b^2 are two neighbor blocks and o is the order between them. We define each vertex of block as *corner*. Each corner has four *links* in four directions: *topright, topleft, bottomright, bottomleft*, and each link links a set of blocks which have the corner as their vertex. The *topright* and *bottomleft* link blocks with the straight order, so we call them *STRAIGHT* links. Similarly, we call the *topleft* and *bottomright INVERTED* links since they link blocks with the inverted order. For convenience, we use $b \hookleftarrow \mathfrak{L}$ to denote that block b is linked by the link $\mathfrak{L}$. Note that the STRAIGHT links can not coexist with the INVERTED links. These definitions are illustrated in Figure 1.

The reordering example extraction algorithm is shown in Figure 2. The basic idea behind this algorithm is to register all neighbor blocks to the associated links of corners which are shared by them. To do this, we keep an array to record link

```
1: **Input**: sentence pair $(s, t)$ and their alignment $M$
2: $\Re := \emptyset$
3: **for** each span $(i_1, i_2) \in s$ **do**
4:     find block $b = (s_{i_1}^{i_2}, t_{j_1}^{j_2})$ that is consistent with $M$
5:     Extend block $b$ on the target boundary with one possi-
       ble non-aligned word to get blocks $E(b)$
6:     **for** each block $b^* \in b \bigcup E(b)$ **do**
7:         Register $b^*$ to the links of four corners of it
8:     **end for**
9: **end for**
10: **for** each corner $\mathcal{C}$ in the matrix $M$ **do**
11:     **if** STRAIGHT links exist **then**
12:         $\Re := \Re \bigcup \{(straight, b^1, b^2)\}$,
            $b^1 \hookleftarrow \mathcal{C}.bottomleft, b^2 \hookleftarrow \mathcal{C}.topright$
13:     **else if** INVERTED links exist **then**
14:         $\Re := \Re \bigcup \{(inverted, b^1, b^2)\}$,
            $b^1 \hookleftarrow \mathcal{C}.topleft, b^2 \hookleftarrow \mathcal{C}.bottomright$
15:     **end if**
16: **end for**
17: **Output**: reordering examples $\Re$
```

Figure 2: Reordering Example Extraction Algorithm.

information of corners when extracting blocks. Line 4 and 5 are similar to the phrase extraction algorithm by Och (2003b). Different from Och, we just extend one word which is aligned to null on the boundary of target side. If we put some length limitation over the extracted blocks and output them, we get bilingual phrases used in standard phrase-based SMT systems and also in our system. Line 7 updates all links associated with the current block. You can attach the current block to each of these links. However this will increase reordering examples greatly, especially those with the $straight$ order. In our Experiments, we just attach the smallest blocks to the STRAIGHT links, and the largest blocks to the INVERTED links. This will keep the number of reordering examples acceptable but without performance degradation. Line 12 and 14 extract reordering examples.

3.2 Features

With the extracted reordering examples, we can obtain features for our MaxEnt-based reordering model. We design two kinds of features, lexical features and collocation features. For a block $b = (s, t)$, we use s_1 to denote the first word of the source s, t_1 to denote the first word of the target t.

Lexical features are defined on the single word s_1 or t_1. Collocation features are defined on the combination s_1 or t_1 between two blocks b^1 and b^2. Three kinds of combinations are used. The first one is source collocation, $b^1.s_1 \& b^2.s_1$. The second is target collocation, $b^1.t_1 \& b^2.t_1$. The last one is block collocation, $b^1.s_1 \& b^1.t_1$ and $b^2.s_1 \& b^2.t_1$. The templates for the lexical feature and the collocation feature are shown in Figure 3.

$$h_i(o, b^1, b^2) = \begin{cases} 1, & b^1.t_1 = E_1, o = O \\ 0, & otherwise \end{cases}$$

$$h_j(o, b^1, b^2) = \begin{cases} 1, & b^1.t_1 = E_1, b^2.t_1 = E_2, o = O \\ 0, & otherwise \end{cases}$$

Figure 3: MaxEnt-based reordering feature templates. The first one is a lexical feature, and the second one is a target collocation feature, where E_i are English words, $O \in \{straight, inverted\}$.

Why do we use the first words as features? These words are nicely at the boundary of blocks. One of assumptions of phrase-based SMT is that phrase cohere across two languages (Fox, 2002), which means phrases in one language tend to be moved together during translation. This indicates that boundary words of blocks may keep information for their movements/reorderings. To test this hypothesis, we calculate the information gain ratio (IGR) for boundary words as well as the whole blocks against the order on the reordering examples extracted by the algorithm described above. The IGR is the measure used in the decision tree learning to select features (Quinlan, 1993). It represents how precisely the feature predicate the class. For feature f and class c, the $IGR(f, c)$

$$IGR(f, c) = \frac{En(c) - En(c|f)}{En(f)} \qquad (11)$$

where $En(\cdot)$ is the entropy and $En(\cdot|\cdot)$ is the conditional entropy. To our surprise, the IGR for the four boundary words $(IGR(\langle b^1.s_1,\ b^2.s_1,\ b^1.t_1,\ b^2.t_1\rangle,\ order) = 0.2637)$ is very close to that for the two blocks together $(IGR(\langle b^1,\ b^2\rangle,\ order) = 0.2655)$. Although our reordering examples do not cover all reordering events in the training data, this result shows that boundary words do provide some clues for predicating reorderings.

4 Experiments

We carried out experiments to compare against various reordering models and systems to demonstrate the competitiveness of MaxEnt-based reordering:

1. Monotone search: the *inverted* rule is not used.

2. Reordering variants: the *NONE*, distortion and flat reordering models described in Section 2.1.

3. Pharaoh: A state-of-the-art distortion-based decoder (Koehn, 2004).

4.1 Corpus

Our experiments were made on two Chinese-to-English translation tasks: NIST MT-05 (news domain) and IWSLT-04 (travel dialogue domain).

NIST MT-05. In this task, the bilingual training data comes from the FBIS corpus with 7.06M Chinese words and 9.15M English words. The trigram language model training data consists of English texts mostly derived from the English side of the UN corpus (catalog number LDC2004E12), which totally contains 81M English words. For the efficiency of minimum error rate training, we built our development set using sentences of length at most 50 characters from the NIST MT-02 evaluation test data.

IWSLT-04. For this task, our experiments were carried out on the small data track. Both the bilingual training data and the trigram language model training data are restricted to the supplied corpus, which contains 20k sentences, 179k Chinese words and 157k English words. We used the CSTAR 2003 test set consisting of 506 sentence pairs as development set.

4.2 Training

We obtained high-precision word alignments using the way described in Section 3.1. Then we ran our reordering example extraction algorithm to output blocks of length at most 7 words on the Chinese side together with their internal alignments. We also limited the length ratio between the target and source language ($max(|s|, |t|)/min(|s|, |t|)$) to 3. After extracting phrases, we calculated the phrase translation probabilities and lexical translation probabilities in both directions for each bilingual phrase.

For the minimum-error-rate training, we reimplemented Venugopal's trainer [3] (Venugopal et al., 2005) in C++. For all experiments, we ran this trainer with the decoder iteratively to tune the weights λs to maximize the BLEU score on the development set.

Pharaoh
We shared the same phrase translation tables between Pharaoh and our system since the two systems use the same features of phrases. In fact, we extracted more phrases than Pharaoh's trainer with its default settings. And we also used our reimplemented trainer to tune lambdas of Pharaoh to maximize its BLEU score. During decoding, we pruned the phrase table with $b = 100$ (default 20), pruned the chart with $n = 100, \alpha = 10^{-5}$ (default setting), and limited distortions to 4 (default 0).

MaxEnt-based Reordering Model
We firstly ran our reordering example extraction algorithm on the bilingual training data without any length limitations to obtain reordering examples and then extracted features from these examples. In the task of NIST MT-05, we obtained about 2.7M reordering examples with the straight order, and 367K with the inverted order, from which 112K lexical features and 1.7M collocation features after deleting those with one occurrence were extracted. In the task of IWSLT-04, we obtained 79.5k reordering examples with the straight order, 9.3k with the inverted order, from which 16.9K lexical features and 89.6K collocation features after deleting those with one occurrence were extracted. Finally, we ran the MaxEnt toolkit by Zhang [4] to tune the feature weights. We set iteration number to 100 and Gaussian prior to 1 for avoiding overfitting.

4.3 Results

We dropped unknown words (Koehn et al., 2005) of translations for both tasks before evaluating their BLEU scores. To be consistent with the official evaluation criterions of both tasks, case-sensitive BLEU-4 scores were computed For the NIST MT-05 task and case-insensitive BLEU-4 scores were computed for the IWSLT-04 task [5]. Experimental results on both tasks are shown in Table 1. Italic numbers refer to results for which the difference to the best result (indicated in bold) is not statistically significant. For all scores, we also show the 95% confidence intervals computed using Zhang's significant tester (Zhang et al., 2004) which was modified to conform to NIST's

[3] See http://www.cs.cmu.edu/ ashishv/mer.html. This is a Matlab implementation.

[4] See http://homepages.inf.ed.ac.uk/s0450736 /maxent_toolkit.html.

[5] Note that the evaluation criterion of IWSLT-04 is not totally matched since we didn't remove punctuation marks.

definition of the BLEU brevity penalty.

We observe that if phrasal reordering is totally dependent on the language model (*NONE*) we get the worst performance, even worse than the monotone search. This indicates that our language models were not strong to discriminate between straight orders and inverted orders. The flat and distortion reordering models (Row 3 and 4) show similar performance with Pharaoh. Although they are not dependent on phrases, they really reorder phrases with penalties to wrong orders supported by the language model and therefore outperform the monotone search. In row 6, only lexical features are used for the MaxEnt-based reordering model; while row 7 uses lexical features and collocation features. On both tasks, we observe that various reordering approaches show similar and stable performance ranks in different domains and the MaxEnt-based reordering models achieve the best performance among them. Using all features for the MaxEnt model (lex + col) is marginally better than using only lex features (lex).

4.4 Scaling to Large Bitexts

In the experiments described above, collocation features do not make great contributions to the performance improvement but make the total number of features increase greatly. This is a problem for MaxEnt parameter estimation if it is scaled to large bitexts. Therefore, for the integration of MaxEnt-based phrase reordering model in the system trained on large bitexts, we remove collocation features and only use lexical features from the last words of blocks (similar to those from the first words of blocks with similar performance). This time the bilingual training data contain 2.4M sentence pairs (68.1M Chinese words and 73.8M English words) and two trigram language models are used. One is trained on the English side of the bilingual training data. The other is trained on the Xinhua portion of the Gigaword corpus with 181.1M words. We also use some rules to translate numbers, time expressions and Chinese person names. The new Bleu score on NIST MT-05 is 0.291 which is very promising.

5 Discussion and Future Work

In this paper we presented a MaxEnt-based phrase reordering model for SMT. We used lexical features and collocation features from boundary words of blocks to predicate reorderings of neighbor blocks. Experiments on standard Chinese-English translation tasks from two different domains showed that our method achieves a significant improvement over the distortion/flat reordering models.

Systems	NIST MT-05	IWSLT-04
monotone	20.1 ± 0.8	37.8 ± 3.2
NONE	19.6 ± 0.8	36.3 ± 2.9
Distortion	20.9 ± 0.8	38.8 ± 3.0
Flat	20.5 ± 0.8	38.7 ± 2.8
Pharaoh	20.8 ± 0.8	38.9 ± 3.3
MaxEnt (lex)	*22.0* ± 0.8	*42.4* ± 3.3
MaxEnt (lex + col)	**22.2** ± 0.8	**42.8** ± 3.3

Table 1: BLEU-4 scores (%) with the 95% confidence intervals. Italic numbers refer to results for which the difference to the best result (indicated in bold) is not statistically significant.

Traditional distortion/flat-based SMT translation systems are good for learning phrase translation pairs, but learn nothing for phrasal reorderings from real-world data. This is our original motivation for designing a new reordering model, which can learn reorderings from training data just like learning phrasal translations. Lexicalized reordering model learns reorderings from training data, but it binds reorderings to individual concrete phrases, which restricts the model to reorderings of phrases seen in training data. On the contrary, the MaxEnt-based reordering model is not limited by this constraint since it is based on features of phrase, not phrase itself. It can be easily generalized to reorder unseen phrases provided that some features are fired on these phrases.

Another advantage of the MaxEnt-based reordering model is that it can take more features into reordering, even though they are non-independent. Tillmann et. al (2005) also use a MaxEnt model to integrate various features. The difference is that they use the MaxEnt model to predict not only orders but also blocks. To do that, it is necessary for the MaxEnt model to incorporate real-valued features such as the block translation probability and the language model probability. Due to the expensive computation, a local model is built. However, our MaxEnt model is just a module of the whole log-linear model of translation which uses its score as a real-valued feature. The modularity afforded by this design does not incur any computation problems, and make it easier to update one sub-model with other modules unchanged.

Beyond the MaxEnt-based reordering model, another feature deserving attention in our system is the CKY style decoder which observes the ITG. This is different from the work of Zens et. al. (2004). In their approach, translation is generated linearly, word by word and phrase by phrase in a traditional way with respect to the incorporation of the language model. It can be said that their decoder did not violate the ITG constraints but not that it observed the ITG. The ITG not only decreases reorderings greatly but also makes reordering hierarchical. Hierarchical reordering is more meaningful for languages which are organized hierarchically. From this point, our decoder is similar to the work by Chiang (2005).

The future work is to investigate other valuable features, e.g. binary features that explain blocks from the syntactical view. We think that there is still room for improvement if more contributing features are used.

Acknowledgements

This work was supported in part by National High Technology Research and Development Program under grant #2005AA114140 and National Natural Science Foundation of China under grant #60573188. Special thanks to Yajuan Lü for discussions of the manuscript of this paper and three anonymous reviewers who provided valuable comments.

References

Ashish Venugopal, Stephan Vogel. 2005. Considerations in Maximum Mutual Information and Minimum Classification Error training for Statistical Machine Translation. In the *Proceedings of EAMT-05*, Budapest, Hungary May 30-31.

Christoph Tillmann. 2004. A block orientation model for statistical machine translation. In *HLT-NAACL*, Boston, MA, USA.

Christoph Tillmann and Tong Zhang. 2005. A Localized Prediction Model for statistical machine translation. In *Proceedings of ACL 2005*, pages 557 – 564.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL 2005*, pages 263 – 270.

Dekai Wu. 1996. A Polynomial-Time Algorithm for Statistical Machine Translation. In *Proceedings of ACL 1996*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. Computational Linguistics, 23:377 – 404.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of ACL 2000*, pages 440 – 447.

Franz Josef Och. 2003a. Minimum error rate training in statistical machine translation. In *Proceedings of ACL 2003*, pages 160 – 167.

Franz Josef Och. 2003b. Statistical Machine Translation: From Single-Word Models to Alignment Templates Thesis.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. Computational Linguistics, 30:417 – 449.

Franz Josef Och, Ignacio Thayer, Daniel Marcu, Kevin Knight, Dragos Stefan Munteanu, Quamrul Tipu, Michel Galley, and Mark Hopkins. 2004. Arabic and Chinese MT at USC/ISI. Presentation given at NIST Machine Translation Evaluation Workshop.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP 2002*.

J. R. Quinlan. 1993. C4.5: progarms for machine learning. Morgan Kaufmann Publishers.

Kevin Knight. 1999. Decoding complexity in wordreplacement translation models. Computational Linguistics, Squibs & Discussion, 25(4).

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.

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT/NAACL*.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proceedings of the Sixth Conference of the Association for Machine Translation in the Americas*, pages 115 – 124.

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*.

R. Zens, H. Ney, T. Watanabe, and E. Sumita. 2004. Reordering Constraints for Phrase-Based Statistical Machine Translation. In *Proceedings of CoLing 2004*, Geneva, Switzerland, pp. 205-211.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL-2002)*.

Shankar Kumar and William Byrne. 2005. Local phrase reordering models for statistical machine translation. In *Proceedings of HLT-EMNLP*.

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 2004*, pages 2051 – 2054.

Distortion Models For Statistical Machine Translation

Yaser Al-Onaizan and Kishore Papineni
IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598, USA
{onaizan, papineni}@us.ibm.com

Abstract

In this paper, we argue that n-gram language models are not sufficient to address word reordering required for Machine Translation. We propose a new distortion model that can be used with existing phrase-based SMT decoders to address those n-gram language model limitations. We present empirical results in Arabic to English Machine Translation that show statistically significant improvements when our proposed model is used. We also propose a novel metric to measure word order similarity (or difference) between any pair of languages based on word alignments.

1 Introduction

A language model is a statistical model that gives a probability distribution over possible sequences of words. It computes the probability of producing a given word w_1 given all the words that precede it in the sentence. An n-gram language model is an n-th order Markov model where the probability of generating a given word depends only on the last $n-1$ words immediately preceding it and is given by the following equation:

$$P(w_1^k) = P(w_1)P(w_2|w_1)\cdots P(w_n|w_1^{n-1}) \quad (1)$$

where $k >= n$.

N-gram language models have been successfully used in Automatic Speech Recognition (ASR) as was first proposed by (Bahl et al., 1983). They play an important role in selecting among several candidate word realization of a given acoustic signal. N-gram language models have also been used in Statistical Machine Translation (SMT) as proposed by (Brown et al., 1990; Brown et al., 1993). The run-time search procedure used to find the most likely translation (or transcription in the case of Speech Recognition) is typically referred to as *decoding*.

There is a fundamental difference between decoding for machine translation and decoding for speech recognition. When decoding a speech signal, words are generated in the same order in which their corresponding acoustic signal is consumed. However, that is not necessarily the case in MT due to the fact that different languages have different word order requirements. For example, in Spanish and Arabic adjectives are mainly noun post-modifiers, whereas in English adjectives are noun pre-modifiers. Therefore, when translating between Spanish and English, words must usually be reordered.

Existing statistical machine translation decoders have mostly relied on language models to select the proper word order among many possible choices when translating between two languages. In this paper, we argue that a language model is not sufficient to adequately address this issue, especially when translating between languages that have very different word orders as suggested by our experimental results in Section 5. We propose a new distortion model that can be used as an additional component in SMT decoders. This new model leads to significant improvements in MT quality as measured by BLEU (Papineni et al., 2002). The experimental results we report in this paper are for Arabic-English machine translation of news stories.

We also present a novel method for measuring word order similarity (or differences) between any given pair of languages based on word alignments as described in Section 3.

The rest of this paper is organized as follows. Section 2 presents a review of related work. In Section 3 we propose a method for measuring the distortion between any given pair of languages. In Section 4, we present our proposed distortion model. In Section 5, we present some empirical results that show the utility of our distortion model for statistical machine translation systems. Then, we conclude this paper with a discussion in Section 6.

2 Related Work

Different languages have different word order requirements. SMT decoders attempt to generate translations in the proper word order by attempting many possible

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 529–536,
Sydney, July 2006. ©2006 Association for Computational Linguistics

word reorderings during the translation process. Trying all possible word reordering is an NP-Complete problem as shown in (Knight, 1999), which makes searching for the optimal solution among all possible permutations computationally intractable. Therefore, SMT decoders typically limit the number of permutations considered for efficiency reasons by placing reordering restrictions. Reordering restrictions for word-based SMT decoders were introduced by (Berger et al., 1996) and (Wu, 1996). (Berger et al., 1996) allow only reordering of at most n words at any given time. (Wu, 1996) propose using contiguity restrictions on the reordering. For a comparison and a more detailed discussion of the two approaches see (Zens and Ney, 2003).

A different approach to allow for a limited reordering is to reorder the input sentence such that the source and the target sentences have similar word order and then proceed to monotonically decode the reordered source sentence.

Monotone decoding translates words in the same order they appear in the source language. Hence, the input and output sentences have the same word order. Monotone decoding is very efficient since the optimal decoding can be found in polynomial time. (Tillmann et al., 1997) proposed a DP-based monotone search algorithm for SMT. Their proposed solution to address the necessary word reordering is to rewrite the input sentence such that it has a similar word order to the desired target sentence. The paper suggests that reordering the input reduces the translation error rate. However, it does not provide a methodology on how to perform this reordering.

(Xia and McCord, 2004) propose a method to automatically acquire rewrite patterns that can be applied to any given input sentence so that the rewritten source and target sentences have similar word order. These rewrite patterns are automatically extracted by parsing the source and target sides of the training parallel corpus. Their approach show a statistically-significant improvement over a phrase-based monotone decoder. Their experiments also suggest that allowing the decoder to consider some word order permutations *in addition to* the rewrite patterns already applied to the source sentence actually decreases the BLEU score.

Rewriting the input sentence whether using syntactic rules or heuristics makes hard decisions that can not be undone by the decoder. Hence, reordering is better handled during the search algorithm and as part of the optimization function.

Phrase-based monotone decoding does not directly address word order issues. Indirectly, however, the phrase dictionary[1] in phrase-based decoders typically captures local reorderings that were seen in the training data. However, it fails to generalize to word reorderings that were never seen in the training data. For example, a phrase-based decoder might translate the Arabic phrase *AlwlAyAt AlmtHdp*[2] correctly into English as *the United States* if it was seen in its training data, was aligned correctly, and was added to the phrase dictionary. However, if the phrase *Almmlkp AlmtHdp* is not in the phrase dictionary, it will not be translated correctly by a monotone phrase decoder even if the individual units of the phrase *Almmlkp* and *AlmtHdp*, and their translations (*Kingdom* and *United*, respectively) are in the phrase dictionary since that would require swapping the order of the two words.

(Och et al., 1999; Tillmann and Ney, 2003) relax the monotonicity restriction in their phrase-based decoder by allowing a restricted set of word reorderings. For their translation task, word reordering is done only for words belonging to the verb group. The context in which they report their results is a Speech-to-Speech translation from German to English.

(Yamada and Knight, 2002) propose a syntax-based decoder that restrict word reordering based on reordering operations on syntactic parse-trees of the input sentence. They reported results that are better than word-based IBM4-like decoder. However, their decoder is outperformed by phrase-based decoders such as (Koehn, 2004), (Och et al., 1999), and (Tillmann and Ney, 2003) . Phrase-based SMT decoders mostly rely on the language model to select among possible word order choices. However, in our experiments we show that the language model is not reliable enough to make the choices that lead to a better MT quality. This observation is also reported by (Xia and McCord, 2004).We argue that the distortion model we propose leads to a better translation as measured by BLEU.

Distortion models were first proposed by (Brown et al., 1993) in the so-called IBM Models. IBM Models 2 and 3 define the distortion parameters in terms of the word positions in the sentence pair, not the actual words at those positions. Distortion probability is also conditioned on the source and target sentence lengths. These models do not generalize well since their parameters are tied to absolute word position within sentences which tend to be different for the same words across sentences. IBM Models 4 and 5 alleviate this limitation by replacing absolute word positions with relative positions. The latter models define the distortion parameters for a cept (one or more words). This models phrasal movement better since words tend to move in blocks and not independently. The distortion is conditioned on classes of the aligned source and target words. The entire source and target vocabularies are reduced to a small number of classes (e.g., 50) for the purpose of estimating those parameters.

Similarly, (Koehn et al., 2003) propose a relative distortion model to be used with a phrase decoder. The model is defined in terms of the difference between the position of the current phrase and the position of the previous phrase in the source sentence. It does not con-

[1] Also referred to in the literature as the set of blocks or clumps.

[2] Arabic text appears throughout this paper in Tim Buckwalter's Romanization.

Arabic	Ezp$_1$ AbrAhym$_2$ ystqbl$_3$ ms&wlA$_4$ AqtSAdyA$_5$ sEwdyA$_6$ fy$_7$ bgdAd$_8$
English	Izzet$_1$ Ibrahim$_2$ Meets$_3$ Saudi$_4$ Trade$_5$ official$_6$ in$_7$ Baghdad$_8$
Word Alignment	(Ezp$_1$,Izzet$_1$) (AbrAhym$_2$,Ibrahim$_2$) (ystqbl$_3$,Meets$_3$) (ms&wlA$_4$,official$_6$) (AqtSAdyA$_5$,Trade$_5$) (sEwdyA$_6$,Saudi$_4$) (fy$_7$,in$_7$) (bgdAd$_8$,Baghdad$_8$)
Reordered English	Izzet$_1$ Ibrahim$_2$ Meets$_3$ official$_6$ Trade$_5$ Saudi$_4$ in$_7$ Baghdad$_8$

Table 1: Alignment-based word reordering. The indices are not part of the sentence pair, they are only used to illustrate word positions in the sentence. The indices in the reordered English denote word position in the original English order.

sider the words in those positions.

The distortion model we propose assigns a probability distribution over possible relative jumps conditioned on source words. Conditioning on the source words allows for a much more fine-grained model. For instance, words that tend to act as modifers (e.g., adjectives) would have a different distribution than verbs or nouns. Our model's parameters are directly estimated from word alignments as we will further explain in Section 4. We will also show how to generalize this word distortion model to a phrase-based model.

(Och et al., 2004; Tillman, 2004) propose orientation-based distortion models lexicalized on the phrase level. There are two important distinctions between their models and ours. First, they lexicalize their model on the phrases, which have many more parameters and hence would require much more data to estimate reliably. Second, their models consider only the direction (i.e., orientation) and not the relative jump.

We are not aware of any work on measuring word order differences between a given language pair in the context of statistical machine translation.

3 Measuring Word Order Similarity Between Two Language

In this section, we propose a simple, novel method for measuring word order similarity (or differences) between any given language pair. This method is based on word-alignments and the BLEU metric.

We assume that we have word-alignments for a set of sentence pairs. We first reorder words in the target sentence (e.g., English when translating from Arabic to English) according to the order in which they are aligned to the source words as shown in Table 1. If a target word is not aligned, then, we assume that it is aligned to the same source word that the preceding aligned target word is aligned to.

Once the reordered target (here English) sentences are generated, we measure the distortion between the language pair by computing the BLEU[3] score between the original target and reordered target, treating the original target as the reference.

Table 2 shows these scores for Arabic-English and Chinese-English. The word alignments we use are both annotated manually by human annotators. The Arabic-English test set is the NIST MT Evaluation 2003 test set. It contains 663 segments (i.e., sentences). The Arabic side consists of 16,652 tokens and the English consists of 19,908 tokens. The Chinese-English test set contains 260 segments. The Chinese side is word segmented and consists of 4,319 tokens and the English consists of 5,525 tokens.

As suggested by the BLEU scores reported in Table 2, Arabic-English has more word order differences than Chinese-English. The difference in n-gPrec is bigger for smaller values of n, which suggests that Arabic-English has more local word order differences than in Chinese-English.

4 Proposed Distortion Model

The distortion model we are proposing consists of three components: **outbound**, **inbound**, and **pair** distortion. Intuitively our distortion models attempt to capture the order in which source words need to be translated. For instance, the outbound distortion component attempts to capture what is typically translated immediately after the word that has just been translated. Do we tend to translate words that precede it or succeed it? Which word position to translate next?

Our distortion parameters are directly estimated from word alignments by simple counting over alignment links in the training data. Any aligner such as (Al-Onaizan et al., 1999) or (Vogel et al., 1996) can be used to obtain word alignments. For the results reported in this paper word alignments were obtained using a maximum-posterior word aligner[4] described in (Ge, 2004).

We will illustrate the components of our model with a partial word alignment. Let us assume that our source sentence[5] is $(f_{10}, f_{250}, f_{300})$[6], and our target sentence is (e_{410}, e_{20}), and their word alignment is $a = ((f_{10}, e_{410}), (f_{300}, e_{20}))$. Word Alignment a can

[3] the BLEU scores reported throughout this paper are for case-sensitive BLEU. The number of references used is also reported (e.g., BLEUr1n4c: $r1$ means 1 reference, $n4$ means upto 4-gram are considred, c means case sensitive).

[4] We also estimated distortion parameters using a Maximum Entropy aligner and the differences were negligible.

[5] In practice, we add special symbols at the start and end of the source and target sentences, we also assume that the start symbols in the source and target are aligned, and similarly for the end symbols. Those special symbols are omitted in our example for ease of presentation.

[6] The indices here represent source and target vocabulary ids.

N-gram Precision	Arabic-English	Chinese-English
1-gPrec	1	1
2-gPrec	0.6192	0.7378
3-gPrec	0.4547	0.5382
4-gPrec	0.3535	0.3990
5-gPrec	0.2878	0.3075
6-gPrec	0.2378	0.2406
7-gPrec	0.1977	0.1930
8-gPrec	0.1653	0.1614
9-gPrec	0.1380	0.1416
BLEUr1n4c	0.3152	0.3340
95% Confidence σ	0.0180	0.0370

Table 2: Word order similarity for two language pairs: Arabic-English and Chinese-English. n-gPrec is the n-gram precision as defined in BLEU.

be rewritten as $a_1 = 1$ and $a_2 = 3$ (i.e., the second target word is aligned to the third source word). From this partial alignment we increase the counts for the following outbound, inbound, and pair distortions: $P_o(\delta = +2|f_{10})$, $P_i(\delta = +2|f_{300})$. and $P_p(\delta = +2|f_{10}, f_{300})$.

Formally, our distortion model components are defined as follows:

Outbound Distortion:

$$P_o(\delta|f_i) = \frac{C(\delta|f_i)}{\sum_k C(\delta_k\,|f_i)} \tag{2}$$

where f_i is a foreign word (i.e., Arabic in our case), δ is the step size, and $C(\delta|f_i)$ is the observed count of this parameter over all word alignments in the training data. The value for δ, in theory, ranges from $-max$ to $+max$ (where max is the maximum source sentence length observed), but in practice only a small number of those step sizes are observed in the training data, and hence, have non-zero value).

Inbound Distortion:

$$P_i(\delta|f_j) = \frac{C(\delta|f_j)}{\sum_k C(\delta_k|f_j)} \tag{3}$$

Pairwise Distortion:

$$P_p(\delta|f_i, f_j) = \frac{C(\delta|f_i, f_j)}{\sum_k C(\delta_k|f_i, f_j)} \tag{4}$$

In order to use these probability distributions in our decoder, they are then turned into costs. The outbound distortion cost is defined as:

$$C_o(\delta|f_i) = \log\left\{\alpha P_o(\delta|f_i) + (1-\alpha)P_s(\delta)\right\} \tag{5}$$

where $P_s(\delta)$ is a smoothing distribution [7] and α is a linear-mixture parameter [8].

The inbound and pair costs ($C_i(\delta|f_i)$ and $C_p(\delta|f_i, f_j)$) can be defined in a similar fashion.

So far, our distortion cost is defined in terms of words, not phrases. Therefore, we need to generalize the distortion cost in order to use it in a phrase-based decoder. This generalization is defined in terms of the internal word alignment within phrases (we used the Viterbi word alignment). We illustrate this with an example: Suppose the last position translated in the source sentence so far is n and we are to cover a source phrase *p=wlAyp wA$nTn* that begins at position m in the source sentence. Also, suppose that our phrase dictionary provided the translation *Washington State*, with internal word alignment $a = (a_1 = 2, a_2 = 1)$ (i.e., *a*=(<*Washington,wA$nTn*>, <*State,wlAyp*>), then the outbound phrase cost is defined as:

$$\begin{aligned} C_o(p, n, m, a) =& C_o(\delta = (m-n)|f_n) + \\ & \sum_{i=1}^{l-1} C_o(\delta = (a_{i+1} - a_i)\,|f_{a_i}) \end{aligned} \tag{6}$$

where l is the length of the target phrase, a is the internal word alignment, f_n is source word at position n (in the sentence), and f_{a_i} is the source word that is aligned to the i-th word in the target side of the phrase (not the sentence).

The inbound and pair distortion costs (i..e, $C_i(p, n, m, a)$ and $C_p(p, n, m, a)$) can be defined in a similar fashion.

The above distortion costs are used in conjunction with other cost components used in our decoder. The ultimate word order choice made is influenced by both the language model cost as well as the distortion cost.

5 Experimental Results

The phrase-based decoder we use is inspired by the decoder described in (Tillmann and Ney, 2003) and similar to that described in (Koehn, 2004). It is a multi-stack, multi-beam search decoder with n stacks (where n is the length of the source sentence being decoded)

[7] The smoothing we use is a geometrically decreasing distribution as the step size increases.

[8] For the experiments reported here we use $\alpha = 0.1$, which is set empirically.

s	0	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4	4	4	4	4
w	0	4	6	8	10	12	4	6	8	10	12	4	6	8	10	12	4	6	8	10	12
BLEUr1n4c	0.5617	0.6507	0.6443	0.6430	0.6461	0.6456	0.6831	0.6706	0.6609	0.6596	0.6626	**0.6919**	0.6751	0.6580	0.6505	0.6490	0.6851	0.6592	0.6317	0.6237	**0.6081**

Table 3: BLEU scores for the word order restoration task. The BLEU scores reported here are with 1 reference. The input is the reordered English in the reference. The 95% Confidence σ ranges from 0.011 to 0.016

and a beam associated with each stack as described in (Al-Onaizan, 2005). The search is done in n time steps. In time step i, only hypotheses that cover exactly i source words are extended. The beam search algorithm attempts to find the translation (i.e., hypothesis that covers all source words) with the minimum cost as in (Tillmann and Ney, 2003) and (Koehn, 2004) . The distortion cost is added to the log-linear mixture of the hypothesis extension in a fashion similar to the language model cost.

A hypothesis covers a subset of the source words. The final translation is a hypothesis that covers all source words and has the minimum cost among all possible [9] hypotheses that cover all source words. A hypothesis h is extended by matching the phrase dictionary against source word sequences in the input sentence that are not covered in h. The cost of the new hypothesis $C(h_{new}) = C(h) + C(e)$, where $C(e)$ is the cost of this extension. The main components of the cost of extension e can be defined by the following equation:

$$C(e) = \lambda_1 C_{LM}(e) + \lambda_2 C_{TM}(e) + \lambda_3 C_D(e)$$

where $C_{LM}(e)$ is the language model cost, $C_{TM}(e)$ is the translation model cost, and $C_D(e)$ is the distortion cost. The extension cost depends on the hypothesis being extended, the phrase being used in the extension, and the source word positions being covered.

The word reorderings that are explored by the search algorithm are controlled by two parameters s and w as described in (Tillmann and Ney, 2003). The first parameter s denotes the number of source words that are temporarily skipped (i.e., temporarily left uncovered) during the search to cover a source word to the right of the skipped words. The second parameter is the window width w, which is defined as the distance (in number of source words) between the left-most uncovered source word and the right-most covered source word.

To illustrate these restrictions, let us assume the input sentence consists of the following sequence (f_1, f_2, f_3, f_4). For s=1 and w=2, the permissible permutations are (f_1, f_2, f_3, f_4), (f_2, f_1, f_3, f_4), (f_2, f_3, f_1, f_4), (f_1, f_3, f_2, f_4),(f_1, f_3, f_4, f_2), and (f_1, f_2, f_4, f_3).

5.1 Experimental Setup

The experiments reported in this section are in the context of SMT from Arabic into English. The training data is a 500K sentence-pairs subsample of the 2005 Large Track Arabic-English Data for NIST MT Evaluation.

The language model used is an interpolated trigram model described in (Bahl et al., 1983). The language model is trained on the LDC English GigaWord Corpus.

The test set used in the experiments in this section is the 2003 NIST MT Evaluation test set (which is not part of the training data).

5.2 Reordering with Perfect Translations

In the experiments in this section, we show the utility of a trigram language model in restoring the correct word order for English. The task is a simplified translation task, where the input is reordered English (English written in Arabic word order) and the output is English in the correct order. The source sentence is a reordered English sentence in the same manner we described in Section 3. The objective of the decoder is to recover the correct English order.

We use the same phrase-based decoder we use for our SMT experiments, except that only the language model cost is used here. Also, the phrase dictionary used is a one-to-one function that maps every English word in our vocabulary to itself. The language model we use for the experiments reported here is the same as the one used for other experiments reported in this paper.

The results in Table 3 illustrate how the language model performs reasonably well for local reorderings (e.g., for $s = 3$ and $w = 4$), but its perfromance deteriorates as we relax the reordering restrictions by increasing the reordering window size (w).

Table 4 shows some examples of original English, English in Arabic order, and the decoder output for two different sets of reordering parameters.

5.3 SMT Experiments

The phrases in the phrase dictionary we use in the experiments reported here are a combination

[9] Exploring all possible hypothesis with all possible word permutations is computationally intractable. Therefore, the search algorithm gives an approximation to the optimal solution. *All possible hypotheses* refers to all hypotheses that were explored by the decoder.

Eng_Ar	Opposition Iraqi Prepares for Meeting mid - January in Kurdistan
Orig. Eng.	Iraqi Opposition Prepares for mid - January Meeting in Kurdistan
Output1	Iraqi Opposition Meeting Prepares for mid - January in Kurdistan
Output2	Opposition Meeting Prepares for Iraqi Kurdistan in mid - January

Eng_Ar	Head of Congress National Iraqi Visits Kurdistan Iraqi
Orig. Eng.	Head of Iraqi National Congress Visits Iraqi Kurdistan
Output1	Head of Iraqi National Congress Visits Iraqi Kurdistan
Output2	Head Visits Iraqi National Congress of Iraqi Kurdistan

Eng_Ar	House White Confirms Presence of Tape New Bin Laden
Orig. Eng.	White House Confirms Presence of New Bin Laden Tape
Output1	White House Confirms Presence of Bin Laden Tape New
Output2	White House of Bin Laden Tape Confirms Presence New

Table 4: Examples of reordering with perfect translations. The examples show English in Arabic order (Eng_Ar.), English in its original order (Orig. Eng.) and decoding with two different parameter settings. Output1 is decoding with (s=3,w=4). Output2 is decoding with (s=4,w=12). The sentence lengths of the examples presented here are much shorter than the average in our test set (~ 28.5).

s	w	Distortion Used?	BLEUr4n4c
0	0	NO	**0.4468**
1	8	NO	0.4346
1	8	YES	0.4715
2	8	NO	0.4309
2	8	YES	0.4775
3	8	NO	0.4283
3	8	YES	**0.4792**
4	8	NO	0.4104
4	8	YES	0.4782

Table 5: BLEU scores for the Arabic-English machine translation task. The 95% Confidence σ ranges from 0.0158 to 0.0176. s is the number of words temporarily skipped, and w is the word permutation window size.

of phrases automatically extracted from maximum-posterior alignments and maximum entropy alignments. Only phrases that conform to the so-called consistent alignment restrictions (Och et al., 1999) are extracted.

Table 5 shows BLEU scores for our SMT decoder with different parameter settings for skip s, window width w, with and without our distortion model. The BLEU scores reported in this table are based on 4 reference translations. The language model, phrase dictionary, and other decoder tuning parameters remain the same in all experiments reported in this table.

Table 5 clearly shows that as we open the search and consider wider range of word reorderings, the BLEU score decreases in the absence of our distortion model when we rely solely on the language model. Wrong reorderings look attractive to the decoder via the language model which suggests that we need a richer model with more parameter. In the absence of richer models such as the proposed distortion model, our results suggest that it is best to decode monotonically and only allow local reorderings that are captured in our phrase dictionary.

However, when the distortion model is used, we see statistically significant increases in the BLEU score as we consider more word reorderings. The best BLEU score achieved when using the distortion model is 0.4792 , compared to a best BLEU score of 0.4468 when the distortion model is not used.

Our results on the 2004 and 2005 NIST MT Evaluation test sets using the distortion model are 0.4497 and 0.4646[10], respectively.

Table 6 shows some Arabic-English translation examples using our decoder with and without the distortion model.

6 Conclusion and Future Work

We presented a new distortion model that can be integrated with existing phrase-based SMT decoders. The proposed model shows statistically significant improvement over a state-of-the-art phrase-based SMT decoder. We also showed that n-gram language mod-

[10] The MT05 BLEU score is the from the official NIST evaluation. The MT04 BLEU score is only our second run on MT04.

Input (Ar)	kwryA Al$mAlyp mstEdp llsmAH lwA$nTn bAltHqq mn AnhA lA tSnE AslHp nwwyp
Ref. (En)	North Korea Prepared to allow Washington to check it is not Manufacturing Nuclear Weapons
Out1	North Korea to Verify Washington That It Was Not Prepared to Make Nuclear Weapons
Out2	North Korea Is Willing to Allow Washington to Verify It Does Not Make Nuclear Weapons
Input (Ar)	wAkd AldblwmAsy An ”AnsHAb (kwryA Al$mAlyp mn AlmEAhdp) ybd> AEtbArA mn Alywm”.
Ref. (En)	The diplomat confirmed that ”North Korea’s withdrawal from the treaty starts as of today.”
Out1	The diplomat said that ” the withdrawal of the Treaty (start) North Korea as of today. ”
Out2	The diplomat said that the ” withdrawal of (North Korea of the treaty) will start as of today ”.
Input (Ar)	snrfE *lk AmAm Almjls Aldstwry”.
Ref. (En)	We will bring this before the Constitutional Assembly.”
Out1	The Constitutional Council to lift it. ”
Out2	This lift before the Constitutional Council ”.
Input (Ar)	wAkd AlbrAdEy An mjls AlAmn ”ytfhm” An 27 kAnwn AlvAny/ynAyr lys mhlp nhA}yp.
Ref. (En)	Baradei stressed that the Security Council ”appreciates” that January 27 is not a final ultimatum.
Out1	Elbaradei said that the Security Council ” understand ” that is not a final period January 27.
Out2	Elbaradei said that the Security Council ” understand ” that 27 January is not a final period.

Table 6: Selected examples of our Arabic-English SMT output. The English is one of the human reference translations. Output 1 is decoding without the distortion model and (s=4, w=8), which corresponds to 0.4104 BLEU score. Output 2 is decoding with the distortion model and (s=3, w=8), which corresponds to 0.4792 BLEU score. The sentences presented here are much shorter than the average in our test set. The average length of the arabic sentence in the MT03 test set is ~ 24.7.

els are not sufficient to model word movement in translation. Our proposed distortion model addresses this weakness of the n-gram language model.

We also propose a novel metric to measure word order similarity (or differences) between any pair of languages based on word alignments. Our metric shows that Chinese-English have a closer word order than Arabic-English.

Our proposed distortion model relies solely on word alignments and is conditioned on the source words. The majority of word movement in translation is mainly due to syntactic differences between the source and target language. For example, Arabic is verb-initial for the most part. So, when translating into English, one needs to move the verb after the subject, which is often a long compounded phrase. Therefore, we would like to incorporate syntactic or part-of-speech information in our distortion model.

Acknowledgment

This work was partially supported by DARPA GALE program under contract number HR0011-06-2-0001. It was also partially supported by DARPA TIDES program monitored by SPAWAR under contract number N66001-99-2-8916.

References

Yaser Al-Onaizan, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz-Josef Och, David Purdy, Noah Smith, and David Yarowsky. 1999. Statistical Machine Translation: Final Report, Johns Hopkins University Summer Workshop (WS 99) on Language Engineering, Center for Language and Speech Processing, Baltimore, MD.

Yaser Al-Onaizan. 2005. IBM Arabic-to-English MT Submission. *Presentation given at DARPA/TIDES NIST MT Evaluation workshop*.

Lalit R. Bahl, Frederick Jelinek, and Robert L. Mercer. 1983. A Maximum Likelihood Approach to Continuous Speech Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):179–190.

Adam L. Berger, Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Andrew S. Kehler, and Robert L. Mercer. 1996. Language Translation Apparatus and Method of Using Context-Based Translation Models. *United States Patent, Patent Number 5510981*, April.

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Frederick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85.

Peter F. Brown, 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.

Niyu Ge. 2004. Improvements in Word Alignments. *Presentation given at DARPA/TIDES NIST MT Evaluation workshop*.

Kevin Knight. 1999. Decoding Complexity in Word-Replacement Translation Models. *Computational Linguistics*, 25(4):607–615.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In Marti Hearst and Mari Ostendorf, editors, *HLT-NAACL 2003: Main Proceedings*, pages 127–133, Edmonton, Alberta, Canada, May 27 – June 1. Association for Computational Linguistics.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-Based Statistical Machine Translation Models. In *Proceedings of the 6th Conference of the Association for Machine Translation in the Americas*, pages 115–124, Washington DC, September-October. The Association for Machine Translation in the Americas (AMTA).

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved Alignment Models for Statistical Machine Translation. In *Joint Conf. of Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28, College Park, Maryland.

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 Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 161–168, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of machine translation. In *40th Annual Meeting of the Association for Computational Linguistics (ACL 02)*, pages 311–318, Philadelphia, PA, July.

Christoph Tillman. 2004. A unigram orientation model for statistical machine translation. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Short Papers*, pages 101–104, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Christoph Tillmann and Hermann Ney. 2003. Word Re-ordering and a DP Beam Search Algorithm for Statistical Machine Translation. *Computational Linguistics*, 29(1):97–133.

Christoph Tillmann, Stephan Vogel, Hermann Ney, and Alex Zubiaga. 1997. A DP-Based Search Using Monotone Alignments in Statistical Translation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 289–296, Madrid. Association for Computational Linguistics.

Stefan Vogel, Hermann Ney, and Christoph Tillmann. 1996. HMM-BasedWord Alignment in Statistical Machine Translation. In *Proc. of the 16th Int. Conf. on Computational Linguistics (COLING 1996)*, pages 836–841, Copenhagen, Denmark, August.

Dekai Wu. 1996. A Polynomial-Time Algorithm for Statistical Machine Translation. In *Proc. of the 34th Annual Conf. of the Association for Computational Linguistics (ACL 96)*, pages 152–158, Santa Cruz, CA, June.

Fei Xia and Michael McCord. 2004. Improving a Statistical MT System with Automatically Learned Rewrite Patterns. In *Proc. of the 20th International Conference on Computational Linguistics (COLING 2004)*, Geneva, Switzerland.

Kenji Yamada and Kevin Knight. 2002. A Decoder for Syntax-based Statistical MT. In *Proc. of the 40th Annual Conf. of the Association for Computational Linguistics (ACL 02)*, pages 303–310, Philadelphia, PA, July.

Richard Zens and Hermann Ney. 2003. A Comparative Study on Reordering Constraints in Statistical Machine Translation. In Erhard Hinrichs and Dan Roth, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 144–151, Sapporo, Japan.

A Study on Automatically Extracted Keywords in Text Categorization

Anette Hulth and **Beáta B. Megyesi**
Department of Linguistics and Philology
Uppsala University, Sweden
anette.hulth@gmail.com bea@stp.lingfil.uu.se

Abstract

This paper presents a study on if and how automatically extracted keywords can be used to improve text categorization. In summary we show that a higher performance — as measured by micro-averaged F-measure on a standard text categorization collection — is achieved when the full-text representation is combined with the automatically extracted keywords. The combination is obtained by giving higher weights to words in the full-texts that are also extracted as keywords. We also present results for experiments in which the keywords are the only input to the categorizer, either represented as unigrams or intact. Of these two experiments, the unigrams have the best performance, although neither performs as well as headlines only.

1 Introduction

Automatic text categorization is the task of assigning any of a set of predefined categories to a document. The prevailing approach is that of *supervised machine learning*, in which an algorithm is trained on documents with known categories. Before any learning can take place, the documents must be represented in a form that is understandable to the learning algorithm. A trained *prediction model* is subsequently applied to previously unseen documents, to assign the categories. In order to perform a text categorization task, there are two major decisions to make: how to represent the text, and what learning algorithm to use to create the prediction model. The decision about the representation is in turn divided into two subquestions: what features to select as input and which type of value to assign to these features.

In most studies, the best performing representation consists of the full length text, keeping the tokens in the document separate, that is as unigrams. In recent years, however, a number of experiments have been performed in which richer representations have been evaluated. For example, Caropreso et al. (2001) compare unigrams and bigrams; Moschitti et al. (2004) add complex nominals to their bag-of-words representation, while Kotcz et al. (2001), and Mihalcea and Hassan (2005) present experiments where automatically extracted sentences constitute the input to the representation. Of these three examples, only the sentence extraction seems to have had any positive impact on the performance of the automatic text categorization.

In this paper, we present experiments in which keywords, that have been automatically extracted, are used as input to the learning, both on their own and in combination with a full-text representation. That the keywords are extracted means that the selected terms are present verbatim in the document. A keyword may consist of one or several tokens. In addition, a keyword may well be a whole expression or phrase, such as *snakes and ladders*. The main goal of the study presented in this paper is to investigate if automatically extracted keywords can improve automatic text categorization. We investigate what impact keywords have on the task by predicting text categories on the basis of keywords only, and by combining full-text representations with automatically extracted keywords. We also experiment with different ways of representing keywords, either as unigrams or intact. In addition, we investigate the effect of using the headlines — represented as unigrams — as input,

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 537–544,
Sydney, July 2006.

to compare their performance to that of the keywords.

The outline of the paper is as follows: in Section 2, we present the algorithm used to automatically extract the keywords. In Section 3, we present the corpus, the learning algorithm, and the experimental setup for the performed text categorization experiments. In Section 4, the results are described. An overview of related studies is given in Section 5, and Section 6 concludes the paper.

2 Selecting the Keywords

This section describes the method that was used to extract the keywords for the text categorization experiments discussed in this paper. One reason why this method, developed by Hulth (2003; 2004), was chosen is because it is tuned for short texts (more specifically for scientific journal abstracts). It was thus suitable for the corpus used in the described text categorization experiments.

The approach taken to the automatic keyword extraction is that of supervised machine learning, and the prediction models were trained on manually annotated data. No new training was done on the text categorization documents, but models trained on other data were used. As a first step to extract keywords from a document, candidate terms are selected from the document in three different manners. One term selection approach is statistically oriented. This approach extracts all uni-, bi-, and trigrams from a document. The two other approaches are of a more linguistic character, utilizing the words' parts-of-speech (PoS), that is, the word class assigned to a word. One approach extracts all noun phrase (NP) chunks, and the other all terms matching any of a set of empirically defined PoS patterns (frequently occurring patterns of manual keywords). All candidate terms are stemmed.

Four features are calculated for each candidate term: term frequency; inverse document frequency; relative position of the first occurrence; and the PoS tag or tags assigned to the candidate term. To make the final selection of keywords, the three predictions models are combined. Terms that are subsumed by another keyword selected for the document are removed. For each selected stem, the most frequently occurring unstemmed form in the document is presented as a keyword. Each document is assigned at the most twelve keywords, provided that the added regression value (given by the prediction models) is higher than an empirically defined threshold value. To avoid that a document gets no keywords, at least three keywords are assigned although the added regression value is below the threshold (provided that there are at least three candidate terms).

Assign. mean	Corr. mean	P	R	F
8.6	3.6	41.5	46.9	44.0

Table 1: The number of assigned (Assign.) keywords in mean per document; the number of correct (Corr.) keywords in mean per document; precision (P); recall (R); and F-measure (F), when 3–12 keywords are extracted per document.

In Hulth (2004) an evaluation on 500 abstracts in English is presented. For the evaluation, keywords assigned to the test documents by professional indexers are used as a gold standard, that is, the manual keywords are treated as the one and only truth. The evaluation measures are *precision* (how many of the automatically assigned keywords that are also manually assigned keywords) and *recall* (how many of the manually assigned keywords that are found by the automatic indexer). The third measure used for the evaluations is the *F-measure* (the harmonic mean of precision and recall). Table 1 shows the result on that particular test set. This result may be considered to be state-of-the-art.

3 Text Categorization Experiments

This section describes in detail the four experimental settings for the text categorization experiments.

3.1 Corpus

For the text categorization experiments we used the *Reuters-21578 corpus*, which contains 20 000 newswire articles in English with multiple categories (Lewis, 1997). More specifically, we used the *ModApte* split, containing 9 603 documents for training and 3 299 documents in the fixed test set, and the 90 categories that are present in both training and test sets.

As a first pre-processing step, we extracted the texts contained in the TITLE and BODY tags. The pre-processed documents were then given as input to the keyword extraction algorithm. In Table 2, the number of keywords assigned to the documents in the training set and the test set are displayed. As can be seen in this table, three is the number of keywords that is most often extracted. In the training data set, 9 549 documents are assigned keywords, while 54 are empty, as they have no text in the TITLE or BODY tags. Of the 3 299 documents in the test set, 3 285 are assigned keywords, and the remaining fourteen are those that are empty. The empty documents are included in the result calculations for the fixed test set, in order to enable comparisons with other experiments. The mean number of keyword extracted per document in the training set is 6.4 and in the test set 6.1 (not counting the empty documents).

Keywords	**Training docs**	**Test docs**
0	54	14
1	68	36
2	829	272
3	2 016	838
4	868	328
5	813	259
6	770	252
7	640	184
8	527	184
9	486	177
10	688	206
11	975	310
12	869	239

Table 2: Number of automatically extracted keywords per document in training set and test set respectively.

3.2 Learning Method

The focus of the experiments described in this paper was the text representation. For this reason, we used only one learning algorithm, namely an implementation of *Linear Support Vector Machines* (Joachims, 1999). This is the learning method that has obtained the best results in text categorization experiments (Dumais et al., 1998; Yang and Liu, 1999).

3.3 Representations

This section describes in detail the input representations that we experimented with. An important step for the feature selection is the dimensionality reduction, that is reducing the number of features. This can be done by removing words that are rare (that occur in too few documents or have too low term frequency), or very common (by applying a stop-word list). Also, terms may be stemmed, meaning that they are merged into a common form. In addition, any of a number of feature selection metrics may be applied to further reduce the space, for example chi-square, or information gain (see for example Forman (2003) for a survey).

Once that the features have been set, the final decision to make is what feature value to assign. There are to this end three common possibilities: a boolean representation (that is, the term exists in the document or not), term frequency, or tf*idf.

Two sets of experiments were run in which the automatically extracted keywords were the only input to the representation. In the first set, keywords that contained several tokens were kept intact. For example a keyword such as *paradise fruit* was represented as `paradise_fruit` and was — from the point of view of the classifier — just as distinct from the single token *fruit* as from *meat-packers*. No stemming was performed in this set of experiments.

In the second set of keywords-only experiments, the keywords were split up into unigrams, and also stemmed. For this purpose, we used Porter's stemmer (Porter, 1980). Thereafter the experiments were performed identically for the two keyword representations.

In a third set of experiments, we extracted only the content in the TITLE tags, that is, the headlines. The tokens in the headlines were stemmed and represented as unigrams. The main motivation for the title experiments was to compare their performance to that of the keywords.

For all of these three feature inputs, we first evaluated which one of the three possible feature values to use (boolean, tf, or tf*idf). Thereafter, we reduced the space by varying the minimum number of occurrences in the training data, for a feature to be kept.

The starting point for the fourth set of experiments was a full-text representation, where all stemmed unigrams occurring three or more times in the training data were selected, with the feature value tf*idf. Assuming that extracted keywords convey information about a document's gist, the feature values in the full-text representation were given higher weights if the feature was identical to a keyword token. This was achieved by adding the term frequency of a full-text unigram to the term frequency of an identical keyword unigram. Note that this does not mean that the term frequency value was necessarily doubled, as a keyword often contains more than one token, and it was the term frequency of the whole keyword that was added.

3.4 Training and Validation

This section describes the parameter tuning, for which we used the training data set. This set was divided into five equally sized folds, to decide which setting of the following two parameters that resulted in the best performing classifier: what feature value to use, and the threshold for the minimum number of occurrence in the training data (in this particular order).

To obtain a baseline, we made a full-text unigram run with boolean as well as with tf*idf feature values, setting the occurrence threshold to three.

As stated previously, in this study, we were concerned only with the representation, and more specifically with the feature input. As we did not tune any other parameters than the two mentioned above, the results can be expected to be lower than the state-of-the art, even for the full-text run with unigrams.

The number of input features for the full-text unigram representation for the whole training set was 10 676, after stemming and removing all tokens that contained only digits, as well as those tokens that occurred less than three times. The total number of keywords assigned to the 9 603 documents in the training data was 61 034. Of these were 29 393 unique. When splitting up the keywords into unigrams, the number of unique stemmed tokens was 11 273.

3.5 Test

As a last step, we tested the best performing representations in the four different experimental settings on the independent test set.

The number of input features for the full-text unigram representation was 10 676. The total number of features for the intact keyword representation was 4 450 with the occurrence threshold set to three, while the number of stemmed keyword unigrams was 6 478, with an occurrence threshold of two. The total number of keywords extracted from the 3 299 documents in the test set was 19 904.

Next, we present the results for the validation and test procedures.

4 Results

To evaluate the performance, we used *precision*, *recall*, and *micro-averaged F-measure*, and we let the F-measure be decisive. The results for the 5-fold cross validation runs are shown in Table 3, where the values given are the average of the five runs made for each experiment. As can be seen in this table, the full-text run with a boolean feature value gave 92.3% precision, 69.4% recall, and 79.2% F-measure. The full-text run with tf*idf gave a better result as it yielded 92.9% precision, 71.3% recall, and 80.7% F-measure. Therefore we defined the latter as baseline.

In the first type of the experiment where each keyword was treated as a feature independently of the number of tokens contained, the recall rates were considerably lower (between 32.0% and 42.3%) and the precision rates were somewhat lower (between 85.8% and 90.5%) compared to the baseline. The best performance was obtained when using a boolean feature value, and setting the minimum number of occurrence in training data to three (giving an F-measure of 56.9%).

In the second type of experiments, where the keywords were split up into unigrams and stemmed, recall was higher but still low (between 60.2% and 64.8%) and precision was somewhat lower (88.9–90.2%) when compared to the baseline. The best results were achieved with a boolean representation (similar to the first experiment) and the minimum number of occurrence in the training data set to two (giving an F-measure of 75.0%)

In the third type of experiments, where only the text in the TITLE tags was used and was represented as unigrams and stemmed, precision rates increased above the baseline to 93.3–94.5%. Here, the best representation was tf*idf with a token occurring at least four times in the training data (with an F-measure of 79.9%).

In the fourth and last set of experiments, we gave higher weights to full-text tokens if the same token was present in an automatically extracted keyword. Here we obtained the best results. In these experiments, the term frequency of a keyword unigram was added to the term frequency for the full-text features, whenever the stems were identical. For this representation, we experimented with setting the minimum number of occurrence in training data both before and after that the term frequency for the keyword token was added to the term frequency of the unigram. The

Input feature	Feature value	Min. occurrence	Precision	Recall	F-measure
full-text unigram	bool	3	92.31	69.40	79.22
full-text unigram	tf*idf	3	92.89	71.30	80.67
keywords-only intact	bool	1	90.54	36.64	52.16
keywords-only intact	tf	1	88.68	33.74	48.86
keywords-only intact	tf*idf	1	89.41	32.05	47.18
keywords-only intact	bool	2	89.27	40.43	55.64
keywords-only intact	bool	3	87.11	42.28	56.90
keywords-only intact	bool	4	85.81	41.97	56.35
keywords-only unigram	bool	1	89.12	64.61	74.91
keywords-only unigram	tf	1	89.89	60.23	72.13
keywords-only unigram	tf*idf	1	90.17	60.36	72.31
keywords-only unigram	bool	2	89.02	64.83	75.02
keywords-only unigram	bool	3	88.90	64.82	74.97
title	bool	1	94.17	68.17	79.08
title	tf	1	94.37	67.89	78.96
title	tf*idf	1	**94.46**	68.49	79.40
title	tf*idf	2	93.92	69.19	79.67
title	tf*idf	3	93.75	69.65	79.91
title	tf*idf	4	93.60	69.74	79.92
title	tf*idf	5	93.31	69.40	79.59
keywords+full	tf*idf	3 (before adding)	92.73	**72.02**	**81.07**
keywords+full	tf*idf	3 (after adding)	92.75	71.94	81.02

Table 3: The average results from 5-fold cross validations for the baseline candidates and the four types of experiments, with various parameter settings.

highest recall (72.0%) and F-measure (81.1%) for all validation runs were achieved when the occurrence threshold was set before the addition of the keywords.

Next, the results on the fixed test data set for the four experimental settings with the best performance on the validation runs are presented.

Table 4 shows the results obtained on the fixed test data set for the baseline and for those experiments that obtained the highest F-measure for each one of the four experiment types.

We can see that the baseline — where the full-text is represented as unigrams with tf*idf as feature value — yields 93.0% precision, 71.7% recall, and 81.0% F-measure. When the intact keywords are used as feature input with a boolean feature value and at least three occurrences in training data, the performance decreases greatly both considering the correctness of predicted categories and the number of categories that are found.

When the keywords are represented as unigrams, a better performance is achieved than when they are kept intact. This is in line with the findings on *n*-grams by Caropreso et al. (2001). However, the results are still not satisfactory since both the precision and recall rates are lower than the baseline.

Titles, on the other hand, represented as unigrams and stemmed, are shown to be a useful information source when it comes to correctly predicting the text categories. Here, we achieve the highest precision rate of 94.2% although the recall rate and the F-measure are lower than the baseline.

Full-texts combined with keywords result in the highest recall value, 72.9%, as well as the highest F-measure, 81.7%, both above the baseline.

Our results clearly show that automatically extracted keywords can be a valuable supplement to full-text representations and that the combination of them yields the best performance, measured as both recall and micro-averaged F-measure. Our experiments also show that it is possible to do a satisfactory categorization having only keywords, given that we treat them as unigrams. Lastly, for higher precision in text classification, we can use the stemmed tokens in the headlines as features

Input feature	Feature value	Min. occurrence	Precision	Recall	F-measure
full-text unigram	tf*idf	3	93.03	71.69	80.98
keywords-only intact	bool	3	89.56	41.48	56.70
keywords-only unigram	bool	2	90.23	64.16	74.99
title	tf*idf	4	**94.23**	68.43	79.28
keywords+full	tf*idf	3	92.89	**72.94**	**81.72**

Table 4: Results on the fixed test set.

with tf*idf values.

As discussed in Section 2 and also presented in Table 2, the number of keywords assigned per document varies from zero to twelve. In Figure 1, we have plotted how the precision, the recall, and the F-measure for the test set vary with the number of assigned keywords for the keywords-only unigram representation.

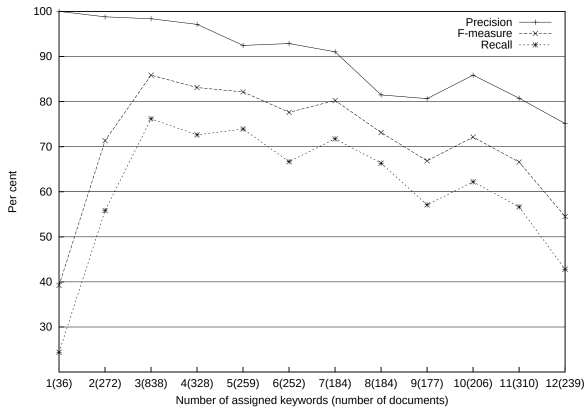


Figure 1: Precision, recall, and F-measure for each number of assigned keywords. The values in brackets denote the number of documents.

We can see that the F-measure and the recall reach their highest points when three keywords are extracted. The highest precision (100%) is obtained when the classification is performed on a single extracted keyword, but then there are only 36 documents present in this group, and the recall is low. Further experiments are needed in order to establish the optimal number of keywords to extract.

5 Related Work

For the work presented in this paper, there are two aspects that are of interest in previous work. These are in how the alternative input features (that is, alternative from unigrams) are selected and in how this alternative representation is used in combination with a bag-of-words representation (if it is).

An early work on linguistic phrases is done by Fürnkranz et al. (1998), where all noun phrases matching any of a number of syntactic heuristics are used as features. This approach leads to a higher precision at the low recall end, when evaluated on a corpus of Web pages. Aizawa (2001) extracts PoS-tagged compounds, matching predefined PoS patterns. The representation contains both the compounds and their constituents, and a small improvement is shown in the results on Reuters-21578. Moschitti and Basili (2004) add complex nominals as input features to their bag-of-words representation. The phrases are extracted by a system for terminology extraction[1]. The more complex representation leads to a small decrease on the Reuters corpus. In these studies, it is unclear how many phrases that are extracted and added to the representations.

Li et al. (2003) map documents (e-mail messages) that are to be classified into a vector space of keywords with associated probabilities. The mapping is based on a training phase requiring both texts and their corresponding summaries.

Another approach to combine different representations is taken by Sahlgren and Cöster (2004), where the full-text representation is combined with a concept-based representation by selecting one or the other for each category. They show that concept-based representations can outperform traditional word-based representations, and that a combination of the two different types of representations improves the performance of the classifier over all categories.

Keywords assigned to a particular text can be seen as a dense summary of the same. Some reports on how automatic summarization can be used to improve text categorization exist. For ex-

[1] In terminology extraction all terms describing a domain are to be extracted. The aim of automatic keyword indexing, on the other hand, is to find a small set of terms that describes a specific document, independently of the domain it belongs to. Thus, the set of terms must be limited to contain only the most salient ones.

ample, Ko et al. (2004) use methods from text summarization to find the sentences containing the important words. The words in these sentences are then given a higher weight in the feature vectors, by modifying the term frequency value with the sentence's score. The F-measure increases from 85.8 to 86.3 on the *Newsgroups* data set using Support vector machines.

Mihalcea and Hassan (2004) use an unsupervised method[2] to extract summaries, which in turn are used to categorize the documents. In their experiments on a sub-set of Reuters-21578 (among others), Mihalcea and Hassan show that the precision is increased when using the summaries rather than the full length documents. Özgür et al. (2005) have shown that limiting the representation to 2 000 features leads to a better performance, as evaluated on Reuters-21578. There is thus evidence that using only a sub-set of a document can give a more accurate classification. The question, though, is which sub-set to use.

In summary, the work presented in this paper has the most resemblance with the work by Ko et al. (2004), who also use a more dense version of a document to alter the feature values of a bag-of-words representation of a full-length document.

6 Concluding Remarks

In the experiments described in this paper, we investigated if automatically extracted keywords can improve automatic text categorization. More specifically, we investigated what impact keywords have on the task of text categorization by making predictions on the basis of keywords only, represented either as unigrams or intact, and by combining the full-text representation with automatically extracted keywords. The combination was obtained by giving higher weights to words in the full-texts that were also extracted as keywords. Throughout the study, we were concerned with the data representation and feature selection procedure. We investigated what feature value should be used (boolean, tf, or tf*idf) and the minimum number of occurrence of the tokens in the training data.

We showed that keywords can improve the performance of the text categorization. When keywords were used as a complement to the full-text representation an F-measure of 81.7% was obtained, higher than without the keywords (81.0%). Our results also clearly indicate that keywords alone can be used for the text categorization task when treated as unigrams, obtaining an F-measure of 75.0%. Lastly, for higher precision (94.2%) in text classification, we can use the stemmed tokens in the headlines.

The results presented in this study are lower than the state-of-the-art, even for the full-text run with unigrams, as we did not tune any other parameters than the feature values (boolean, term frequency, or tf*idf) and the threshold for the minimum number of occurrence in the training data.

There are, of course, possibilities for further improvements. One possibility could be to combine the tokens in the headlines and keywords in the same way as the full-text representation was combined with the keywords. Another possible improvement concerns the automatic keyword extraction process. The keywords are presented in order of their estimated "keywordness", based on the added regression value given by the three prediction models. This means that one alternative experiment would be to give different weights depending on which rank the keyword has achieved from the keyword extraction system. Another alternative would be to use the actual regression value.

We would like to emphasize that the automatically extracted keywords used in our experiments are not statistical phrases, such as bigrams or trigrams, but meaningful phrases selected by including linguistic analysis in the extraction procedure.

One insight that we can get from these experiments is that the automatically extracted keywords, which themselves have an F-measure of 44.0, can yield an F-measure of 75.0 in the categorization task. One reason for this is that the keywords have been evaluated using manually assigned keywords as the gold standard, meaning that paraphrasing and synonyms are severely punished. Kotcz et al. (2001) propose to use text categorization as a way to more objectively judge automatic text summarization techniques, by comparing how well an automatic summary fares on the task compared to other automatic summaries (that is, as an *extrinsic* evaluation method). The same would be valuable for automatic keyword indexing. Also, such an approach would facilitate comparisons between different systems, as common test-beds are lacking.

[2] This method has also been used to extract keywords (Mihalcea and Tarau, 2004).

In this study, we showed that automatic text categorization can benefit from automatically extracted keywords, although the bag-of-words representation is competitive with the best performance. Automatic keyword extraction as well as automatic text categorization are research areas where further improvements are needed in order to be useful for more efficient information retrieval.

Acknowledgments

The authors are grateful to the anonymous reviewers for their valuable suggestions on how to improve the paper.

References

Akiko Aizawa. 2001. Linguistic techniques to improve the performance of automatic text categorization. In *Proceedings of NLPRS-01, 6th Natural Language Processing Pacific Rim Symposium*, pages 307–314.

Maria Fernanda Caropreso, Stan Matwin, and Fabrizio Sebastiani. 2001. A learner-independent evaluation of the usefulness of statistical phrases for automated text categorization. In *Text Databases and Document Management: Theory and Practice*, pages 78–102.

Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. 1998. Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management (CIKM'98)*, pages 148–155.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, March.

Johannes Fürnkranz, Tom Mitchell, and Ellen Riloff. 1998. A case study using linguistic phrases for text categorization on the WWW. In *AAAI-98 Workshop on Learning for Text Categorization*.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2003)*, pages 216–223.

Anette Hulth. 2004. *Combining Machine Learning and Natural Language Processing for Automatic Keyword Extraction*. Ph.D. thesis, Department of Computer and Systems Sciences, Stockholm University.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*. MIT-Press.

Youngjoong Ko, Jinwoo Park, and Jungyun Seo. 2004. Improving text categorization using the importance of sentences. *Information Processing and Management*, 40(1):65–79.

Aleksander Kolcz, Vidya Prabakarmurthi, and Jugal Kalita. 2001. Summarization as feature selection for text categorization. In *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM'01)*, pages 365–370.

David D. Lewis. 1997. Reuters-21578 text categorization test collection, Distribution 1.0. AT&T Labs Research.

Cong Li, Ji-Rong Wen, and Hang Li. 2003. Text classification using stochastic keyword generation. In *Proceedings of the 20th International Conference on Machine Learning (ICML-2003)*.

Rada Mihalcea and Samer Hassan. 2005. Using the essence of texts to improve document classification. In *Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP 2005)*.

Rada Mihalcea and Paul Tarau. 2004. TextRank: bringing order into texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*.

Alessandro Moschitti and Roberto Basili. 2004. Complex linguistic features for text classification: A comprehensive study. In Sharon McDonald and John Tait, editors, *Proceedings of ECIR-04, 26th European Conference on Information Retrieval Research*, pages 181–196. Springer-Verlag.

Arzucan Özgür, Levent Özgür, and Tunga Güngör. 2005. Text categorization with class-based and corpus-based keyword selection. In *Proceedings of the 20th International Symposium on Computer and Information Sciences*, volume 3733 of *Lecture Notes in Computer Science*, pages 607–616. Springer-Verlag.

Martin Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Magnus Sahlgren and Rickard Cöster. 2004. Using bag-of-concepts to improve the performance of support vector machines in text categorization. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 487–493.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–49.

A Comparison and Semi-Quantitative Analysis of Words and Character-Bigrams as Features in Chinese Text Categorization

Jingyang Li **Maosong Sun** **Xian Zhang**
National Lab. of Intelligent Technology & Systems, Department of Computer Sci. & Tech.
Tsinghua University, Beijing 100084, China
lijingyang@gmail.com sms@tsinghua.edu.cn kevinn9@gmail.com

Abstract

Words and character-bigrams are both used as features in Chinese text processing tasks, but no systematic comparison or analysis of their values as features for Chinese text categorization has been reported heretofore. We carry out here a full performance comparison between them by experiments on various document collections (including a manually word-segmented corpus as a golden standard), and a semi-quantitative analysis to elucidate the characteristics of their behavior; and try to provide some preliminary clue for feature term choice (in most cases, character-bigrams are better than words) and dimensionality setting in text categorization systems.

1 Introduction[1]

Because of the popularity of the Vector Space Model (VSM) in text information processing, document indexing (term extraction) acts as a pre-requisite step in most text information processing tasks such as Information Retrieval (Baeza-Yates and Ribeiro-Neto, 1999) and Text Categorization (Sebastiani, 2002). It is empirically known that the indexing scheme is a non-trivial complication to system performance, especially for some Asian languages in which there are no explicit word margins and even no natural semantic unit. Concretely, in Chinese Text Categorization tasks, the two most important indexing units (feature terms) are word and character-bigram, so the problem is: which kind of terms[2] should be chosen as the feature terms, words or character-bigrams?

To obtain an all-sided idea about feature choice beforehand, we review here the possible feature variants (or, options). First, at the word level, we can do stemming, do stop-word pruning, include POS (Part of Speech) information, etc. Second, term combinations (such as “word-bigram”, “word + word-bigram”, “character-bigram + character-trigram”[3], etc.) can also be used as features (Nie et al., 2000). But, for Chinese Text Categorization, the “word or bigram” question is fundamental. They have quite different characteristics (e.g. bigrams overlap each other in text, but words do not) and influence the classification performance in different ways.

In Information Retrieval, it is reported that bigram indexing schemes outperforms word schemes to some or little extent (Luk and Kwok, 1997; Leong and Zhou 1998; Nie et al., 2000). Few similar comparative studies have been reported for Text Categorization (Li et al., 2003) so far in literature.

Text categorization and Information Retrieval are tasks that sometimes share identical aspects (Sebastiani, 2002) apart from term extraction (document indexing), such as *tfidf* term weighting and performance evaluation. Nevertheless, they are different tasks. One of the generally accepted connections between Information Retrieval and Text Categorization is that an information retrieval task could be partially taken as a binary classification problem with the query as the only positive training document. From this

[1] This research is supported by the National Natural Science Foundation of China under grant number 60573187 and 60321002, and the Tsinghua-ALVIS Project co-sponsored by the National Natural Science Foundation of China under grant number 60520130299 and EU FP6.

[2] The terminology “term” stands for both word and character-bigram. Term or combination of terms (in word-bigram or other forms) might be chosen as “feature”.

[3] The terminology “character” stands for Chinese character, and “bigram” stands for character-bigram in this paper.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 545–552,
Sydney, July 2006. ©2006 Association for Computational Linguistics

viewpoint, an IR task and a general TC task have a large difference in granularity. To better illustrate this difference, an example is present here. The words "制片人(film producer)" and "译制片(dubbed film)" should be taken as different terms in an IR task because a document with one would not necessarily be a good match for a query with the other, so the bigram "制片(film production)" is semantically not a shared part of these two words, i.e. not an appropriate feature term. But in a Text Categorization task, both words might have a similar meaning at the category level ("film" category, generally), which enables us to regard the bigram "制片" as a semantically acceptable representative word snippet for them, or for the category.

There are also differences in some other aspects of IR and TC. So it is significant to make a detailed comparison and analysis here on the relative value of words and bigrams as features in Text Categorization. The organization of this paper is as follows: Section 2 shows some experiments on different document collections to observe the common trends in the performance curves of the word-scheme and bigram-scheme; Section 3 qualitatively analyses these trends; Section 4 makes some statistical analysis to corroborate the issues addressed in Section 3; Section 5 summarizes the results and concludes.

2 Performance Comparison

Three document collections in Chinese language are used in this study.

The electronic version of *Chinese Encyclopedia* ("CE"): It has 55 subject categories and 71674 single-labeled documents (entries). It is randomly split by a proportion of 9:1 into a training set with 64533 documents and a test set with 7141 documents. Every document has the full-text. This data collection does not have much of a sparseness problem.

The training data from a national Chinese text categorization evaluation[4] ("CTC"): It has 36 subject categories and 3600 single-labeled[5] documents. It is randomly split by a proportion of 4:1 into a training set with 2800 documents and a test set with 720 documents. Documents in this data collection are from various sources including news websites, and some documents may be very short. This data collection has a moderate sparseness problem.

A manually word-segmented corpus from the State Language Affairs Commission ("LC"): It has more than 100 categories and more than 20000 single-labeled documents[6]. In this study, we choose a subset of 12 categories with the most documents (totally 2022 documents). It is randomly split by a proportion of 2:1 into a training set and a test set. Every document has the full-text and has been entirely word-segmented[7] by hand (which could be regarded as a golden standard of segmentation).

All experiments in this study are carried out at various feature space dimensionalities to show the scalability. Classifiers used in this study are Rocchio and SVM. All experiments here are multi-class tasks and each document is assigned a single category label.

The outline of this section is as follows: Subsection 2.1 shows experiments based on the Rocchio classifier, feature selection schemes besides *Chi* and term weighting schemes besides *tfidf* to compare the automatic segmented word features with bigram features on CE and CTC, and both document collections lead to similar behaviors; Subsection 2.2 shows experiments on CE by a SVM classifier, in which, unlike with the Rocchio method, *Chi* feature selection scheme and *tfidf* term weighting scheme outperform other schemes; Subsection 2.3 shows experiments by a SVM classifier with *Chi* feature selection and *tfidf* term weighting on LC (manual word segmentation) to compare the best word features with bigram features.

2.1 The Rocchio Method and Various Settings

The Rocchio method is rooted in the IR tradition, and is very different from machine learning ones (such as SVM) (Joachims, 1997; Sebastiani, 2002). Therefore, we choose it here as one of the representative classifiers to be examined. In the experiment, the control parameter of negative examples is set to 0, so this Rocchio based classifier is in fact a centroid-based classifier.

$Chi_{\max}$ is a state-of-the-art feature selection criterion for dimensionality reduction (Yang and Peterson, 1997; Rogati and Yang, 2002). $Chi_{\max*}CIG$ (Xue and Sun, 2003a) is reported to be better in Chinese text categorization by a cen-

[4] The Annual Evaluation of Chinese Text Categorization 2004, by 863 National Natural Science Foundation.

[5] In the original document collection, a document might have a secondary category label. In this study, only the primary category label is reserved.

[6] Not completed.

[7] And POS (part-of-speech) tagged as well. But POS tags are not used in this study.

troid based classifier, so we choose it as another representative feature selection criterion besides $Chi_{\max}$.

Likewise, as for term weighting schemes, in addition to *tfidf*, the state of the art (Baeza-Yates and Ribeiro-Neto, 1999), we also choose *tfidf*CIG* (Xue and Sun, 2003b).

Two word segmentation schemes are used for the word-indexing of documents. One is the m*aximum match* algorithm ("mmword" in the figures), which is a representative of simple and fast word segmentation algorithms. The other is ICTCLAS[8] ("lqword" in the figures). ICTCLAS is one of the best word segmentation systems (SIGHAN 2003) and reaches a segmentation precision of more than 97%, so we choose it as a representative of state-of-the-art schemes for automatic word-indexing of document).

For evaluation of single-label classifications, F_1-measure, *precision*, *recall* and *accuracy* (Baeza-Yates and Ribeiro-Neto, 1999; Sebastiani, 2002) have the same value by microaveraging[9], and are labeled with "performance" in the following figures.

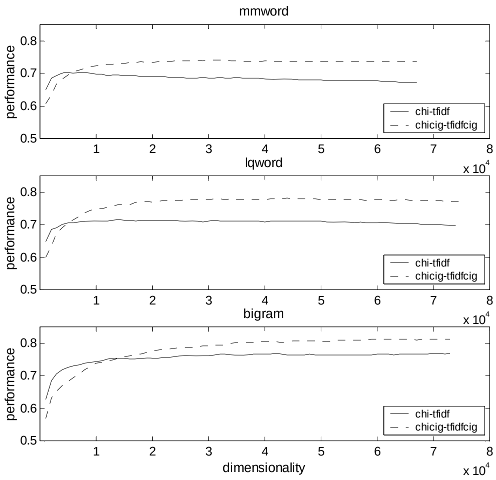


Figure 1. *chi-tfidf* and *chicig-tfidfcig* on CE

Figure 1 shows the performance-dimensionality curves of the *chi-tfidf* approach and the approach with *CIG*, by *mmword*, *lqword* and *bigram* document indexing, on the CE document collection. We can see that the original *chi-tfidf* approach is better at low dimensionalities (less than 10000 dimensions), while the *CIG* version is better at high dimensionalities and reaches a higher limit.[10]

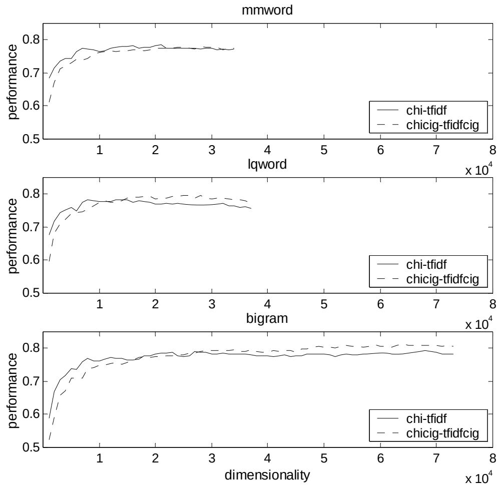


Figure 2. *chi-tfidf* and *chicig-tfidfcig* on CTC

Figure 2 shows the same group of curves for the CTC document collection. The curves fluctuate more than the curves for the CE collection because of sparseness; The CE collection is more sensitive to the additions of terms that come with the increase of dimensionality. The CE curves in the following figures show similar fluctuations for the same reason.

For a parallel comparison among *mmword*, *lqword* and *bigram* schemes, the curves in Figure 1 and Figure 2 are regrouped and shown in Figure 3 and Figure 4.

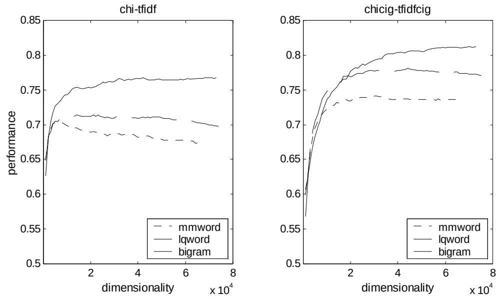


Figure 3. *mmword*, *lqword* and *bigram* on CE

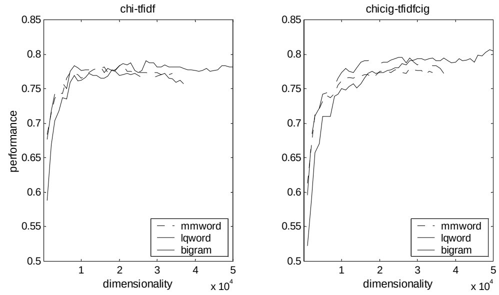


Figure 4. *mmword*, *lqword* and *bigram* on CTC

[8] http://www.nlp.org.cn/project/project.php?proj_id=6

[9] Microaveraging is more prefered in most cases than macroaveraging (Sebastiani 2002).

[10] In all figures in this paper, curves might be truncated due to the large scale of dimensionality, especially the curves of bigram scheme. For these kinds of figures, at least one of the following is satisfied: (a) every curve has shown its zenith; (b) only one curve is not complete and has shown a higher zenith than other curves; (c) a margin line is shown to indicate the limit of the incomplete curve.

We can see that the *lqword* scheme outperforms the *mmword* scheme at almost any dimensionality, which means the more precise the word segmentation the better the classification performance. At the same time, the *bigram* scheme outperforms both of the word schemes on a high dimensionality, wherea the word schemes might outperform the *bigram* scheme on a low dimensionality.

Till now, the experiments on CE and CTC show the same characteristics despite the performance fluctuation on CTC caused by sparseness. Hence in the next subsections CE is used instead of both of them because its curves are smoother.

2.2 SVM on Words and Bigrams

As stated in the previous subsection, the *lqword* scheme always outperforms the *mmword* scheme; we compare here only the *lqword* scheme with the *bigram* scheme.

Support Vector Machine (SVM) is one of the best classifiers at present (Vapnik, 1995; Joachims, 1998), so we choose it as the main classifier in this study. The SVM implementation used here is LIBSVM (Chang, 2001); the type of SVM is set to “C-SVC” and the kernel type is set to linear, which means a one-with-one scheme is used in the multi-class classification.

Because the *CIG*’s effectiveness on a SVM classifier is not examined in Xue and Sun (2003a, 2003b)’s report, we make here the four combinations of schemes with and without *CIG* in feature selection and term weighting. The experiment results are shown in Figure 5. The collection used is CE.

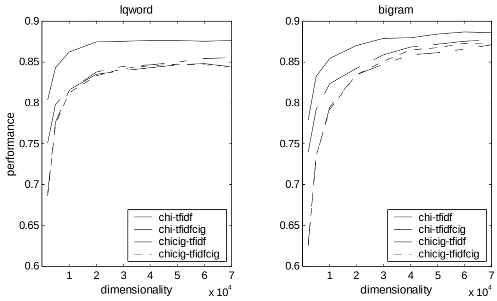


Figure 5. *chi-tfidf* and *cig*-involved approaches on *lqword* and *bigram*

Here we find that the *chi-tfidf* combination outperforms any approach with *CIG*, which is the opposite of the results with the Rocchio method. And the results with SVM are all better than the results with the Rocchio method. So we find that the feature selection scheme and the term weighting scheme are related to the classifier, which is worth noting. In other words, no feature selection scheme or term weighting scheme is absolutely the best for all classifiers. Therefore, a reasonable choice is to select the best performing combination of feature selection scheme, term weighting scheme and classifier, i.e. *chi-tfidf* and SVM. The curves for the *lqword* scheme and the *bigram* scheme are redrawn in Figure 6 to make them clearer.

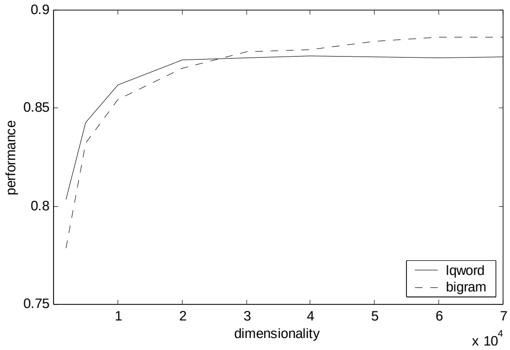


Figure 6. *lqword* and *bigram* on CE

The curves shown in Figure 6 are similar to those in Figure 3. The differences are: (a) a larger dimensionality is needed for the *bigram* scheme to start outperforming the *lqword* scheme; (b) the two schemes have a smaller performance gap.

The *lqword* scheme reaches its top performance at a dimensionality of around 40000, and the *bigram* scheme reaches its top performance at a dimensionality of around 60000 to 70000, after which both schemes’ performances slowly decrease. The reason is that the low ranked terms in feature selection are in fact noise and do not help to classification, which is why the feature selection phase is necessary.

2.3 Comparing Manually Segmented Words and Bigrams

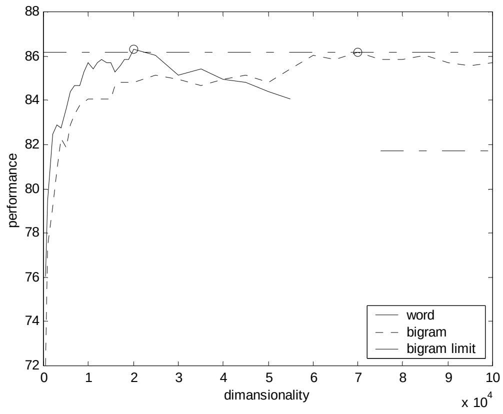


Figure 7. *word* and *bigram* on LC

Up to now, bigram features seem to be better than word ones for fairly large dimensionalities. But it appears that word segmentation precision impacts classification performance. So we choose here a fully manually segmented document collection to detect the best performance a word scheme could reach and compare it with the bigram scheme.

Figure 7 shows such an experiment result on the LC document collection (the circles indicate the maximums and the dash-dot lines indicate the superior limit and the asymptotic interior limit of the bigram scheme). The word scheme reaches a top performance around the dimensionality of 20000, which is a little higher than the bigram scheme's zenith around 70000.

Besides this experiment on 12 categories of the LC document collection, some experiments on fewer (2 to 6) categories of this subset were also done, and showed similar behaviors. The word scheme shows a better performance than the bigram scheme and needs a much lower dimensionality. The simpler the classification task is, the more distinct this behavior is.

3 Qualitative Analysis

To analyze the performance of words and bigrams as feature terms in Chinese text categorization, we need to investigate two aspects as follows.

3.1 An Individual Feature Perspective

The word is a natural semantic unit in Chinese language and expresses a complete meaning in text. The bigram is not a natural semantic unit and might not express a complete meaning in text, but there are also reasons for the bigram to be a good feature term.

First, two-character words and three-character words account for most of all multi-character Chinese words (Liu and Liang, 1986). A two-character word can be substituted by the same bigram. At the granularity of most categorization tasks, a three-character words can often be substituted by one of its sub-bigrams (namely the "intraword bigram" in the next section) without a change of meaning. For instance, "标赛" is a sub-bigram of the word "锦标赛(tournament)" and could represent it without ambiguity.

Second, a bigram may overlap on two successive words (namely the "interword bigram" in the next section), and thus to some extent fills the role of a word-bigram. The word-bigram as a more definite (although more sparse) feature surely helps the classification. For instance, "气预" is a bigram overlapping on the two successive words "天气(weather)" and "预报(forecast)", and could almost replace the word-bigram (also a phrase) "天气预报(weather forecast)", which is more likely to be a representative feature of the category "气象学(meteorology)" than either word.

Third, due to the first issue, bigram features have some capability of identifying OOV (out-of-vocabulary) words[11], and help improve the *recall* of classification.

The above issues state the advantages of bigrams compared with words. But in the first and second issue, the equivalence between bigram and word or word-bigram is not perfect. For instance, the word "文学(literature)" is a also sub-bigram of the word "天文学(astronomy)", but their meanings are completely different. So the loss and distortion of semantic information is a disadvantage of bigram features over word features.

Furthermore, one-character words cover about 7% of words and more than 30% of word occurrences in the Chinese language; they are effevtive in the word scheme and are not involved in the above issues. Note that the impact of effective one-character words on the classification is not as large as their total frequency, because the high frequency ones are often too common to have a good classification power, for instance, the word "的 (of, 's)".

3.2 A Mass Feature Perspective

Features are not independently acting in text classification. They are assembled together to constitute a feature space. Except for a few models such as Latent Semantic Indexing (LSI) (Deerwester et al., 1990), most models assume the feature space to be orthogonal. This assumption might not affect the effectiveness of the models, but the semantic redundancy and complementation among the feature terms do impact on the classification efficiency at a given dimensionality.

According to the first issue addressed in the previous subsection, a bigram might cover for more than one word. For instance, the bigram "织物" is a sub-bigram of the words "织物(fabric)", "棉织物(cotton fabric)", "针织物(knitted fabric)", and also a good substitute of

[11] The "OOV words" in this paper stand for the words that occur in the test documents but not in the training document.

them. So, to a certain extent, word features are redundant with regard to the bigram features associated to them. Similarly, according to the second issue addressed, a bigram might cover for more than one word-bigram. For instance, the bigram "篇小" is a sub-bigram of the word-bigrams (phrases) "短篇小说(short story)", "中篇小说(novelette)", "长篇小说(novel)" and also a good substitute for them. So, as an addition to the second issue stated in the previous subsection, a bigram feature might even cover for more than one word-bigram.

On the other hand, bigrams features are also redundant with regard to word features associated with them. For instance, the "锦标" and "标赛" are both sub-bigrams of the previously mentioned word "锦标赛". In some cases, more than one sub-bigram can be a good representative of a word.

We make a word list and a bigram list sorted by the feature selection criterion in a descending order. We now try to find how the relative redundancy degrees of the word list and the bigram list vary with the dimensionality. Following issues are elicited by an observation on the two lists (not shown here due to space limitations).

The relative redundancy rate in the word list keeps even while the dimensionality varies to a certain extent, because words that share a common sub-bigram might not have similar statistics and thus be scattered in the word feature list. Note that these words are possibly ranked lower in the list than the sub-bigram because feature selection criteria (such as *Chi*) often prefer higher frequency terms to lower frequency ones, and every word containing the bigram certainly has a lower frequency than the bigram itself.

The relative redundancy in the bigram list might be not as even as in the word list. Good (representative) sub-bigrams of a word are quite likely to be ranked close to the word itself. For instance, "作曲" and "曲家" are sub-bigrams of the word "作曲家(music composer)", both the bigrams and the word are on the top of the lists. Theretofore, the bigram list has a relatively large redundancy rate at low dimensionalities. The redundancy rate should decrease along with the increas of dimensionality for: (a) the relative redundancy in the word list counteracts the redundancy in the bigram list, because the words that contain a same bigram are gradually included as the dimensionality increases; (b) the proportion of interword bigrams increases in the bigram list and there is generally no redundancy between interword bigrams and intraword bigrams.

Last, there are more bigram features than word features because bigrams can overlap each other in the text but words can not. Thus the bigrams as a whole should theoretically contain more information than the words as a whole.

From the above analysis and observations, bigram features are expected to outperform word features at high dimensionalities. And word features are expected to outperform bigram features at low dimensionalities.

4 Semi-Quantitative Analysis

In this section, a preliminary statistical analysis is presented to corroborate the statements in the above qualitative analysis and expected to be identical with the experiment results shown in Section 1. All statistics in this section are based on the CE document collection and the *lqword* segmentation scheme (because the CE document collection is large enough to provide good statistical characteristics).

4.1 Intraword Bigrams and Interword Bigrams

In the previous section, only the intraword bigrams were discussed together with the words. But every bigram may have both intraword occurrences and interword occurrences. Therefore we need to distinguish these two kinds of bigrams at a statistical level. For every bigram, the number of intraword occurrences and the number of interword occurrences are counted and we can use

$$\log\left(\frac{interword\# + 1}{intraword\# + 1}\right)$$

as a metric to indicate its natual propensity to be a intraword bigram. The probability density of bigrams about on this metric is shown in Figure 8.

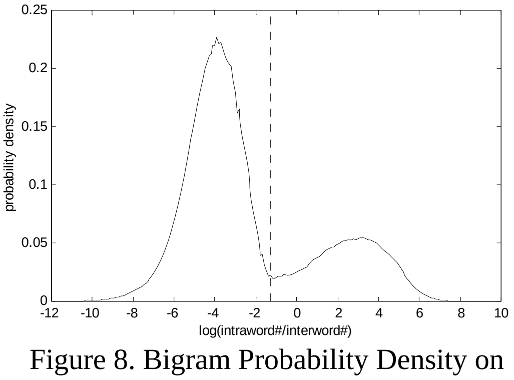


Figure 8. Bigram Probability Density on log(*intraword#*/*interword#*)

The figure shows a mixture of two Gaussian distributions, the left one for "natural interword bigrams" and the right one for "natural intraword bigrams". We can moderately distinguish these two kinds of bigrams by a division at -1.4.

4.2 Overall Information Quantity of a Feature Space

The performance limit of a classification is related to the quantity of information used. So a quantitative metric of the information a feature space can provide is need. *Feature Quantity* (Aizawa, 2000) is suitable for this purpose because it comes from information theory and is additive; *tfidf* was also reported as an appropriate metric of feature quantity (defined as "*probability · information*"). Because of the probability involved as a factor, the overall information provided by a feature space can be calculated on training data by summation.

The redundancy and complementation mentioned in Subsection 3.2 must be taken into account in the calculation of overall information quantity. For bigrams, the redundancy with regard to words associated with them between two intraword bigrams is given by

$$\sum_{b_{1,2} \subset w} tf(w) \cdot \min\{idf(b_1), idf(b_2)\}$$

in which b_1 and b_2 stand for the two bigrams and w stands for any word containing both of them. The overall information quantity is obtained by subtracting the redundancy between each pair of bigrams from the sum of all features' *feature quantity* (*tfidf*). Redundancy among more than two bigrams is ignored. For words, there is only complementation among words but not redundancy, the complementation with regard to bigrams associated with them is given by

$$\begin{cases} tf(w) \cdot \min\limits_{b \subset w}\{idf(b)\}, & \text{if } b \text{ exists;} \\ tf(w) \cdot idf(w), & \text{if } b \text{ does not exists.} \end{cases}$$

in which b is an intraword bigram contained by w. The overall information is calculated by summing the complementations of all words.

4.3 Statistics and Discussion

Figure 9 shows the variation of these overall information metrics on the CE document collection. It corroborates the characteristics analyzed in Section 3 and corresponds with the performance curves in Section 2.

Figure 10 shows the proportion of interword bigrams at different dimensionalities, which also corresponds with the analysis in Section 3.

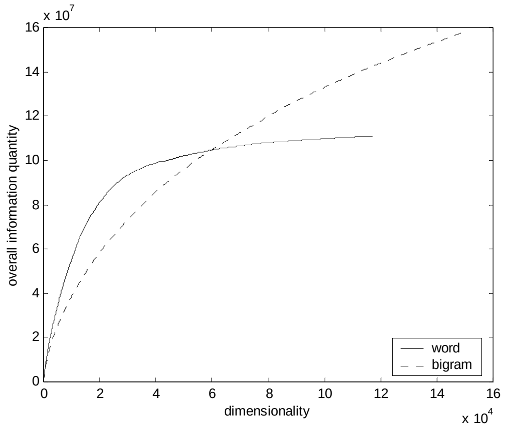


Figure 9. Overall Information Quantity on CE

The curves do not cross at exactly the same dimensionality as in the figures in Section 1, because other complications impact on the classification performance: (a) OOV word identifying capability, as stated in Subsection 3.1; (b) word segmentation precision; (c) granularity of the categories (words have more definite semantic meaning than bigrams and lead to a better performance for small category granularities); (d) noise terms, introduced in the feature space during the increase of dimensionality. With these factors, the actual curves would not keep increasing as they do in Figure 9.

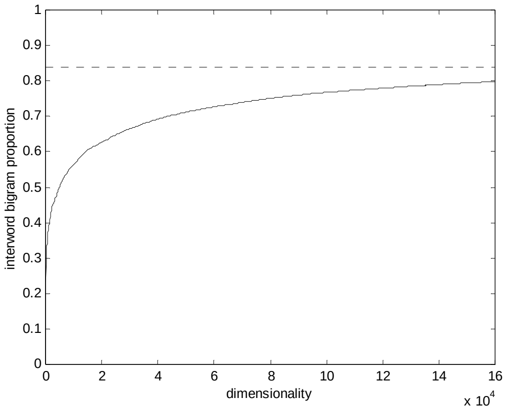


Figure 10. Interword Bigram Proportion on CE

5 Conclusion

In this paper, we aimed to thoroughly compare the value of words and bigrams as feature terms in text categorization, and make the implicit mechanism explicit.

Experimental comparison showed that the *Chi* feature selection scheme and the *tfidf* term weighting scheme are still the best choices for (Chinese) text categorization on a SVM classifier. In most cases, the bigram scheme outperforms the word scheme at high dimensionalities and usually reaches its top performance at a dimensionality of around 70000. The word scheme often outperforms the bigram scheme at low dimensionalities and reaches its top performance at a dimensionality of less than 40000.

Whether the best performance of the word scheme is higher than the best performance scheme depends considerably on the word segmentation precision and the number of categories. The word scheme performs better with a higher word segmentation precision and fewer (<10) categories.

A word scheme costs more document indexing time than a bigram scheme does; however a bigram scheme costs more training time and classification time than a word scheme does at the same performance level due to its higher dimensionality. Considering that the document indexing is needed in both the training phase and the classification phase, a high precision word scheme is more time consuming as a whole than a bigram scheme.

As a concluding suggestion: a word scheme is more fit for small-scale tasks (with no more than 10 categories and no strict classification speed requirements) and needs a high precision word segmentation system; a bigram scheme is more fit for large-scale tasks (with dozens of categories or even more) without too strict training speed requirements (because a high dimensionality and a large number of categories lead to a long training time).

Reference

Akiko Aizawa. 2000. *The Feature Quantity: An Information Theoretic Perspective of Tfidf-like Measures, Proceedings of ACM SIGIR 2000,* 104-111.

Ricardo Baeza-Yates, Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval,* Addison-Wesley

Chih-Chung Chang, Chih-Jen Lin. 2001. *LIBSVM: A Library for Support Vector Machines*, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Steve Deerwester, Sue T. Dumais, George W. Furnas, Richard Harshman. 1990. *Indexing by Latent Semantic Analysis, Journal of the American Society for Information Science,* 41:391-407.

Thorsten Joachims. 1997. *A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization, Proceedings of 14th International Conference on Machine Learning* (Nashville, TN, 1997), 143-151.

Thorsten Joachims. 1998. *Text Categorization with Support Vector Machine: Learning with Many Relevant Features, Proceedings of the 10th European Conference on Machine Learning,* 137-142.

Mun-Kew Leong, Hong Zhou. 1998. *Preliminary Qualitative Analysis of Segmented vs. Bigram Indexing in Chinese, The 6th Text Retrieval Conference (TREC-6), NIST Special Publication 500-240,* 551-557.

Baoli Li, Yuzhong Chen, Xiaojing Bai, Shiwen Yu. 2003. *Experimental Study on Representing Units in Chinese Text Categorization, Proceedings of the 4th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2003),* 602-614.

Yuan Liu, Nanyuan Liang. 1986. *Basic Engineering for Chinese Processing – Contemporary Chinese Words Frequency Count, Journal of Chinese Information Processing,* 1(1):17-25.

Robert W.P. Luk, K.L. Kwok. 1997. *Comparing representations in Chinese information retrieval. Proceedings of ACM SIGIR 1997*, 34-41.

Jianyun Nie, Fuji Ren. 1999. *Chinese Information Retrieval: Using Characters or Words? Information Processing and Management*, 35:443-462.

Jianyun Nie, Jianfeng Gao, Jian Zhang, Ming Zhou. 2000. *On the Use of Words and N-grams for Chinese Information Retrieval, Proceedings of 5th International Workshop on Information Retrieval with Asian Languages*

Monica Rogati, Yiming Yang. 2002. *High-performing Feature Selection for Text Classification, Proceedings of ACM Conference on Information and Knowledge Management 2002,* 659-661.

Gerard Salton, Christopher Buckley. 1988. *Term Weighting Approaches in Automatic Text Retrieval, Information Processing and Management,* 24(5):513-523.

Fabrizio Sebastiani. 2002. *Machine Learning in Automated Text Categorization, ACM Computing Surveys*, 34(1):1-47

Dejun Xue, Maosong Sun. 2003a. *Select Strong Information Features to Improve Text Categorization Effectiveness, Journal of Intelligent Systems,* Special Issue.

Dejun Xue, Maosong Sun. 2003b. *A Study on Feature Weighting in Chinese Text Categorization, Proceedings of the 4th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2003),* 594-604.

Vladimir Vapnik. 1995. *The Nature of Statistical Learning Theory,* Springer.

Yiming Yang, Jan O. Pederson. 1997. *A Comparative Study on Feature Selection in Text Categorization, Proceedings of ICML 1997*, 412-420.

Exploiting Comparable Corpora and Bilingual Dictionaries for Cross-Language Text Categorization

Alfio Gliozzo and **Carlo Strapparava**
ITC-Irst
via Sommarive, I-38050, Trento, ITALY
{gliozzo,strappa}@itc.it

Abstract

Cross-language Text Categorization is the task of assigning semantic classes to documents written in a target language (e.g. English) while the system is trained using labeled documents in a source language (e.g. Italian).

In this work we present many solutions according to the availability of bilingual resources, and we show that it is possible to deal with the problem even when no such resources are accessible. The core technique relies on the automatic acquisition of Multilingual Domain Models from comparable corpora.

Experiments show the effectiveness of our approach, providing a low cost solution for the Cross Language Text Categorization task. In particular, when bilingual dictionaries are available the performance of the categorization gets close to that of monolingual text categorization.

1 Introduction

In the worldwide scenario of the Web age, multilinguality is a crucial issue to deal with and to investigate, leading us to reformulate most of the classical Natural Language Processing (NLP) problems into a multilingual setting. For instance the classical monolingual Text Categorization (TC) problem can be reformulated as a *Cross Language Text Categorization* (CLTC) task, in which the system is trained using labeled examples in a source language (e.g. *English*), and it classifies documents in a different target language (e.g. *Italian*).

The applicative interest for the CLTC is immediately clear in the globalized Web scenario. For example, in the community based trade (e.g. eBay) it is often necessary to archive texts in different languages by adopting common merceological categories, very often defined by collections of documents in a source language (e.g. English). Another application along this direction is Cross Lingual Question Answering, in which it would be very useful to filter out the candidate answers according to their topics.

In the literature, this task has been proposed quite recently (Bel et al., 2003; Gliozzo and Strapparava, 2005). In those works, authors exploited comparable corpora showing promising results. A more recent work (Rigutini et al., 2005) proposed the use of Machine Translation techniques to approach the same task.

Classical approaches for multilingual problems have been conceived by following two main directions: (i) knowledge based approaches, mostly implemented by rule based systems and (ii) empirical approaches, in general relying on statistical learning from parallel corpora. Knowledge based approaches are often affected by low accuracy. Such limitation is mainly due to the problem of tuning large scale multilingual lexical resources (e.g. MultiWordNet, EuroWordNet) for the specific application task (e.g. discarding irrelevant senses, extending the lexicon with domain specific terms and their translations). On the other hand, empirical approaches are in general more accurate, because they can be trained from domain specific collections of parallel text to represent the application needs. There exist many interesting works about using parallel corpora for multilingual applications (Melamed, 2001), such as Machine Translation (Callison-Burch et al., 2004), Cross Lingual

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 553–560,
Sydney, July 2006. ©2006 Association for Computational Linguistics

Information Retrieval (Littman et al., 1998), and so on.

However it is not always easy to find or build parallel corpora. This is the main reason why the "weaker" notion of comparable corpora is a matter of recent interest in the field of Computational Linguistics (Gaussier et al., 2004). In fact, comparable corpora are easier to collect for most languages (e.g. collections of international news agencies), providing a low cost knowledge source for multilingual applications.

The main problem of adopting comparable corpora for multilingual knowledge acquisition is that only weaker statistical evidence can be captured. In fact, while parallel corpora provide stronger (text-based) statistical evidence to detect translation pairs by analyzing term co-occurrences in translated documents, comparable corpora provides weaker (term-based) evidence, because text alignments are not available.

In this paper we present some solutions to deal with CLTC according to the availability of bilingual resources, and we show that it is possible to deal with the problem even when no such resources are accessible. The core technique relies on the automatic acquisition of Multilingual Domain Models (MDMs) from comparable corpora. This allows us to define a kernel function (i.e. a similarity function among documents in different languages) that is then exploited inside a Support Vector Machines classification framework. We also investigate this problem exploiting synset-aligned multilingual WordNets and standard bilingual dictionaries (e.g. Collins).

Experiments show the effectiveness of our approach, providing a simple and low cost solution for the Cross-Language Text Categorization task. In particular, when bilingual dictionaries/repositories are available, the performance of the categorization gets close to that of monolingual TC.

The paper is structured as follows. Section 2 briefly discusses the notion of comparable corpora. Section 3 shows how to perform cross-lingual TC when no bilingual dictionaries are available and it is possible to rely on a comparability assumption. Section 4 present a more elaborated technique to acquire MDMs exploiting bilingual resources, such as MultiWordNet (i.e. a synset-aligned WordNet) and Collins bilingual dictionary. Section 5 evaluates our methodologies and Section 6 concludes the paper suggesting some future developments.

2 Comparable Corpora

Comparable corpora are collections of texts in different languages regarding similar topics (e.g. a collection of news published by agencies in the same period). More restrictive requirements are expected for parallel corpora (i.e. corpora composed of texts which are mutual translations), while the class of the multilingual corpora (i.e. collection of texts expressed in different languages without any additional requirement) is the more general. Obviously parallel corpora are also comparable, while comparable corpora are also multilingual.

In a more precise way, let $L = \{L^1, L^2, \ldots, L^l\}$ be a set of languages, let $T^i = \{t^i_1, t^i_2, \ldots, t^i_n\}$ be a collection of texts expressed in the language $L^i \in L$, and let $\psi(t^j_h, t^i_z)$ be a function that returns 1 if t^i_z is the translation of t^j_h and 0 otherwise. A *multilingual corpus* is the collection of texts defined by $T^* = \bigcup_i T^i$. If the function ψ exists for every text $t^i_z \in T^*$ and for every language L^j, and is known, then the corpus is *parallel* and *aligned* at document level.

For the purpose of this paper it is enough to assume that two corpora are comparable, i.e. they are composed of documents about the same topics and produced in the same period (e.g. possibly from different news agencies), and it is not known if a function ψ exists, even if in principle it could exist and return 1 for a strict subset of document pairs.

The texts inside comparable corpora, being about the same topics, should refer to the same concepts by using various expressions in different languages. On the other hand, most of the proper nouns, relevant entities and words that are not yet lexicalized in the language, are expressed by using their original terms. As a consequence the *same entities* will be denoted with the *same words* in different languages, allowing us to automatically detect couples of translation pairs just by looking at the word shape (Koehn and Knight, 2002). Our hypothesis is that comparable corpora contain a large amount of such words, just because texts, referring to the same topics in different languages, will often adopt the same terms to denote the same entities[1].

[1] According to our assumption, a possible additional cri-

However, the simple presence of these shared words is not enough to get significant results in CLTC tasks. As we will see, we need to exploit these common words to induce a second-order similarity for the other words in the lexicons.

2.1 The Multilingual Vector Space Model

Let $T = \{t_1, t_2, \ldots, t_n\}$ be a corpus, and $V = \{w_1, w_2, \ldots, w_k\}$ be its vocabulary. In the monolingual settings, the Vector Space Model (VSM) is a k-dimensional space $\mathbf{R}^k$, in which the text $t_j \in T$ is represented by means of the vector $\vec{t_j}$ such that the z^{th} component of $\vec{t_j}$ is the frequency of w_z in t_j. The similarity among two texts in the VSM is then estimated by computing the cosine of their vectors in the VSM.

Unfortunately, such a model cannot be adopted in the multilingual settings, because the VSMs of different languages are mainly disjoint, and the similarity between two texts in different languages would always turn out to be zero. This situation is represented in Figure 1, in which both the left-bottom and the rigth-upper regions of the matrix are totally filled by zeros.

On the other hand, the assumption of corpora comparability seen in Section 2, implies the presence of a number of common words, represented by the central rows of the matrix in Figure 1.

As we will show in Section 5, this model is rather poor because of its sparseness. In the next section, we will show how to use such words as seeds to induce a Multilingual Domain VSM, in which second order relations among terms and documents in different languages are considered to improve the similarity estimation.

3 Exploiting Comparable Corpora

Looking at the multilingual term-by-document matrix in Figure 1, a first attempt to merge the subspaces associated to each language is to exploit the information provided by external knowledge sources, such as bilingual dictionaries, e.g. collapsing all the rows representing translation pairs. In this setting, the similarity among texts in different languages could be estimated by exploiting the classical VSM just described. However, the main disadvantage of this approach to estimate inter-lingual text similarity is that it strongly relies on the availability of a multilingual lexical resource. For languages with scarce resources a bilingual dictionary could be not easily available. Secondly, an important requirement of such a resource is its coverage (i.e. the amount of possible translation pairs that are actually contained in it). Finally, another problem is that ambiguous terms could be translated in different ways, leading us to collapse together rows describing terms with very different meanings. In Section 4 we will see how the availability of bilingual dictionaries influences the techniques and the performance. In the present Section we want to explore the case in which such resources are supposed not available.

3.1 Multilingual Domain Model

A MDM is a multilingual extension of the concept of Domain Model. In the literature, Domain Models have been introduced to represent ambiguity and variability (Gliozzo et al., 2004) and successfully exploited in many NLP applications, such as Word Sense Disambiguation (Strapparava et al., 2004), Text Categorization and Term Categorization.

A Domain Model is composed of soft clusters of terms. Each cluster represents a semantic domain, i.e. a set of terms that often co-occur in texts having similar topics. Such clusters identify groups of words belonging to the same semantic field, and thus highly paradigmatically related. MDMs are Domain Models containing terms in more than one language.

A MDM is represented by a matrix $\mathbf{D}$, containing the degree of association among terms in all the languages and domains, as illustrated in Table 1. For example the term *virus* is associated to both the domain COMPUTER_SCIENCE and the domain MEDICINE while the domain MEDICINE is associated to both the terms *AIDS* and *HIV*. Inter-lingual

	MEDICINE	COMPUTER_SCIENCE
$HIV^{e/i}$	1	0
$AIDS^{e/i}$	1	0
$virus^{e/i}$	0.5	0.5
$hospital^{e}$	1	0
$laptop^{e}$	0	1
$Microsoft^{e/i}$	0	1
$clinica^{i}$	1	0

Table 1: Example of Domain Matrix. w^e denotes English terms, w^i Italian terms and $w^{e/i}$ the common terms to both languages.

terion to decide whether two corpora are comparable is to estimate the percentage of terms in the intersection of their vocabularies.

$$\left(\begin{array}{ll|ccccc|ccccc}
 & & & & \textit{English texts} & & & & & \textit{Italian texts} & & \\
 & & t_1^e & t_2^e & \cdots & t_{n-1}^e & t_n^e & t_1^i & t_2^i & \cdots & t_{m-1}^i & t_m^i \\
\hline
 & w_1^e & 0 & 1 & \cdots & 0 & 1 & 0 & 0 & \cdots & & \\
\textit{English Lexicon} & w_2^e & 1 & 1 & \cdots & 1 & 0 & 0 & \ddots & & & \\
 & \vdots & \multicolumn{5}{c|}{\dots\dots\dots\dots\dots\dots} & \vdots & & 0 & & \vdots \\
 & w_{p-1}^e & 0 & 1 & \cdots & 0 & 0 & & & & \ddots & 0 \\
 & w_p^e & 0 & 1 & \cdots & 0 & 0 & & & \cdots & 0 & 0 \\
\hline
\textit{common } w_i & w_1^{e/i} & 0 & \mathbf{1} & \cdots & 0 & 0 & 0 & 0 & \cdots & \mathbf{1} & 0 \\
 & \vdots & \multicolumn{5}{c|}{\dots\dots\dots\dots\dots\dots} & \multicolumn{5}{c}{\dots\dots\dots\dots\dots\dots} \\
\hline
 & w_1^i & 0 & 0 & \cdots & & & 0 & 1 & \cdots & 1 & 1 \\
\textit{Italian Lexicon} & w_2^i & 0 & \ddots & & & & 1 & 1 & \cdots & 0 & 1 \\
 & \vdots & \vdots & & 0 & & \vdots & \multicolumn{5}{c}{\dots\dots\dots\dots\dots\dots} \\
 & w_{q-1}^i & & & & \ddots & 0 & 0 & 1 & \cdots & 0 & 1 \\
 & w_q^i & & & \cdots & 0 & 0 & 0 & 1 & \cdots & 1 & 0
\end{array}\right)$$

Figure 1: Multilingual term-by-document matrix

domain relations are captured by placing different terms of different languages in the same semantic field (as for example $HIV^{e/i}$, $AIDS^{e/i}$, $hospital^e$, and $clinica^i$). Most of the named entities, such as *Microsoft* and *HIV* are expressed using the same string in both languages.

Formally, let $V^i = \{w_1^i, w_2^i, \ldots, w_{k_i}^i\}$ be the vocabulary of the corpus T^i composed of document expressed in the language L^i, let $V^* = \bigcup_i V^i$ be the set of all the terms in all the languages, and let $k^* = |V^*|$ be the cardinality of this set. Let $\mathcal{D} = \{D_1, D_2, ..., D_d\}$ be a set of domains. A DM is fully defined by a $k^* \times d$ *domain matrix* $\mathbf{D}$ representing in each cell $\mathbf{d_{i,z}}$ the *domain relevance* of the i^{th} term of V^* with respect to the domain D_z. The domain matrix $\mathbf{D}$ is used to define a function $\mathcal{D} : \mathbf{R}^{k^*} \rightarrow \mathbf{R}^d$, that maps the document vectors $\vec{t_j}$ expressed into the multilingual classical VSM (see Section 2.1), into the vectors $\vec{t'_j}$ in the *multilingual domain VSM*. The function $\mathcal{D}$ is defined by[2]

$$\mathcal{D}(\vec{t_j}) = \vec{t_j}(\mathbf{I^{IDF}D}) = \vec{t'_j} \qquad (1)$$

where $\mathbf{I^{IDF}}$ is a diagonal matrix such that $i_{i,l}^{IDF} = IDF(w_i^l)$, $\vec{t_j}$ is represented as a row vector, and $IDF(w_i^l)$ is the *Inverse Document Frequency* of w_i^l evaluated in the corpus T^l.

In this work we exploit Latent Semantic Analysis (LSA) (Deerwester et al., 1990) to automatically acquire a MDM from comparable corpora. LSA is an unsupervised technique for estimating the similarity among texts and terms in a large corpus. In the monolingual settings LSA is performed by means of a Singular Value Decomposition (SVD) of the term-by-document matrix $\mathbf{T}$ describing the corpus. SVD decomposes the term-by-document matrix $\mathbf{T}$ into three matrixes $\mathbf{T} \simeq \mathbf{V\Sigma_{k'}U}^T$ where $\mathbf{\Sigma_{k'}}$ is the diagonal $k \times k$ matrix containing the highest $k' \ll k$ eigenvalues of $\mathbf{T}$, and all the remaining elements are set to 0. The parameter k' is the dimensionality of the Domain VSM and can be fixed in advance (i.e. $k' = d$).

In the literature (Littman et al., 1998) LSA has been used in multilingual settings to define a multilingual space in which texts in different languages can be represented and compared. In that work LSA strongly relied on the availability of aligned parallel corpora: documents in all the languages are represented in a term-by-document matrix (see Figure 1) and then the columns corresponding to sets of translated documents are collapsed (i.e. they are substituted by their sum) before starting the LSA process. The effect of this step is to merge the subspaces (i.e. the right and the left sectors of the matrix in Figure 1) in which

[2] In (Wong et al., 1985) the formula 1 is used to define a Generalized Vector Space Model, of which the Domain VSM is a particular instance.

the documents have been originally represented.

In this paper we propose a variation of this strategy, performing a multilingual LSA in the case in which an aligned parallel corpus is not available. It exploits the presence of common words among different languages in the term-by-document matrix. The SVD process has the effect of creating a LSA space in which documents in both languages are represented. Of course, the higher the number of common words, the more information will be provided to the SVD algorithm to find common LSA dimension for the two languages. The resulting LSA dimensions can be perceived as multilingual clusters of terms and document. LSA can then be used to define a Multilingual Domain Matrix $\mathbf{D_{LSA}}$. For further details see (Gliozzo and Strapparava, 2005).

As Kernel Methods are the state-of-the-art supervised framework for learning and they have been successfully adopted to approach the TC task (Joachims, 2002), we chose this framework to perform all our experiments, in particular Support Vector Machines[3]. Taking into account the external knowledge provided by a MDM it is possible estimate the topic similarity among two texts expressed in different languages, with the following kernel:

$$K_D(t_i, t_j) = \frac{\langle \mathcal{D}(t_i), \mathcal{D}(t_j) \rangle}{\sqrt{\langle \mathcal{D}(t_j), \mathcal{D}(t_j) \rangle \langle \mathcal{D}(t_i), \mathcal{D}(t_i) \rangle}} \tag{2}$$

where $\mathcal{D}$ is defined as in equation 1.

Note that when we want to estimate the similarity in the standard Multilingual VSM, as described in Section 2.1, we can use a simple *bag_of_words* kernel. The BoW kernel is a particular case of the Domain Kernel, in which $\mathbf{D} = \mathbf{I}$, and $\mathbf{I}$ is the identity matrix. In the evaluation typically we consider the BoW Kernel as a baseline.

4 Exploiting Bilingual Dictionaries

When bilingual resources are available it is possible to augment the the "common" portion of the matrix in Figure 1. In our experiments we exploit two alternative multilingual resources: MultiWordNet and the Collins English-Italian bilingual dictionary.

MultiWordNet[4]. It is a multilingual computational lexicon, conceived to be strictly aligned with the Princeton WordNet. The available languages are Italian, Spanish, Hebrew and Romanian. In our experiment we used the English and the Italian components. The last version of the Italian WordNet contains around 58,000 Italian word senses and 41,500 lemmas organized into 32,700 synsets aligned whenever possible with WordNet English synsets. The Italian synsets are created in correspondence with the Princeton WordNet synsets, whenever possible, and semantic relations are imported from the corresponding English synsets. This implies that the synset index structure is the same for the two languages.

Thus for the all the monosemic words, we augment each text in the dataset with the corresponding *synset-id*, which act as an expansion of the "common" terms of the matrix in Figure 1. Adopting the methodology described in Section 3.1, we exploit these common sense-indexing to induce a second-order similarity for the other terms in the lexicons. We evaluate the performance of the cross-lingual text categorization, using both the BoW Kernel and the Multilingual Domain Kernel, observing that also in this case the leverage of the external knowledge brought by the MDM is effective.

It is also possible to augment each text with all the synset-ids of all the words (i.e. monosemic and polysemic) present in the dataset, hoping that the SVM machine learning device cut off the noise due to the inevitable *spurious* senses introduced in the training examples. Obviously in this case, differently from the "monosemic" enrichment seen above, it does not make sense to apply any dimensionality reduction supplied by the Multilingual Domain Model (i.e. the resulting second-order relations among terms and documents produced on a such "extended" corpus should not be meaningful)[5].

Collins. The Collins machine-readable bilingual dictionary is a medium size dictionary including 37,727 headwords in the English Section and 32,602 headwords in the Italian Section.

This is a traditional dictionary, without sense indexing like the WordNet repository. In this case

[3] We adopted the efficient implementation freely available at *http://svmlight.joachims.org/*.

[4] Available at *http://multiwordnet.itc.it*.

[5] The use of a WSD system would help in this issue. However the rationale of this paper is to see how far it is possible to go with very few resources. And we suppose that a multilingual all-words WSD system is not easily available.

Categories	*English* Training	Test	Total	*Italian* Training	Test	Total
`Quality_of_Life`	5759	1989	7748	5781	1901	7682
`Made_in_Italy`	5711	1864	7575	6111	2068	8179
`Tourism`	5731	1857	7588	6090	2015	8105
`Culture_and_School`	3665	1245	4910	6284	2104	8388
Total	20866	6955	27821	24266	8088	32354

Table 2: Number of documents in the data set partitions

we follow the way, for each text of one language, to augment all the present words with the translation words found in the dictionary. For the same reason, we chose not to exploit the MDM, while experimenting along this way.

5 Evaluation

The CLTC task has been rarely attempted in the literature, and standard evaluation benchmark are not available. For this reason, we developed an evaluation task by adopting a news corpus kindly put at our disposal by *AdnKronos*, an important Italian news provider. The corpus consists of 32,354 Italian and 27,821 English news partitioned by AdnKronos into four fixed categories: QUALITY_OF_LIFE, MADE_IN_ITALY, TOURISM, CULTURE_AND_SCHOOL. The English and the Italian corpora are comparable, in the sense stated in Section 2, i.e. they cover the same topics and the same period of time. Some news stories are translated in the other language (but *no* alignment indication is given), some others are present only in the English set, and some others only in the Italian. The average length of the news stories is about 300 words. We randomly split both the English and Italian part into 75% training and 25% test (see Table 2). We processed the corpus with PoS taggers, keeping only nouns, verbs, adjectives and adverbs.

Table 3 reports the vocabulary dimensions of the English and Italian training partitions, the vocabulary of the merged training, and how many common lemmata are present (about 14% of the total). Among the common lemmata, 97% are nouns and most of them are proper nouns. Thus the initial term-by-document matrix is a 43,384 × 45,132 matrix, while the $\mathbf{D_{LSA}}$ was acquired using 400 dimensions.

As far as the CLTC task is concerned, we tried the many possible options. In all the cases we trained on the English part and we classified the Italian part, and we trained on the Italian and classified on the English part. When used, the MDM was acquired running the SVD only on the joint (English and Italian) training parts.

	# lemmata
English training	22,704
Italian training	26,404
English + Italian	43,384
common lemmata	5,724

Table 3: Number of lemmata in the training parts of the corpus

Using only comparable corpora. Figure 2 reports the performance without any use of bilingual dictionaries. Each graph show the learning curves respectively using a BoW kernel (that is considered here as a baseline) and the multilingual domain kernel. We can observe that the latter largely outperform a standard BoW approach. Analyzing the learning curves, it is worth noting that when the quantity of training increases, the performance becomes better and better for the Multilingual Domain Kernel, suggesting that with more available training it could be possible to improve the results.

Using bilingual dictionaries. Figure 3 reports the learning curves exploiting the addition of the synset-ids of the monosemic words in the corpus. As expected the use of a multilingual repository improves the classification results. Note that the MDM outperforms the BoW kernel.

Figure 4 shows the results adding in the English and Italian parts of the corpus all the synset-ids (i.e. monosemic and polisemic) and all the translations found in the Collins dictionary respectively. These are the best results we get in our experiments. In these figures we report also the performance of the corresponding monolingual TC (we used the SVM with the BoW kernel), which can be considered as an upper bound. We can observe that the CLTC results are quite close to the performance obtained in the monolingual classification tasks.

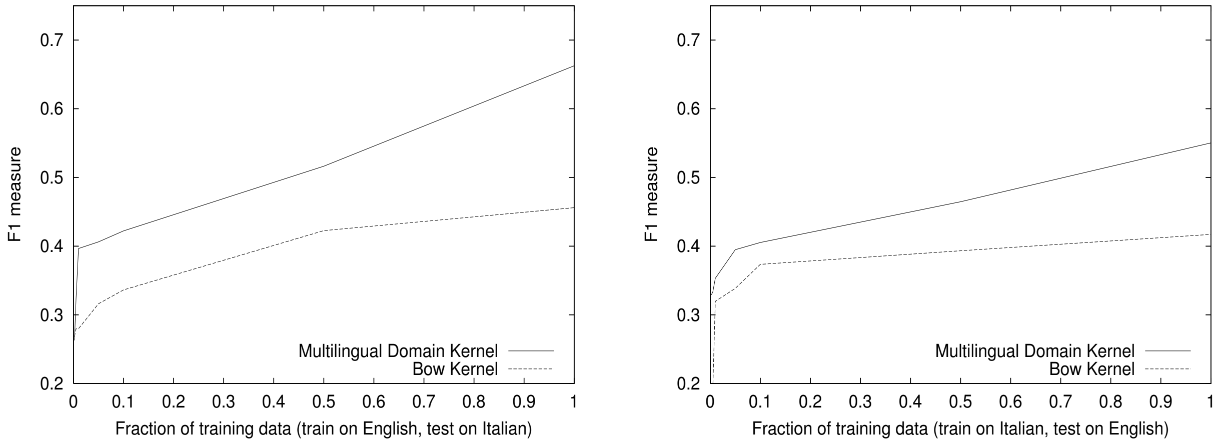


Figure 2: Cross-language learning curves: no use of bilingual dictionaries

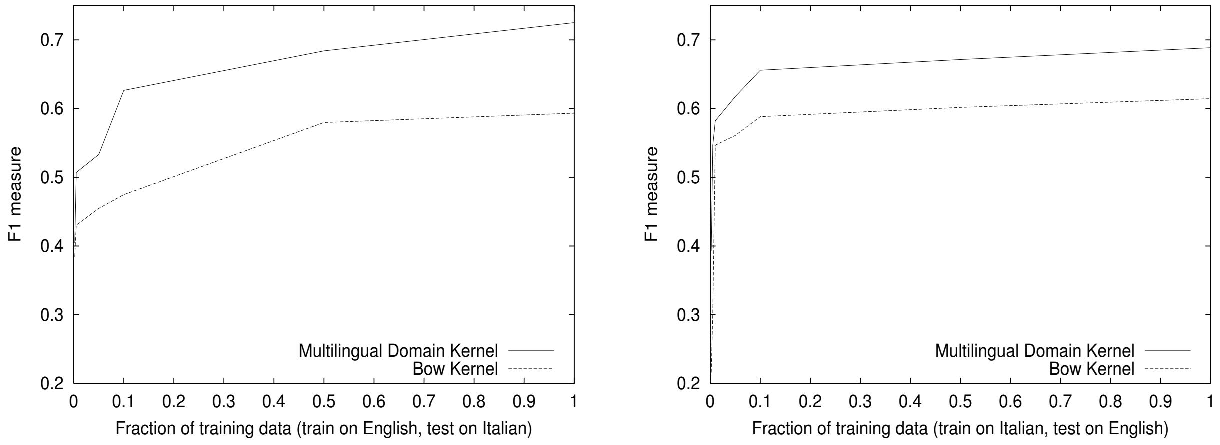


Figure 3: Cross-language learning curves: monosemic synsets from MultiWordNet

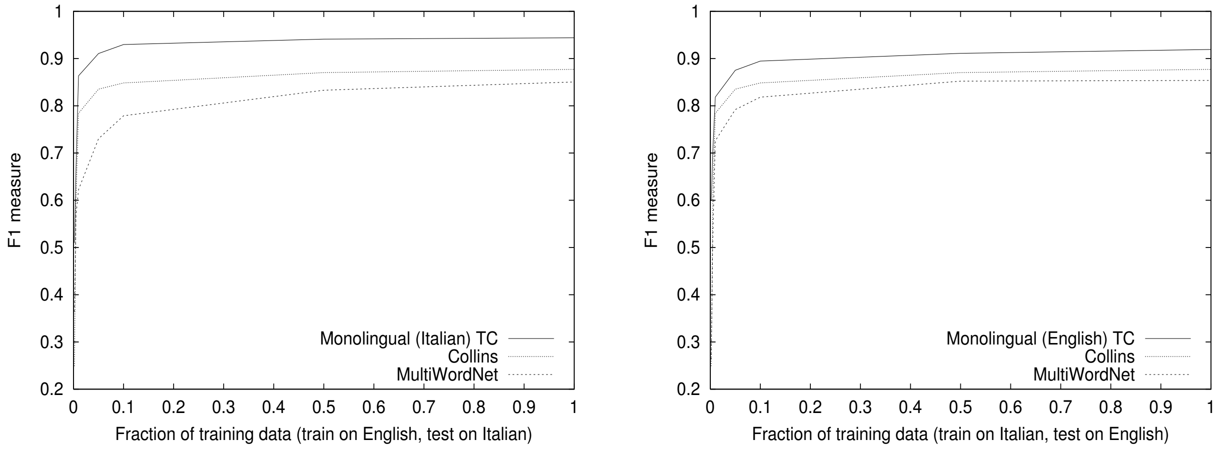


Figure 4: Cross-language learning curves: all synsets from MultiWordNet // All translations from Collins

6 Conclusion and Future Work

In this paper we have shown that the problem of cross-language text categorization on comparable corpora is a feasible task. In particular, it is possible to deal with it even when no bilingual resources are available. On the other hand when it is possible to exploit bilingual repositories, such as a synset-aligned WordNet or a bilingual dictionary, the obtained performance is close to that achieved for the monolingual task. In any case we think that our methodology is low-cost and simple, and it can represent a technologically viable solution for multilingual problems. For the future we try to explore also the use of a word sense disambiguation all-words system. We are confident that even with the actual state-of-the-art WSD performance, we can improve the actual results.

Acknowledgments

This work has been partially supported by the ONTOTEXT (From Text to Knowledge for the Semantic Web) project, funded by the Autonomous Province of Trento under the FUP-2004 program.

References

N. Bel, C. Koster, and M. Villegas. 2003. Cross-lingual text categorization. In *Proceedings of European Conference on Digital Libraries (ECDL)*, Trondheim, August.

C. Callison-Burch, D. Talbot, and M. Osborne. 2004. Statistical machine translation with word-and sentence-aligned parallel corpora. In *Proceedings of ACL-04*, Barcelona, Spain, July.

S. Deerwester, S. T. Dumais, G. W. Furnas, T.K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

E. Gaussier, J. M. Renders, I. Matveeva, C. Goutte, and H. Dejean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *Proceedings of ACL-04*, Barcelona, Spain, July.

A. Gliozzo and C. Strapparava. 2005. Cross language text categorization by acquiring multilingual domain models from comparable corpora. In *Proc. of the ACL Workshop on Building and Using Parallel Texts (in conjunction of ACL-05)*, University of Michigan, Ann Arbor, June.

A. Gliozzo, C. Strapparava, and I. Dagan. 2004. Unsupervised and supervised exploitation of semantic domains in lexical disambiguation. *Computer Speech and Language*, 18:275–299.

T. Joachims. 2002. *Learning to Classify Text using Support Vector Machines*. Kluwer Academic Publishers.

P. Koehn and K. Knight. 2002. Learning a translation lexicon from monolingual corpora. In *Proceedings of ACL Workshop on Unsupervised Lexical Acquisition*, Philadelphia, July.

M. Littman, S. Dumais, and T. Landauer. 1998. Automatic cross-language information retrieval using latent semantic indexing. In G. Grefenstette, editor, *Cross Language Information Retrieval*, pages 51–62. Kluwer Academic Publishers.

D. Melamed. 2001. *Empirical Methods for Exploiting Parallel Texts*. The MIT Press.

L. Rigutini, M. Maggini, and B. Liu. 2005. An EM based training algorithm for cross-language text categorizaton. In *Proceedings of Web Intelligence Conference (WI-2005)*, Compiègne, France, September.

C. Strapparava, A. Gliozzo, and C. Giuliano. 2004. Pattern abstraction and term similarity for word sense disambiguation. In *Proceedings of SENSEVAL-3*, Barcelona, Spain, July.

S.K.M. Wong, W. Ziarko, and P.C.N. Wong. 1985. Generalized vector space model in information retrieval. In *Proceedings of the 8^{th} ACM SIGIR Conference*.

A Progressive Feature Selection Algorithm for Ultra Large Feature Spaces

Qi Zhang
Computer Science Department
Fudan University
Shanghai 200433, P.R. China
qi_zhang@fudan.edu.cn

Fuliang Weng
Research and Technology Center
Robert Bosch Corp.
Palo Alto, CA 94304, USA
fuliang.weng@rtc.bosch.com

Zhe Feng
Research and Technology Center
Robert Bosch Corp.
Palo Alto, CA 94304, USA
zhe.feng@rtc.bosch.com

Abstract

Recent developments in statistical modeling of various linguistic phenomena have shown that additional features give consistent performance improvements. Quite often, improvements are limited by the number of features a system is able to explore. This paper describes a novel progressive training algorithm that selects features from virtually unlimited feature spaces for conditional maximum entropy (CME) modeling. Experimental results in edit region identification demonstrate the benefits of the progressive feature selection (PFS) algorithm: the PFS algorithm maintains the same accuracy performance as previous CME feature selection algorithms (e.g., Zhou et al., 2003) when the same feature spaces are used. When additional features and their combinations are used, the PFS gives 17.66% relative improvement over the previously reported best result in edit region identification on Switchboard corpus (Kahn et al., 2005), which leads to a 20% relative error reduction in parsing the Switchboard corpus when gold edits are used as the upper bound.

1 Introduction

Conditional Maximum Entropy (CME) modeling has received a great amount of attention within natural language processing community for the past decade (e.g., Berger et al., 1996; Reynar and Ratnaparkhi, 1997; Koeling, 2000; Malouf, 2002; Zhou et al., 2003; Riezler and Vasserman, 2004). One of the main advantages of CME modeling is the ability to incorporate a variety of features in a uniform framework with a sound mathematical foundation. Recent improvements on the original incremental feature selection (IFS) algorithm, such as Malouf (2002) and Zhou et al. (2003), greatly speed up the feature selection process. However, like many other statistical modeling algorithms, such as boosting (Schapire and Singer, 1999) and support vector machine (Vapnik 1995), the algorithm is limited by the size of the defined feature space. Past results show that larger feature spaces tend to give better results. However, finding a way to include an unlimited amount of features is still an open research problem.

In this paper, we propose a novel progressive feature selection (PFS) algorithm that addresses the feature space size limitation. The algorithm is implemented on top of the Selective Gain Computation (SGC) algorithm (Zhou et al., 2003), which offers fast training and high quality models. Theoretically, the new algorithm is able to explore an unlimited amount of features. Because of the improved capability of the CME algorithm, we are able to consider many new features and feature combinations during model construction.

To demonstrate the effectiveness of our new algorithm, we conducted a number of experiments on the task of identifying edit regions, a practical task in spoken language processing. Based on the convention from Shriberg (1994) and Charniak and Johnson (2001), a disfluent spoken utterance is divided into three parts: the *reparandum*, the part that is repaired; the *inter-*

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 561–568,
Sydney, July 2006. ©2006 Association for Computational Linguistics

regnum, which can be filler words or empty; and the *repair/repeat*, the part that replaces or repeats the reparandum. The first two parts combined are called an *edit* or *edit region*. An example is shown below:

$$\underbrace{\text{It is,}}_{\text{reparandum}}\ \underbrace{\text{you know,}}_{\text{interregnum}}\ \underbrace{\text{this is}}_{\text{repair}}\text{ a tough problem.}$$

In section 2, we briefly review the CME modeling and SGC algorithm. Then, section 3 gives a detailed description of the PFS algorithm. In section 4, we describe the Switchboard corpus, features used in the experiments, and the effectiveness of the PFS with different feature spaces. Section 5 concludes the paper.

2 Background

Before presenting the PFS algorithm, we first give a brief review of the conditional maximum entropy modeling, its training process, and the SGC algorithm. This is to provide the background and motivation for our PFS algorithm.

2.1 Conditional Maximum Entropy Model

The goal of CME is to find the most uniform conditional distribution of y given observation x, $p(y|x)$, subject to constraints specified by a set of features $f_i(x, y)$, where features typically take the value of either 0 or 1 (Berger et al., 1996). More precisely, we want to maximize

$$H(p) = -\sum_{x,y} \tilde{p}(x)p(y|x)\log(p(y|x)) \qquad (1)$$

given the constraints:

$$E(f_i) = \tilde{E}(f_i) \qquad (2)$$

where

$$\tilde{E}(f_i) = \sum_{x,y} \tilde{p}(x,y) f_i(x,y)$$

is the empirical expected feature count from the training data and

$$E(f_i) = \sum_{x,y} \tilde{p}(x)p(y|x) f_i(x,y)$$

is the feature expectation from the conditional model $p(y|x)$.

This results in the following exponential model:

$$p(y|x) = \frac{1}{Z(x)}\exp\left(\sum_j \lambda_j f_j(x,y)\right) \qquad (3)$$

where λ_j is the weight corresponding to the feature f_j, and $Z(x)$ is a normalization factor.

A variety of different phenomena, including lexical, structural, and semantic aspects, in natural language processing tasks can be expressed in terms of features. For example, a feature can be whether the word in the current position is a verb, or the word is a particular lexical item. A feature can also be about a particular syntactic subtree, or a dependency relation (e.g., Charniak and Johnson, 2005).

2.2 Selective Gain Computation Algorithm

In real world applications, the number of possible features can be in the millions or beyond. Including all the features in a model may lead to data over-fitting, as well as poor efficiency and memory overflow. Good feature selection algorithms are required to produce efficient and high quality models. This leads to a good amount of work in this area (Ratnaparkhi et al., 1994; Berger et al., 1996; Pietra et al, 1997; Zhou et al., 2003; Riezler and Vasserman, 2004)

In the most basic approach, such as Ratnaparkhi et al. (1994) and Berger et al. (1996), training starts with a uniform distribution over all values of y and an empty feature set. For each candidate feature in a predefined feature space, it computes the likelihood gain achieved by including the feature in the model. The feature that maximizes the gain is selected and added to the current model. This process is repeated until the gain from the best candidate feature only gives marginal improvement. The process is very slow, because it has to re-compute the gain for every feature at each selection stage, and the computation of a parameter using Newton's method becomes expensive, considering that it has to be repeated many times.

The idea behind the SGC algorithm (Zhou et al., 2003) is to use the gains computed in the previous step as approximate upper bounds for the subsequent steps. The gain for a feature needs to be re-computed only when the feature reaches the top of a priority queue ordered by gain. In other words, this happens when the feature is the top candidate for inclusion in the model. If the re-computed gain is smaller than that of the next candidate in the list, the feature is re-ranked according to its newly computed gain, and the feature now at the top of the list goes through the same gain re-computing process.

This heuristics comes from evidences that the gains become smaller and smaller as more and more good features are added to the model. This can be explained as follows: assume that the Maximum Likelihood (ML) estimation lead to the best model that reaches a ML value. The ML value is the upper bound. Since the gains need to be positive to proceed the process, the difference between the Likelihood of the current and the ML value becomes smaller and smaller. In other words, the possible gain each feature may add to the model gets smaller. Experiments in Zhou et al. (2003) also confirm the prediction that the gains become smaller when more and more features are added to the model, and the gains do not get unexpectively bigger or smaller as the model grows. Furthermore, the experiments in Zhou et al. (2003) show no significant advantage for looking ahead beyond the first element in the feature list. The SGC algorithm runs hundreds to thousands of times faster than the original IFS algorithm without degrading classification performance. We used this algorithm for it enables us to find high quality CME models quickly.

The original SGC algorithm uses a technique proposed by Darroch and Ratcliff (1972) and elaborated by Goodman (2002): when considering a feature f_i, the algorithm only modifies those un-normalized conditional probabilities:

$$\exp\left(\sum_j \lambda_j f_j(x, y)\right)$$

for (x, y) that satisfy $f_i(x, y)=1$, and subsequently adjusts the corresponding normalizing factors $Z(x)$ in (3). An implementation often uses a mapping table, which maps features to the training instance pairs (x, y).

3 Progressive Feature Selection Algorithm

In general, the more contextual information is used, the better a system performs. However, richer context can lead to combinatorial explosion of the feature space. When the feature space is huge (e.g., in the order of tens of millions of features or even more), the SGC algorithm exceeds the memory limitation on commonly available computing platforms with gigabytes of memory.

To address the limitation of the SGC algorithm, we propose a progressive feature selection algorithm that selects features in multiple rounds. The main idea of the PFS algorithm is to split the feature space into tractable disjoint sub-spaces such that the SGC algorithm can be performed on each one of them. In the merge step, the features that SGC selects from different sub-spaces are merged into groups. Instead of re-generating the feature-to-instance mapping table for each sub-space during the time of splitting and merging, we create the new mapping table from the previous round's tables by collecting those entries that correspond to the selected features. Then, the SGC algorithm is performed on each of the feature groups and new features are selected from each of them. In other words, the feature space splitting and subspace merging are performed mainly on the feature-to-instance mapping tables. This is a key step that leads to this very efficient PFS algorithm.

At the beginning of each round for feature selection, a uniform prior distribution is always assumed for the new CME model. A more precise description of the PFS algorithm is given in Table 1, and it is also graphically illustrated in Figure 1.

```
Given:
    Feature space $\mathbf{F}^{(0)} = \{f_1^{(0)}, f_2^{(0)}, \ldots, f_N^{(0)}\}$,
    step_num = $m$,  select_factor = $s$

1. Split the feature space into $N_1$ parts
    $\{\mathbf{F}_1^{(1)}, \mathbf{F}_2^{(1)}, \ldots, \mathbf{F}_{N_1}^{(1)}\} = \text{split}(\mathbf{F}^{(0)})$

2. for $k$=1 to m-1 do
      //2.1 Feature selection
      for each feature space $\mathbf{F}_i^{(k)}$ do
          $\mathbf{FS}_i^{(k)} = \text{SGC}(\mathbf{F}_i^{(k)}, s)$
      //2.2 Combine selected features
      $\{\mathbf{F}_1^{(k+1)}, \ldots, \mathbf{F}_{N_{k+1}}^{(k+1)}\} =$
                 $\text{merge}(\mathbf{FS}_1^{(k)}, \ldots, \mathbf{FS}_{N_k}^{(k)})$

3. Final feature selection & optimization
    $\mathbf{F}^{(m)} = \text{merge}(\mathbf{FS}_1^{(m-1)}, \ldots, \mathbf{FS}_{N_{m-1}}^{(m-1)})$
    $\mathbf{FS}^{(m)} = \text{SGC}(\mathbf{F}^{(m)}, s)$
    $\mathbf{M}_{final} = \text{Opt}(\mathbf{FS}^{(m)})$
```

Table 1. The PFS algorithm.

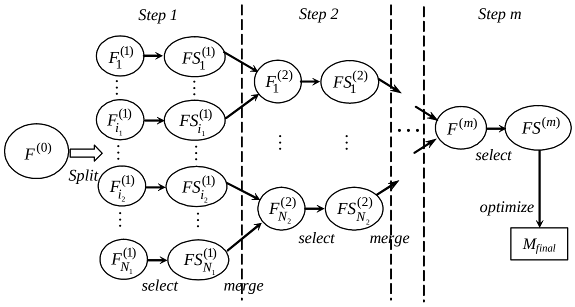


Figure 1. Graphic illustration of PFS algorithm.

In Table 1, SGC() invokes the SGC algorithm, and Opt() optimizes feature weights. The functions split() and merge() are used to split and merge the feature space respectively.

Two variations of the split() function are investigated in the paper and they are described below:

1. **random-split**: randomly split a feature space into *n*- disjoint subspaces, and select an equal amount of features for each feature subspace.
2. **dimension-based-split**: split a feature space into disjoint subspaces based on fea-

ture dimensions/variables, and select the number of features for each feature subspace with a certain distribution.

We use a simple method for merge() in the experiments reported here, i.e., adding together the features from a set of selected feature subspaces.

One may image other variations of the split() function, such as allowing overlapping subspaces. Other alternatives for merge() are also possible, such as randomly grouping the selected feature subspaces in the dimension-based split. Due to the limitation of the space, they are not discussed here.

This approach can in principle be applied to other machine learning algorithms as well.

4 Experiments with PFS for Edit Region Identification

In this section, we will demonstrate the benefits of the PFS algorithm for identifying edit regions. The main reason that we use this task is that the edit region detection task uses features from several levels, including prosodic, lexical, and syntactic ones. It presents a big challenge to find a set of good features from a huge feature space.

First we will present the additional features that the PFS algorithm allows us to include. Then, we will briefly introduce the variant of the Switchboard corpus used in the experiments. Finally, we will compare results from two variants of the PFS algorithm.

4.1 Edit Region Identification Task

In spoken utterances, disfluencies, such as self-editing, pauses and repairs, are common phenomena. Charniak and Johnson (2001) and Kahn et al. (2005) have shown that improved edit region identification leads to better parsing accuracy – they observe a relative reduction in parsing f-score error of 14% (2% absolute) between automatic and oracle edit removal.

The focus of our work is to show that our new PFS algorithm enables the exploration of much larger feature spaces for edit identification – including prosodic features, their confidence scores, and various feature combinations – and consequently, it further improves edit region identification. Memory limitation prevents us from including all of these features in experiments using the boosting method described in Johnson and Charniak (2004) and Zhang and Weng (2005). We couldn't use the new features with the SGC algorithm either for the same reason.

The features used here are grouped according to variables, which define feature sub-spaces as in Charniak and Johnson (2001) and Zhang and Weng (2005). In this work, we use a total of 62 variables, which include 16 [1] variables from Charniak and Johnson (2001) and Johnson and Charniak (2004), an additional 29 variables from Zhang and Weng (2005), 11 hierarchical POS tag variables, and 8 prosody variables (labels and their confidence scores). Furthermore, we explore 377 combinations of these 62 variables, which include 40 combinations from Zhang and Weng (2005). The complete list of the variables is given in Table 2, and the combinations used in the experiments are given in Table 3. One additional note is that some features are obtained after the rough copy procedure is performed, where we used the same procedure as the one by Zhang and Weng (2005). For a fair comparison with the work by Kahn et al. (2005), word fragment information is retained.

4.2 The Re-segmented Switchboard Data

In order to include prosodic features and be able to compare with the state-oft-art, we use the University of Washington re-segmented Switchboard corpus, described in Kahn et al. (2005). In this corpus, the Switchboard sentences were segmented into V5-style sentence-like units (SUs) (LDC, 2004). The resulting sentences fit more closely with the boundaries that can be detected through automatic procedures (e.g., Liu et al., 2005). Because the edit region identification results on the original Switchboard are not directly comparable with the results on the newly segmented data, the state-of-art results reported by Charniak and Johnson (2001) and Johnson and Charniak (2004) are repeated on this new corpus by Kahn et al. (2005).

The re-segmented UW Switchboard corpus is labeled with a simplified subset of the ToBI prosodic system (Ostendorf et al., 2001). The three simplified labels in the subset are *p*, 1 and 4, where *p* refers to a general class of disfluent boundaries (e.g., word fragments, abruptly shortened words, and hesitation); 4 refers to break level 4, which describes a boundary that has a boundary tone and phrase-final lengthening;

[1] Among the original 18 variables, two variables, P_f and T_f are not used in our experiments, because they are mostly covered by the other variables. Partial word flags only contribute to 3 features in the final selected feature list.

Categories		Variable Name	Short Description
Words	Orthographic Words	W_{-5}, … , W_{+5}	Words at the current position and the left and right 5 positions.
	Partial Word Flags	P_{-3}, …, P_{+3}	Partial word flags at the current position and the left and right 3 positions
	Distance	D_{INTJ}, D_{W}, D_{Bigram}, $D_{Trigram}$	Distance features
Tags	POS Tags	T_{-5}, …, T_{+5}	POS tags at the current position and the left and right 5 positions.
	Hierarchical POS Tags (HTag)	HT_{-5}, …, HT_{+5}	Hierarchical POS tags at the current position and the left and right 5 positions.
Rough Copy	HTag Rough Copy	N_m, N_n, N_i, N_l, N_r, T_i	Hierarchical POS rough copy features.
	Word Rough Copy	WN_m, WN_i, WN_l, WN_r	Word rough copy features.
Prosody	Prosody Labels	PL_0, …, PL_3	Prosody label with largest post possibility at the current position and the right 3 positions.
	Prosody Scores	PC_0, …, PC_3	Prosody confidence at the current position and the right 3 positions.

Table 2. A complete list of variables used in the experiments.

Categories			Short Description	Number of Combinations
Tags	HTagComb		Combinations among *Hierarchical POS Tags*	55
Words	WTComb	OrthWordComb	Combinations among *Orthographic Words*	55
Tags		WTTComb	Combinations of *Orthographic Words* and *POS Tags*; Combination among *POS Tags*	176
Rough Copy	RCComb		Combinations of *HTag Rough Copy* and *Word Rough Copy*	55
Prosody	PComb		Combinations among *Prosody*, and with *Words*	36

Table 3. All the variable combinations used in the experiments.

and 1 is used to include the break index levels BL 0, 1, 2, and 3. Since the majority of the corpus is labeled via automatic methods, the f-scores for the prosodic labels are not high. In particular, 4 and *p* have f-scores of about 70% and 60% respectively (Wong et al., 2005). Therefore, in our experiments, we also take prosody confidence scores into consideration.

Besides the symbolic prosody labels, the corpus preserves the majority of the previously annotated syntactic information as well as edit region labels.

In following experiments, to make the results comparable, the same data subsets described in Kahn et al. (2005) are used for training, developing and testing.

4.3 Experiments

The best result on the UW Switchboard for edit region identification uses a TAG-based approach (Kahn et al., 2005). On the original Switchboard corpus, Zhang and Weng (2005) reported nearly 20% better results using the boosting method with a much larger feature space[2]. To allow comparison with the best past results, we create a new CME baseline with the same set of features as that used in Zhang and Weng (2005).

We design a number of experiments to test the following hypotheses:

1. PFS can include a huge number of new features, which leads to an overall performance improvement.
2. Richer context, represented by the combinations of different variables, has a positive impact on performance.
3. When the same feature space is used, PFS performs equally well as the original SGC algorithm.

The new models from the PFS algorithm are trained on the training data and tuned on the development data. The results of our experiments on the test data are summarized in Table 4. The first three lines show that the TAG-based approach is outperformed by the new CME baseline (line 3) using all the features in Zhang and Weng (2005). However, the improvement from

[2] PFS is not applied to the boosting algorithm at this time because it would require significant changes to the available algorithm.

Feature Space Codes	number of features	Results on test data		
		Precision	Recall	F-Value
TAG-based result on UW-SWBD reported in Kahn et al. (2005)				**78.20**
CME with all the variables from Zhang and Weng (2005)	2412382	89.42	71.22	79.29
CME with all the variables from Zhang and Weng (2005) + post	2412382	87.15	73.78	**79.91**
+HTag +HTagComb +WTComb +RCComb	17116957	90.44	72.53	80.50
+HTag +HTagComb +WTComb +RCComb +PL_0 … PL_3	17116981	88.69	74.01	80.69
+HTag +HTagComb +WTComb +RCComb +PComb: without cut	20445375	89.43	73.78	80.86
+HTag +HTagComb +WTComb +RCComb +PComb: cut2	19294583	88.95	74.66	**81.18**
+HTag +HTagComb +WTComb +RCComb +PComb: cut2 +Gau	19294583	90.37	74.40	81.61
+HTag +HTagComb +WTComb +RCComb +PComb: cut2 +post	19294583	86.88	77.29	81.80
+HTag +HTagComb +WTComb +RCComb +PComb: cut2 +Gau +post	19294583	87.79	77.02	**82.05**

Table 4. Summary of experimental results with PFS.

CME is significantly smaller than the reported results using the boosting method. In other words, using CME instead of boosting incurs a performance hit.

The next four lines in Table 4 show that additional combinations of the feature variables used in Zhang and Weng (2005) give an absolute improvement of more than 1%. This improvement is realized through increasing the search space to more than 20 million features, 8 times the maximum size that the original boosting and CME algorithms are able to handle.

Table 4 shows that prosody labels alone make no difference in performance. Instead, for each position in the sentence, we compute the entropy of the distribution of the labels' confidence scores. We normalize the entropy to the range [0, 1], according to the formula below:

$$score = 1 - H(p)/H(Uniform) \qquad (4)$$

Including this feature does result in a good improvement. In the table, *cut2* means that we equally divide the feature scores into 10 buckets and any number below 0.2 is ignored. The total contribution from the combined feature variables leads to a 1.9% absolute improvement. This confirms the first two hypotheses.

When Gaussian smoothing (Chen and Rosenfeld, 1999), labeled as *+Gau*, and post-processing (Zhang and Weng, 2005), labeled as *+post*, are added, we observe 17.66% relative improvement (or 3.85% absolute) over the previous best f-score of 78.2 from Kahn et al. (2005).

To test hypothesis 3, we are constrained to the feature spaces that both PFS and SGC algorithms can process. Therefore, we take all the variables from Zhang and Weng (2005) as the feature space for the experiments. The results are listed in Table 5. We observed no f-score degradation with PFS. Surprisingly, the total amount of time PFS spends on selecting its best features is smaller than the time SGC uses in selecting its best features. This confirms our hypothesis 3.

Split / Non-split	Results on test data		
	Precision	Recall	F-Value
non-split	89.42	71.22	79.29
split by 4 parts	89.67	71.68	79.67
split by 10 parts	89.65	71.29	79.42

Table 5. Comparison between PFS and SGC with all the variables from Zhang and Weng (2005).

The last set of experiments for edit identification is designed to find out what split strategies PFS algorithm should adopt in order to obtain good results. Two different split strategies are tested here. In all the experiments reported so far, we use 10 random splits, i.e., all the features are randomly assigned to 10 subsets of equal size. We may also envision a split strategy that divides the features based on feature variables (or *dimensions*), such as word-based, tag-based, etc. The four dimensions used in the experiments are listed as the top categories in Tables 2 and 3, and the results are given in Table 6.

Split Criteria	Allocation Criteria	Results on test data		
		Precision	Recall	F-Value
Random	Uniform	88.95	74.66	81.18
Dimension	Uniform	89.78	73.42	80.78
Dimension	Prior	89.78	74.01	81.14

Table 6. Comparison of split strategies using feature space +HTag+HTagComb+WTComb+RCComb+PComb: cut2

In Table 6, the first two columns show criteria for splitting feature spaces and the number of features to be allocated for each group. *Random* and *Dimension* mean random-split and dimension-based-split, respectively. When the criterion is *Random*, the features are allocated to different groups randomly, and each group gets the same number of features. In the case of dimension-based split, we determine the number of features allocated for each dimension in two ways. When the split is *Uniform*, the same number of features is allocated for each dimension. When the split is *Prior*, the number of features to be allocated in each dimension is determined in proportion to the importance of each dimension. To determine the importance, we use the distribution of the selected features from each dimension in the model "+ HTag + HTagComb + WTComb + RCComb + PComb: cut2", namely: Word-based 15%, Tag-based 70%, RoughCopy-based 7.5% and Prosody-based 7.5%[3]. From the results, we can see no significant difference between the random-split and the dimension-based-split.

To see whether the improvements are translated into parsing results, we have conducted one more set of experiments on the UW Switchboard corpus. We apply the latest version of Charniak's parser (2005-08-16) and the same procedure as Charniak and Johnson (2001) and Kahn et al. (2005) to the output from our best edit detector in this paper. To make it more comparable with the results in Kahn et al. (2005), we repeat the same experiment with the gold edits, using the latest parser. Both results are listed in Table 7. The difference between our best detector and the gold edits in parsing (1.51%) is smaller than the difference between the TAG-based detector and the gold edits (1.9%). In other words, if we use the gold edits as the upper bound, we see a relative error reduction of 20.5%.

Methods	Edit *F-score*	Parsing *F-score*		
		Reported in Kahn et al. (2005)	Latest Charniak Parser	Diff. with Oracle
Oracle	100	86.9	87.92	--
Kahn et al. (2005)	78.2	85.0	--	**1.90**
PFS best results	82.05	--	86.41	**1.51**

Table 7. Parsing F-score various different edit region identification results.

5 Conclusion

This paper presents our progressive feature selection algorithm that greatly extends the feature space for conditional maximum entropy modeling. The new algorithm is able to select features from feature space in the order of tens of millions in practice, i.e., 8 times the maximal size previous algorithms are able to process, and unlimited space size in theory. Experiments on edit region identification task have shown that the increased feature space leads to 17.66% relative improvement (or 3.85% absolute) over the best result reported by Kahn et al. (2005), and 10.65% relative improvement (or 2.14% absolute) over the new baseline SGC algorithm with all the variables from Zhang and Weng (2005). We also show that symbolic prosody labels together with confidence scores are useful in edit region identification task.

In addition, the improvements in the edit identification lead to a relative 20% error reduction in parsing disfluent sentences when gold edits are used as the upper bound.

Acknowledgement

This work is partly sponsored by a NIST ATP funding. The authors would like to express their many thanks to Mari Ostendorf and Jeremy Kahn for providing us with the re-segmented UW Switchboard Treebank and the corresponding prosodic labels. Our thanks also go to Jeff Russell for his careful proof reading, and the anonymous reviewers for their useful comments. All the remaining errors are ours.

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22 (1): 39-71.

Eugene Charniak and Mark Johnson. 2001. Edit Detection and Parsing for Transcribed Speech. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, 118-126, Pittsburgh, PA, USA.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting of Association for Computational Linguistics*, 173-180, Ann Arbor, MI, USA.

Stanley Chen and Ronald Rosenfeld. 1999. A Gaussian Prior for Smoothing Maximum Entropy Mod-

[3] It is a bit of cheating to use the distribution from the selected model. However, even with this distribution, we do not see any improvement over the version with random-split.

els. *Technical Report CMUCS-99-108*, Carnegie Mellon University.

John N. Darroch and D. Ratcliff. 1972. Generalized Iterative Scaling for Log-Linear Models. In *Annals of Mathematical Statistics*, 43(5): 1470-1480.

Stephen A. Della Pietra, Vincent J. Della Pietra, and John Lafferty. 1997. Inducing Features of Random Fields. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4): 380-393.

Joshua Goodman. 2002. Sequential Conditional Generalized Iterative Scaling. In *Proceedings of the 40th Annual Meeting of Association for Computational Linguistics*, 9-16, Philadelphia, PA, USA.

Mark Johnson, 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*, 33-39, Barcelona, Spain.

Jeremy G. Kahn, Matthew Lease, Eugene Charniak, Mark Johnson, and Mari Ostendorf. 2005. Effective Use of Prosody in Parsing Conversational Speech. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing*, 233-240, Vancouver, Canada.

Rob Koeling. 2000. Chunking with Maximum Entropy Models. In *Proceedings of the CoNLL-2000 and LLL-2000*, 139-141, Lisbon, Portugal.

LDC. 2004. Simple MetaData Annotation Specification. *Technical Report of Linguistic Data Consortium*. (http://www.ldc.upenn.edu/Projects/MDE).

Yang Liu, Elizabeth Shriberg, Andreas Stolcke, Barbara Peskin, Jeremy Ang, Dustin Hillard, Mari Ostendorf, Marcus Tomalin, Phil Woodland and Mary Harper. 2005. Structural Metadata Research in the EARS Program. In *Proceedings of the 30th ICASSP*, volume V, 957-960, Philadelphia, PA, USA.

Robert Malouf. 2002. A Comparison of Algorithms for Maximum Entropy Parameter Estimation. In *Proceedings of the 6th Conference on Natural Language Learning (CoNLL-2002)*, 49-55, Taibei, Taiwan.

Mari Ostendorf, Izhak Shafran, Stefanie Shattuck-Hufnagel, Leslie Charmichael, and William Byrne. 2001. A Prosodically Labeled Database of Spontaneous Speech. In *Proceedings of the ISCA Workshop of Prosody in Speech Recognition and Understanding*, 119-121, Red Bank, NJ, USA.

Adwait Ratnaparkhi, Jeff Reynar and Salim Roukos. 1994. A Maximum Entropy Model for Prepositional Phrase Attachment. In *Proceedings of the ARPA Workshop on Human Language Technology*, 250-255, Plainsboro, NJ, USA.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A Maximum Entropy Approach to Identifying Sentence Boundaries. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, 16-19, Washington D.C., USA.

Stefan Riezler and Alexander Vasserman. 2004. Incremental Feature Selection and L1 Regularization for Relaxed Maximum-entropy Modeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 174-181, Barcelona, Spain.

Robert E. Schapire and Yoram Singer, 1999. Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37(3): 297-336.

Elizabeth Shriberg. 1994. *Preliminaries to a Theory of Speech Disfluencies*. Ph.D. Thesis, University of California, Berkeley.

Vladimir Vapnik. 1995. T*he Nature of Statistical Learning Theory*. Springer, New York, NY, USA.

Darby Wong, Mari Ostendorf, Jeremy G. Kahn. 2005. Using Weakly Supervised Learning to Improve Prosody Labeling. *Technical Report UWEETR-2005-0003*, University of Washington.

Qi Zhang and Fuliang Weng. 2005. Exploring Features for Identifying Edited Regions in Disfluent Sentences. In *Proc. of the 9th International Workshop on Parsing Technologies*, 179-185, Vancouver, Canada.

Yaqian Zhou, Fuliang Weng, Lide Wu, and Hauke Schmidt. 2003. A Fast Algorithm for Feature Selection in Conditional Maximum Entropy Modeling. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, 153-159, Sapporo, Japan.

Annealing Structural Bias in Multilingual Weighted Grammar Induction[*]

Noah A. Smith and **Jason Eisner**
Department of Computer Science / Center for Language and Speech Processing
Johns Hopkins University, Baltimore, MD 21218 USA
{nasmith,jason}@cs.jhu.edu

Abstract

We first show how a structural **locality bias** can improve the accuracy of state-of-the-art dependency grammar induction models trained by EM from unannotated examples (Klein and Manning, 2004). Next, by annealing the free parameter that controls this bias, we achieve further improvements. We then describe an alternative kind of structural bias, toward "broken" hypotheses consisting of partial structures over segmented sentences, and show a similar pattern of improvement. We relate this approach to contrastive estimation (Smith and Eisner, 2005a), apply the latter to grammar induction in six languages, and show that our new approach improves accuracy by 1–17% (absolute) over CE (and 8–30% over EM), achieving to our knowledge the best results on this task to date. Our method, **structural annealing**, is a general technique with broad applicability to hidden-structure discovery problems.

1 Introduction

Inducing a weighted context-free grammar from flat text is a hard problem. A common starting point for weighted grammar induction is the Expectation-Maximization (EM) algorithm (Dempster et al., 1977; Baker, 1979). EM's mediocre performance (Table 1) reflects two problems. First, it seeks to maximize likelihood, but a grammar that makes the training data likely does not necessarily assign a linguistically defensible syntactic structure. Second, the likelihood surface is not globally concave, and learners such as the EM algorithm can get trapped on local maxima (Charniak, 1993).

We seek here to capitalize on the intuition that, at least early in learning, the learner should search primarily for *string-local* structure, because most structure is local.[1] By penalizing dependencies between two words that are farther apart in the string, we obtain consistent improvements in accuracy of the learned model (§3).

We then explore how gradually *changing* δ over time affects learning (§4): we start out with a strong preference for short dependencies, then relax the preference. The new approach, **structural annealing**, often gives superior performance.

	model selection among values of λ and $\Theta^{(0)}$			
	worst	unsup.	sup.	oracle
German	19.8	19.8	54.4	54.4
English	21.8	41.6	41.6	42.0
Bulgarian	24.7	44.6	45.6	45.6
Mandarin	31.8	37.2	50.0	50.0
Turkish	32.1	41.2	48.0	51.4
Portuguese	35.4	37.4	42.3	43.0

Table 1: Baseline performance of EM-trained dependency parsing models: F_1 on non-$ attachments in test data, with various model selection conditions (3 initializers × 6 smoothing values). The languages are listed in decreasing order by the training set size. Experimental details can be found in the appendix.

An alternative structural bias is explored in §5. This approach views a sentence as a sequence of one or more yields of separate, independent trees. The points of segmentation are a hidden variable, and during learning all possible segmentations are entertained probabilistically. This allows the learner to accept hypotheses that explain the sentences as independent pieces.

In §6 we briefly review **contrastive estimation** (Smith and Eisner, 2005a), relating it to the new method, and show its performance alone and when augmented with structural bias.

2 Task and Model

In this paper we use a simple unlexicalized dependency model due to Klein and Manning (2004). The model is a probabilistic head automaton grammar (Alshawi, 1996) with a "split" form that renders it parseable in cubic time (Eisner, 1997).

Let $\mathbf{x} = \langle x_1, x_2, ..., x_n \rangle$ be the sentence. x_0 is a special "wall" symbol, $, on the left of every sentence. A tree $\mathbf{y}$ is defined by a pair of functions $\mathbf{y}_{\mathit{left}}$ and $\mathbf{y}_{\mathit{right}}$ (both $\{0, 1, 2, ..., n\} \rightarrow 2^{\{1,2,...,n\}}$) that map each word to its sets of left and right dependents, respectively. The graph is constrained to be a *projective* tree rooted at $: each word except $ has a single parent, and there are no cycles

[*] This work was supported by a Fannie and John Hertz Foundation fellowship to the first author and NSF ITR grant IIS-0313193 to the second author. The views expressed are not necessarily endorsed by the sponsors. We thank three anonymous COLING-ACL reviewers for comments.

[1] To be concrete, in the corpora tested here, 95% of dependency links cover ≤ 4 words (English, Bulgarian, Portuguese), ≤ 5 words (German, Turkish), ≤ 6 words (Mandarin).

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 569–576,
Sydney, July 2006. ©2006 Association for Computational Linguistics

or crossing dependencies.[2] $\mathbf{y}_{\mathit{left}}(0)$ is taken to be empty, and $\mathbf{y}_{\mathit{right}}(0)$ contains the sentence's single head. Let y^i denote the subtree rooted at position i. The probability $P(y^i \mid x_i)$ of generating this subtree, given its head word x_i, is defined recursively:

$$\prod_{D\in\{\mathit{left},\mathit{right}\}} p_{\text{stop}}(\text{stop} \mid x_i, D, [\mathbf{y}_D(i) = \emptyset]) \quad (1)$$
$$\times \prod_{j\in\mathbf{y}_D(i)} p_{\text{stop}}(\neg\text{stop} \mid x_i, D, \text{first}_{\mathbf{y}}(j))$$
$$\times p_{\text{child}}(x_j \mid x_i, D) \times P(y^j \mid x_j)$$

where $\text{first}_{\mathbf{y}}(j)$ is a predicate defined to be true iff x_j is the closest child (on either side) to its parent x_i. The probability of the entire tree is given by $p_\Theta(\mathbf{x},\mathbf{y}) = P(y^0 \mid \$)$. The parameters Θ are the conditional distributions p_{stop} and p_{child}.

Experimental baseline: EM. Following common practice, we always replace words by part-of-speech (POS) tags before training or testing. We used the EM algorithm to train this model on POS sequences in six languages. Complete experimental details are given in the appendix. Performance with unsupervised and supervised model selection across different λ values in add-λ smoothing and three initializers $\Theta^{(0)}$ is reported in Table 1. The supervised-selected model is in the 40–55% F_1-accuracy range on directed dependency attachments. (Here $F_1 \approx$ precision $\approx$ recall; see appendix.) Supervised model selection, which uses a small annotated development set, performs almost as well as the oracle, but unsupervised model selection, which selects the model that maximizes likelihood on an *unannotated* development set, is often much worse.

3 Locality Bias among Trees

Hidden-variable estimation algorithms—including EM—typically work by iteratively manipulating the model parameters Θ to improve an objective function $F(\Theta)$. EM explicitly alternates between the computation of a *posterior* distribution over hypotheses, $p_\Theta(\mathbf{y} \mid \mathbf{x})$ (where $\mathbf{y}$ is any tree with yield $\mathbf{x}$), and computing a new parameter estimate Θ.[3]

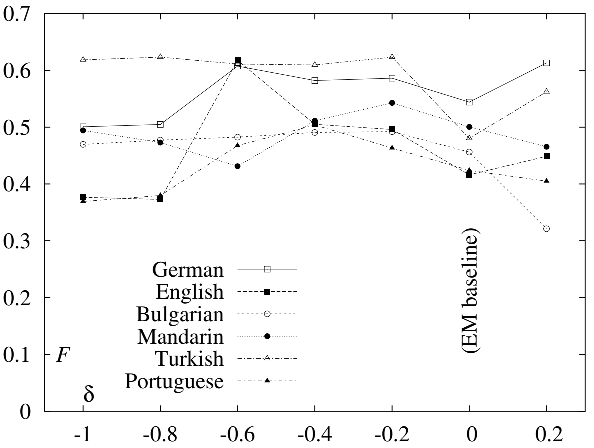


Figure 1: Test-set F_1 performance of models trained by EM with a **locality bias** at varying δ. Each curve corresponds to a different language and shows performance of supervised model selection *within* a given δ, across λ and $\Theta^{(0)}$ values. (See Table 3 for performance of models selected *across* δs.) We decode with $\delta = 0$, though we found that keeping the training-time value of δ would have had almost no effect. The EM baseline corresponds to $\delta = 0$.

One way to bias a learner toward local explanations is to penalize longer attachments. This was done for supervised parsing in different ways by Collins (1997), Klein and Manning (2003), and McDonald et al. (2005), all of whom considered intervening material or coarse distance classes when predicting children in a tree. Eisner and Smith (2005) achieved speed and accuracy improvements by modeling distance directly in a ML-estimated (deficient) generative model.

Here we use *string distance* to measure the length of a dependency link and consider the inclusion of a sum-of-lengths feature in the probabilistic model, for learning only. Keeping our original model, we will simply multiply into the probability of each tree another factor that penalizes long dependencies, giving:

$$p'_\Theta(\mathbf{x},\mathbf{y}) \propto p_\Theta(\mathbf{x},\mathbf{y}) \cdot e^{\left(\delta \sum_{i=1}^{n} \sum_{j\in\mathbf{y}(i)} |i-j|\right)} \quad (2)$$

where $\mathbf{y}(i) = \mathbf{y}_{\mathit{left}}(i) \cup \mathbf{y}_{\mathit{right}}(i)$. Note that if $\delta = 0$, we have the original model. As $\delta \to -\infty$, the new model p'_Θ will favor parses with shorter dependencies. The dynamic programming algorithms remain the same as before, with the appropriate $e^{\delta|i-j|}$ factor multiplied in at each attachment between x_i and x_j. Note that when $\delta = 0$, $p'_\Theta \equiv p_\Theta$.

Experiment. We applied a locality bias to the same dependency model by setting δ to different

[2] A projective parser could achieve perfect accuracy on our English and Mandarin datasets, $>$ 99% on Bulgarian, Turkish, and Portuguese, and $>$ 98% on German.

[3] For weighted grammar-based models, the posterior does not need to be explicitly represented; instead expectations under p_Θ are used to compute updates to Θ.

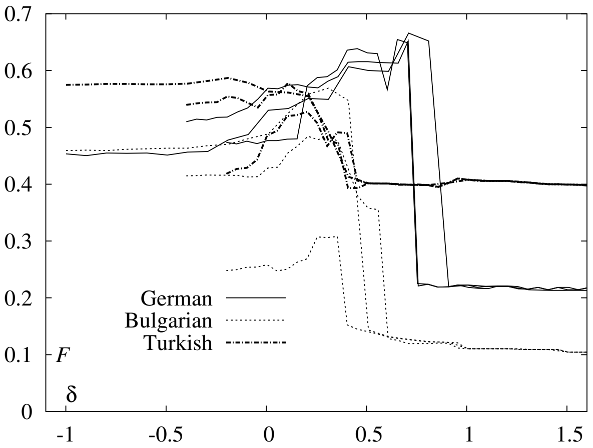


Figure 2: Test-set F_1 performance of models trained by EM with **structural annealing** on the distance weight δ. Here we show performance with add-10 smoothing, the all-zero initializer, for three languages with three different initial values δ_0. Time progresses from left to right. Note that it is generally best to start at $\delta_0 \ll 0$; note also the importance of picking the right point on the curve to stop. See Table 3 for performance of models selected across smoothing, initialization, starting, and stopping choices, in all six languages.

values in $[-1, 0.2]$ (see Eq. 2). The same initializers $\Theta^{(0)}$ and smoothing conditions were tested. Performance of supervised model selection among models trained at different δ values is plotted in Fig. 1. When a model is selected across *all* conditions (3 initializers × 6 smoothing values × 7 δs) using annotated development data, performance is notably better than the EM baseline using the same selection procedure (see Table 3, second column).

4 Structural Annealing

The central idea of this paper is to gradually *change* (anneal) the bias δ. Early in learning, local dependencies are emphasized by setting $\delta \ll 0$. Then δ is iteratively increased and training repeated, using the last learned model to initialize.

This idea bears a strong similarity to **deterministic annealing** (DA), a technique used in clustering and classification to smooth out objective functions that are piecewise constant (hence discontinuous) or bumpy (non-concave) (Rose, 1998; Ueda and Nakano, 1998). In unsupervised learning, DA iteratively re-estimates parameters like EM, but begins by requiring that the entropy of the posterior $p_\Theta(\mathbf{y} \mid \mathbf{x})$ be maximal, then gradually relaxes this entropy constraint. Since entropy is concave in Θ, the initial task is easy (maximize a concave, continuous function). At each step the optimization task becomes more difficult, but the initializer is given by the previous step and, in practice, tends to be close to a good local maximum of the more difficult objective. By the last iteration the objective is the same as in EM, but the annealed search process has acted like a good initializer. This method was applied with some success to grammar induction models by Smith and Eisner (2004).

In this work, instead of imposing constraints on the entropy of the model, we manipulate bias toward local hypotheses. As δ increases, we penalize long dependencies less. We call this **structural annealing**, since we are varying the strength of a soft constraint (bias) on structural hypotheses. In structural annealing, the final objective would be the same as EM if our final δ, $\delta_f = 0$, but we found that annealing farther ($\delta_f > 0$) works much better.[4]

Experiment: Annealing δ. We experimented with annealing schedules for δ. We initialized at $\delta_0 \in \{-1, -0.4, -0.2\}$, and increased δ by 0.1 (in the first case) or 0.05 (in the others) up to $\delta_f = 3$. Models were trained to convergence at each δ-epoch. Model selection was applied over the same initialization and regularization conditions as before, δ_0, and also over the choice of δ_f, with stopping allowed at any stage along the δ trajectory.

Trajectories for three languages with three different δ_0 values are plotted in Fig. 2. Generally speaking, $\delta_0 \ll 0$ performs better. There is consistently an early increase in performance as δ increases, but the stopping δ_f matters tremendously. Selected annealed-δ models surpass EM in all six languages; see the third column of Table 3. Note that structural annealing does not always outperform fixed-δ training (English and Portuguese). This is because we only tested a few values of δ_0, since annealing requires longer runtime.

5 Structural Bias via Segmentation

A related way to focus on local structure early in learning is to *broaden* the set of hypotheses to include *partial* parse structures. If $\mathbf{x} = \langle x_1, x_2, ..., x_n \rangle$, the standard approach assumes that $\mathbf{x}$ corresponds to the vertices of a single dependency tree. Instead, we entertain every hypothesis in which $\mathbf{x}$ is a *sequence* of yields from *separate*, independently-generated trees. For example, $\langle x_1, x_2, x_3 \rangle$ is the yield of one tree, $\langle x_4, x_5 \rangle$ is the

[4] The reader may note that $\delta_f > 0$ actually corresponds to a bias toward *longer* attachments. A more apt description in the context of annealing is to say that during early stages the learner starts liking local attachments too much, and we need to exaggerate δ to "coax" it to new hypotheses. See Fig. 2.

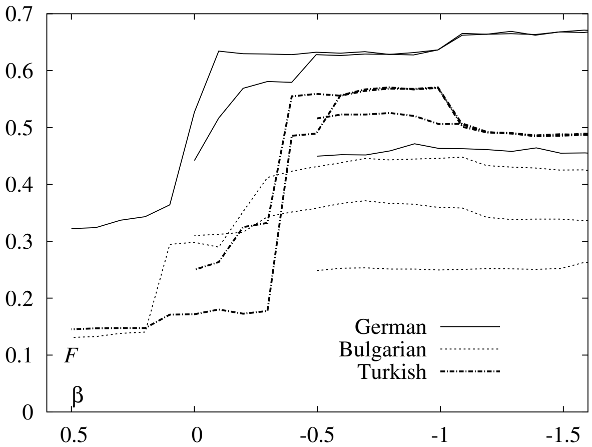


Figure 3: Test-set F_1 performance of models trained by EM with **structural annealing** on the breakage weight β. Here we show performance with add-10 smoothing, the all-zero initializer, for three languages with three different initial values β_0. Time progresses from left (large β) to right. See Table 3 for performance of models selected across smoothing, initialization, and stopping choices, in all six languages.

yield of a second, and $\langle x_6, ..., x_n\rangle$ is the yield of a third. One extreme hypothesis is that $\mathbf{x}$ is n single-node trees. At the other end of the spectrum is the original set of hypotheses—full trees on $\mathbf{x}$. Each has a nonzero probability.

Segmented analyses are intermediate representations that may be helpful for a learner to use to formulate notions of probable local structure, without committing to full trees.[5] We only *allow* unobserved breaks, never positing a hard segmentation of the training sentences. Over time, we increase the bias against broken structures, forcing the learner to commit most of its probability mass to full trees.

5.1 Vine Parsing

At first glance broadening the hypothesis space to entertain all 2^{n-1} possible segmentations may seem expensive. In fact the dynamic programming computation is almost the same as summing or maximizing over connected dependency trees. For the latter, we use an inside-outside algorithm that computes a score for every parse tree by computing the scores of *items*, or partial structures, through a bottom-up process. Smaller items are built first, then assembled using a set of rules defining how larger items can be built.[6]

Now note that any *sequence* of partial trees over $\mathbf{x}$ can be constructed by combining the same items into trees. The only difference is that we are willing to consider unassembled sequences of these partial trees as hypotheses, in addition to the fully connected trees. One way to accomplish this in terms of $\mathbf{y}_{right}(0)$ is to say that the root, \$, is allowed to have multiple children, instead of just one. Here, these children are independent of each other (e.g., generated by a unigram Markov model). In supervised dependency parsing, Eisner and Smith (2005) showed that imposing a hard constraint on the whole structure—specifically that each non-\$ dependency arc cross fewer than k words—can give guaranteed $O(nk^2)$ runtime with little to no loss in accuracy (for simple models). This constraint could lead to highly contrived parse trees, or none at all, for some sentences—both are avoided by the allowance of segmentation into a sequence of trees (each attached to \$). The construction of the “vine” (sequence of \$’s children) takes only $O(n)$ time once the chart has been assembled.

Our broadened hypothesis model is a probabilistic vine grammar with a unigram model over \$’s children. We allow (but do not require) segmentation of sentences, where each independent child of \$ is the root of one of the segments. We do not impose any constraints on dependency length.

5.2 Modeling Segmentation

Now the total probability of an n-length sentence $\mathbf{x}$, marginalizing over its hidden structures, sums up not only over trees, but over segmentations of $\mathbf{x}$. For completeness, we must include a probability model over the number of trees generated, which could be anywhere from 1 to n. The model over the number T of trees given a sentence of length n will take the following log-linear form:

$$P(T = t \mid n) = e^{t\beta} \Bigg/ \sum_{i=1}^{n} e^{i\beta}$$

where $\beta \in \mathbb{R}$ is the sole parameter. When $\beta = 0$, every value of T is equally likely. For $\beta \ll 0$, the model prefers larger structures with few breaks. At the limit ($\beta \to -\infty$), we achieve the standard learning setting, where the model must explain $\mathbf{x}$ using a single tree. We start however at $\beta \gg 0$, where the model prefers smaller trees with more breaks, in the limit preferring each word in $\mathbf{x}$ to be its own tree. We could describe “brokenness” as a feature in the model whose weight, β, is chosen extrinsically (and time-dependently), rather than empirically—just as was done with δ.

[5] See also work on partial parsing as a task in its own right: Hindle (1990) *inter alia*.

[6] See Eisner and Satta (1999) for the relevant algorithm used in the experiments.

		model selection among values of σ^2 and $\Theta^{(0)}$			
		worst	unsup.	sup.	oracle
Ger.	DORT1	32.5	**59.3**	**63.4**	63.4
	LENGTH	30.5	**56.4**	**57.3**	57.8
Eng.	DORT1	20.9	**56.6**	**57.4**	57.4
	LENGTH	29.1	37.2	**46.2**	46.2
Bul.	DORT1	19.4	26.0	40.5	43.1
	LENGTH	25.1	35.3	38.3	38.3
Man.	DORT1	9.4	24.2	41.1	41.1
	LENGTH	13.7	17.9	26.2	26.2
Tur.	DORT1	7.3	38.6	**58.2**	58.2
	LENGTH	21.5	34.1	**55.5**	55.5
Por.	DORT1	35.0	**59.8**	**71.8**	71.8
	LENGTH	30.8	33.6	33.6	33.6

Table 2: Performance of CE on test data, for different neighborhoods and with different levels of regularization. Boldface marks scores better than EM-trained models selected the same way (Table 1). The score is the F_1 measure on non-\$ attachments.

Annealing β resembles the popular **bootstrapping** technique (Yarowsky, 1995), which starts out aiming for high precision, and gradually improves coverage over time. With strong bias ($\beta \gg 0$), we seek a model that maintains high dependency precision on (non-\$) attachments by attaching most tags to \$. Over time, as this is iteratively weakened ($\beta \rightarrow -\infty$), we hope to improve coverage (dependency recall). Bootstrapping was applied to syntax learning by Steedman et al. (2003). Our approach differs in being able to remain partly agnostic about each tag's true parent (e.g., by giving 50% probability to attaching to \$), whereas Steedman et al. make a hard decision to retrain on a whole *sentence* fully or leave it out fully. In earlier work, Brill and Marcus (1992) adopted a "local first" iterative merge strategy for discovering phrase structure.

Experiment: Annealing β. We experimented with different annealing schedules for β. The initial value of β, β_0, was one of $\{-\frac{1}{2}, 0, \frac{1}{2}\}$. After EM training, β was diminished by $\frac{1}{10}$; this was repeated down to a value of $\beta_f = -3$. Performance after training at each β value is shown in Fig. 3.[7] We see that, typically, there is a sharp increase in performance somewhere during training, which typically lessens as $\beta \rightarrow -\infty$. Starting β too high can also damage performance. This method, then, is not robust to the choice of λ, β_0, or β_f, nor does it always do as well as annealing δ, although considerable gains are possible; see the fifth column of Table 3.

By testing models trained with a *fixed* value of β (for values in $[-1, 1]$), we ascertained that the performance improvement is due largely to annealing, not just the injection of segmentation bias (fourth vs. fifth column of Table 3).[8]

6 Comparison and Combination with Contrastive Estimation

Contrastive estimation (CE) was recently introduced (Smith and Eisner, 2005a) as a class of alternatives to the likelihood objective function locally maximized by EM. CE was found to outperform EM on the task of focus in this paper, when applied to English data (Smith and Eisner, 2005b). Here we review the method briefly, show how it performs across languages, and demonstrate that it can be combined effectively with structural bias.

Contrastive training defines for each example $\mathbf{x}_i$ a class of presumably poor, but similar, instances called the "neighborhood," $\mathcal{N}(\mathbf{x}_i)$, and seeks to maximize

$$\begin{aligned} C_{\mathcal{N}}(\Theta) &= \sum_i \log p_\Theta(\mathbf{x}_i \mid \mathcal{N}(\mathbf{x}_i)) \\ &= \sum_i \log \frac{\sum_{\mathbf{y}} p_\Theta(\mathbf{x}_i, \mathbf{y})}{\sum_{\mathbf{x}' \in \mathcal{N}(\mathbf{x}_i)} \sum_{\mathbf{y}} p_\Theta(\mathbf{x}', \mathbf{y})} \end{aligned}$$

At this point we switch to a log-linear (rather than stochastic) parameterization of the same weighted grammar, for ease of numerical optimization. All this means is that Θ (specifically, p_{stop} and p_{child} in Eq. 1) is now a set of nonnegative weights rather than probabilities.

Neighborhoods that can be expressed as finite-state lattices built from $\mathbf{x}_i$ were shown to give significant improvements in dependency parser quality over EM. Performance of CE using two of those neighborhoods on the current model and datasets is shown in Table 2.[9] 0-mean diagonal Gaussian smoothing was applied, with different variances, and model selection was applied over smoothing conditions and the same initializers as

[7] Performance measures are given using a *full* parser that finds the single best parse of the sentence with the learned parsing parameters. Had we decoded with a *vine* parser, we would see a precision$\searrow$, recall$\nearrow$ curve as β decreased.

[8] In principle, segmentation can be combined with the locality bias in §3 (δ). In practice, we found that this usually under-performed the EM baseline.

[9] We experimented with DELETE1, TRANSPOSE1, DELETEORTRANSPOSE1, and LENGTH. To conserve space we show only the latter two, which tend to perform best.

	EM	fixed δ		annealed δ		fixed β		annealed β		CE		fixed δ + CE	
			δ		$\delta_0 \to \delta_f$		β		$\beta_0 \to \beta_f$		$\mathcal{N}$		$\mathcal{N}, \delta$
German	54.4	61.3	0.2	**70.0**	-0.4 → 0.4	66.2	0.4	**68.9**	0.5 → -2.4	63.4	DOrT1	63.8	DOrT1, -0.2
English	41.6	**61.8**	-0.6	53.8	-0.4 → 0.3	55.6	0.2	58.4	0.5 → 0.0	57.4	DOrT1	**63.5**	DOrT1, -0.4
Bulgarian	45.6	49.2	-0.2	**58.3**	-0.4 → 0.2	47.3	-0.2	56.5	0 → -1.7	40.5	DOrT1	–	
Mandarin	50.0	51.1	-0.4	**58.0**	-1.0 → 0.2	38.0	0.2	**57.2**	0.5 → -1.4	43.4	Del1	–	
Turkish	48.0	**62.3**	-0.2	**62.4**	-0.2 → -0.15	53.6	-0.2	59.4	0.5 → -0.7	58.2	DOrT1	61.8	DOrT1, -0.6
Portuguese	42.3	50.4	-0.4	50.2	-0.4 → -0.1	51.5	0.2	62.7	0.5 → -0.5	**71.8**	DOrT1	**72.6**	DOrT1, -0.2

Table 3: Summary comparing models trained in a variety of ways with some relevant hyperparameters. Supervised model selection was applied in all cases, including EM (see the appendix). Boldface marks the best performance overall and trials that this performance did not significantly surpass under a sign test (i.e., $p \not< 0.05$). The score is the F_1 measure on non-\$ attachments. The fixed δ + CE condition was tested only for languages where CE improved over EM.

before. Four of the languages have at least one effective CE condition, supporting our previous English results (Smith and Eisner, 2005b), but CE was harmful for Bulgarian and Mandarin. Perhaps better neighborhoods exist for these languages, or there is some ideal neighborhood that would perform well for all languages.

Our approach of allowing broken trees (§5) is a natural extension of the CE framework. Contrastive estimation views learning as a process of moving posterior probability mass *from* (implicit) negative examples *to* (explicit) positive examples. The positive evidence, as in MLE, is taken to be the observed data. As originally proposed, CE allowed a redefinition of the implicit negative evidence from "all other sentences" (as in MLE) to "sentences like $\mathbf{x}_i$, but perturbed." Allowing segmentation of the training sentences redefines the positive *and* negative evidence. Rather than moving probability mass only to full analyses of the training example $\mathbf{x}_i$, we also allow probability mass to go to partial analyses of $\mathbf{x}_i$.

By injecting a bias ($\delta \neq 0$ or $\beta > -\infty$) among tree hypotheses, however, we have gone beyond the CE framework. We have added features to the tree model (dependency length-sum, number of breaks), whose weights we extrinsically manipulate over time to impose locality bias $C_{\mathcal{N}}$ and improve search on $C_{\mathcal{N}}$. Another idea, not explored here, is to change the contents of the neighborhood $\mathcal{N}$ over time.

Experiment: Locality Bias within CE. We combined CE with a fixed-δ locality bias for neighborhoods that were successful in the earlier CE experiment, namely DeleteOrTranspose1 for German, English, Turkish, and Portuguese. Our results, shown in the seventh column of Table 3, show that, in all cases except Turkish, the combination improves over either technique on its own. We leave exploration of structural annealing with CE to future work.

Experiment: Segmentation Bias within CE. For (language, $\mathcal{N}$) pairs where CE was effective, we trained models using CE with a fixed-β segmentation model. Across conditions ($\beta \in [-1, 1]$), these models performed very badly, hypothesizing extremely local parse trees: typically over 90% of dependencies were length 1 and pointed in the same direction, compared with the 60–70% length-1 rate seen in gold standards. To understand why, consider that the CE goal is to maximize the score of a sentence *and* all its segmentations while minimizing the scores of neighborhood sentences and their segmentations. An n-gram model can accomplish this, since the same n-grams are present in all segmentations of $\mathbf{x}$, and (some) different n-grams appear in $\mathcal{N}(\mathbf{x})$ (for Length and DeleteOrTranspose1). A bigram-like model that favors monotone branching, then, is not a bad choice for a CE learner that must account for segmentations of $\mathbf{x}$ and $\mathcal{N}(\mathbf{x})$.

Why doesn't CE *without* segmentation resort to n-gram-like models? Inspection of models trained using the standard CE method (no segmentation) with transposition-based neighborhoods Transpose1 and DeleteOrTranspose1 *did* have high rates of length-1 dependencies, while the poorly-performing Delete1 models found *low* length-1 rates. This suggests that a bias toward locality ("n-gram-ness") is built into the former neighborhoods, and may partly explain why CE works when it does. We achieved a similar locality bias in the likelihood framework when we broadened the hypothesis space, but doing so under CE *over*-focuses the model on local structures.

7 Error Analysis

We compared errors made by the selected EM condition with the best overall condition, for each language. We found that the number of corrected attachments always outnumbered the number of new errors by a factor of two or more.

Further, the new models are not getting better by merely reversing the *direction* of links made by EM; undirected accuracy also improved significantly under a sign test ($p < 10^{-6}$), across all six languages. While the most common corrections were to nouns, these account for only 25–41% of corrections, indicating that corrections are not "all of the same kind."

Finally, since more than half of corrections in every language involved reattachment to a noun or a verb (content word), we believe the improved models to be getting closer than EM to the deeper semantic relations between words that, ideally, syntactic models should uncover.

8 Future Work

One weakness of all recent weighted grammar induction work—including Klein and Manning (2004), Smith and Eisner (2005b), and the present paper—is a sensitivity to hyperparameters, including smoothing values, choice of $\mathcal{N}$ (for CE), and annealing schedules—not to mention initialization. This is quite observable in the results we have presented. An obstacle for unsupervised learning in general is the need for automatic, efficient methods for model selection. For annealing, inspiration may be drawn from continuation methods; see, e.g., Elidan and Friedman (2005). Ideally one would like to select values simultaneously for many hyperparameters, perhaps using a small annotated corpus (as done here), extrinsic figures of merit on successful learning trajectories, or plausibility criteria (Eisner and Karakos, 2005).

Grammar induction serves as a tidy example for structural annealing. In future work, we envision that other kinds of structural bias and annealing will be useful in other difficult learning problems where hidden structure is required, including machine translation, where the structure can consist of word correspondences or phrasal or recursive syntax with correspondences. The technique bears some similarity to the estimation methods described by Brown et al. (1993), which started by estimating simple models, using each model to seed the next.

9 Conclusion

We have presented a new unsupervised parameter estimation method, structural annealing, for learning hidden structure that biases toward simplicity and gradually weakens (anneals) the bias over time. We applied the technique to weighted dependency grammar induction and achieved a significant gain in accuracy over EM and CE, raising the state-of-the-art across six languages from 42–54% to 58–73% accuracy.

References

S. Afonso, E. Bick, R. Haber, and D. Santos. 2002. Floresta sintá(c)tica: a treebank for Portuguese. In *Proc. of LREC*.

H. Alshawi. 1996. Head automata and bilingual tiling: Translation with minimal representations. In *Proc. of ACL*.

N. B. Atalay, K. Oflazer, and B. Say. 2003. The annotation process in the Turkish treebank. In *Proc. of LINC*.

J. K. Baker. 1979. Trainable grammars for speech recognition. In *Proc. of the Acoustical Society of America*.

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. 2002. The TIGER Treebank. In *Proc. of Workshop on Treebanks and Linguistic Theories*.

E. Brill and M. Marcus. 1992. Automatically acquiring phrase structure using distributional analysis. In *Proc. of DARPA Workshop on Speech and Natural Language*.

P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

S. Buchholz and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

E. Charniak. 1993. *Statistical Language Learning*. MIT Press.

M. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of ACL*.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.

J. Eisner and D. Karakos. 2005. Bootstrapping without the boot. In *Proc. of HLT-EMNLP*.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proc. of ACL*.

J. Eisner and N. A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *Proc. of IWPT*.

J. Eisner. 1997. Bilexical grammars and a cubic-time probabilistic parser. In *Proc. of IWPT*.

G. Elidan and N. Friedman. 2005. Learning hidden variable networks: the information bottleneck approach. *Journal of Machine Learning Research*, 6:81–127.

D. Hindle. 1990. Noun classification from predicate-argument structure. In *Proc. of ACL*.

D. Klein and C. D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proc. of ACL*.

D. Klein and C. D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *NIPS 15*.

D. Klein and C. D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of ACL*.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330.

R. McDonald, K. Crammer, and F. Pereira. 2005. Online large-margin training of dependency parsers. In *Proc. of ACL*.

K. Oflazer, B. Say, D. Z. Hakkani-Tür, and G. Tür. 2003. Building a Turkish treebank. In A. Abeille, editor, *Building and Exploiting Syntactically-Annotated Corpora*. Kluwer.

K. Rose. 1998. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proc. of the IEEE*, 86(11):2210–2239.

K. Simov and P. Osenova. 2003. Practical annotation scheme for an HPSG treebank of Bulgarian. In *Proc. of LINC*.

K. Simov, G. Popova, and P. Osenova. 2002. HPSG-based syntactic treebank of Bulgarian (BulTreeBank). In A. Wilson, P. Rayson, and T. McEnery, editors, *A Rainbow of Corpora: Corpus Linguistics and the Languages of the World*, pages 135–42. Lincom-Europa.

K. Simov, P. Osenova, A. Simov, and M. Kouylekov. 2004. Design and implementation of the Bulgarian HPSG-based Treebank. *Journal of Research on Language and Computation*, 2(4):495–522.

N. A. Smith and J. Eisner. 2004. Annealing techniques for unsupervised statistical language learning. In *Proc. of ACL*.

N. A. Smith and J. Eisner. 2005a. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL*.

N. A. Smith and J. Eisner. 2005b. Guiding unsupervised grammar induction using contrastive estimation. In *Proc. of IJCAI Workshop on Grammatical Inference Applications*.

M. Steedman, M. Osborne, A. Sarkar, S. Clark, R. Hwa, J. Hockenmaier, P. Ruhlen, S. Baker, and J. Crim. 2003. Bootstrapping statistical parsers from small datasets. In *Proc. of EACL*.

N. Ueda and R. Nakano. 1998. Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–282.

N. Xue, F. Xia, F.-D. Chiou, and M. Palmer. 2004. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 10(4):1–30.

D. Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*.

A Experimental Setup

Following the usual conventions (Klein and Manning, 2002), our experiments use treebank POS sequences of length ≤ 10, stripped of words and punctuation. For smoothing, we apply add-λ, with six values of λ (in CE trials, we use a 0-mean diagonal Gaussian prior with five different values of σ^2). Our training datasets are:

- 8,227 **German** sentences from the TIGER Treebank (Brants et al., 2002),
- 5,301 **English** sentences from the WSJ Penn Treebank (Marcus et al., 1993),
- 4,929 **Bulgarian** sentences from the BulTreeBank (Simov et al., 2002; Simov and Osenova, 2003; Simov et al., 2004),
- 2,775 **Mandarin** sentences from the Penn Chinese Treebank (Xue et al., 2004),
- 2,576 **Turkish** sentences from the METU-Sabanci Treebank (Atalay et al., 2003; Oflazer et al., 2003), and
- 1,676 **Portuguese** sentences from the Bosque portion of the Floresta Sintá(c)tica Treebank (Afonso et al., 2002).

The Bulgarian, Turkish, and Portuguese datasets come from the CoNLL-X shared task (Buchholz and Marsi, 2006); we thank the organizers.

When comparing a hypothesized tree $\mathbf{y}$ to a gold standard $\mathbf{y}^*$, precision and recall measures are available. If every tree in the gold standard and every hypothesis tree is such that $|\mathbf{y}_{right}(0)| = 1$, then precision = recall = F_1, since $|\mathbf{y}| = |\mathbf{y}^*|$. $|\mathbf{y}_{right}(0)| = 1$ for all hypothesized trees in this paper, but not all treebank trees; hence we report the F_1 measure. The test set consists of around 500 sentences (in each language).

Iterative training proceeds until either 100 iterations have passed, or the objective converges within a relative tolerance of $\epsilon = 10^{-5}$, whichever occurs first.

Models trained at different hyperparameter settings and with different initializers are selected using a 500-sentence development set. *Unsupervised* model selection means the model with the highest training objective value on the development set was chosen. *Supervised* model selection chooses the model that performs best on the annotated development set. (*Oracle* and *worst* model selection are chosen based on performance on the test data.)

We use three initialization methods. We run a single special E step (to get expected counts of model events) then a single M step that renormalizes to get a probabilistic model $\Theta^{(0)}$. In initializer 1, the E step scores each tree as follows (only connected trees are scored):

$$u(\mathbf{x}, \mathbf{y}_{left}, \mathbf{y}_{right}) = \prod_{i=1}^{n} \prod_{j \in \mathbf{y}(i)} \left(1 + \frac{1}{|i-j|}\right)$$

(Proper) expectations under these scores are computed using an inside-outside algorithm. Initializer 2 computes expected counts directly, without dynamic programming. For an n-length sentence, $p(\mathbf{y}_{right}(0) = \{i\}) = \frac{1}{n}$ and $p(j \in \mathbf{y}(i)) \propto \frac{1}{|i-j|}$. These are scaled by an appropriate constant for each sentence, then summed across sentences to compute expected event counts. Initializer 3 assumes a uniform distribution over hidden structures in the special E step by setting all log probabilities to zero.

Maximum Entropy Based Restoration of Arabic Diacritics

Imed Zitouni, Jeffrey S. Sorensen, Ruhi Sarikaya
IBM T.J. Watson Research Center
1101 Kitchawan Rd, Yorktown Heights, NY 10598
{izitouni, sorenj, sarikaya}@us.ibm.com

Abstract

Short vowels and other diacritics are not part of written Arabic scripts. Exceptions are made for important political and religious texts and in scripts for beginning students of Arabic. Script without diacritics have considerable ambiguity because many words with different diacritic patterns appear identical in a diacritic-less setting. We propose in this paper a maximum entropy approach for restoring diacritics in a document. The approach can easily integrate and make effective use of diverse types of information; the model we propose integrates a wide array of lexical, segment-based and *part-of-speech* tag features. The combination of these feature types leads to a state-of-the-art diacritization model. Using a publicly available corpus (LDC's Arabic Treebank Part 3), we achieve a diacritic error rate of 5.1%, a segment error rate 8.5%, and a word error rate of 17.3%. In case-ending-less setting, we obtain a diacritic error rate of 2.2%, a segment error rate 4.0%, and a word error rate of 7.2%.

1 Introduction

Modern Arabic written texts are composed of scripts without short vowels and other diacritic marks. This often leads to considerable ambiguity since several words that have different diacritic patterns may appear identical in a diacritic-less setting. Educated modern Arabic speakers are able to accurately restore diacritics in a document. This is based on the context and their knowledge of the grammar and the lexicon of Arabic. However, a text without diacritics becomes a source of confusion for beginning readers and people with learning disabilities. A text without diacritics is also problematic for applications such as text-to-speech or speech-to-text, where the lack of diacritics adds another layer of ambiguity when processing the data. As an example, full vocalization of text is required for text-to-speech applications, where the mapping from graphemes to phonemes is simple compared to languages such as English and French; where there is, in most cases, one-to-one relationship. Also, using data with diacritics shows an improvement in the accuracy of speech-recognition applications (Afify et al., 2004). Currently, text-to-speech, speech-to-text, and other applications use data where diacritics are placed manually, which is a tedious and time consuming excercise. A diacritization system that restores the diacritics of scripts, i.e. supply the full diacritical markings, would be of interest to these applications. It also would greatly benefit nonnative speakers, sufferers of dyslexia and could assist in restoring diacritics of children's and poetry books, a task that is currently done manually.

We propose in this paper a statistical approach that restores diacritics in a text document. The proposed approach is based on the maximum entropy framework where several diverse sources of information are employed. The model implicitly learns the correlation between these types of information and the output diacritics.

In the next section, we present the set of diacritics to be restored and the ambiguity we face when processing a non-diacritized text. Section 3 gives a brief summary of previous related works. Section 4 presents our diacritization model; we explain the training and decoding process as well as the different feature categories employed to restore the diacritics. Section 5 describes a clearly defined and replicable split of the LDC's Arabic Treebank Part 3 corpus, used to built and evaluate the system, so that the reproduction of the results and future comparison can accurately be established. Section 6 presents the experimental results. Section 7 reports a comparison of our approach to the finite state machine modeling technique that showed promissing results in (Nelken and Shieber, 2005). Finally, section 8 concludes the paper and discusses future directions.

2 Arabic Diacritics

The Arabic alphabet consists of 28 letters that can be extended to a set of 90 by additional shapes, marks, and vowels (Tayli and Al-Salamah, 1990). The 28 letters represent the consonants and long

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 577–584,
Sydney, July 2006.

vowels such as ـَا, ـَى (both pronounced as /a:/), ـي (pronounced as /i:/), and ـو (pronounced as /u:/). Long vowels are constructed by combining ـَا, ـَى, ـي, and ـو with the short vowels. The short vowels and certain other phonetic information such as consonant doubling (shadda) are not represented by letters, but by diacritics. A diacritic is a short stroke placed above or below the consonant. Table 1 shows the complete set of Arabic diacritics. We split the Arabic diacritics into three sets: short vowels, doubled case endings, and syllabification marks. Short vowels are written as symbols either above or below the letter in text with diacritics, and dropped all together in text without diacritics. We find three short vowels:

Diacritic on ة	**Name**	**Meaning/ Pronunciation**
Short vowels		
ةَ	fatha	/a/
ةُ	damma	/u/
ةِ	kasra	/i/
Doubled case ending ("tanween")		
ةً	tanween al-fatha	/an/
ةٌ	tanween al-damma	/un/
ةٍ	tanween al-kasra	/in/
Syllabification marks		
ةّ	shadda	consonant doubling
ةْ	sukuun	vowel absence

Table 1: Arabic diacritics on the letter – consonant – ة (pronounced as */t/*).

- fatha: it represents the /a/ sound and is an oblique dash over a consonant as in ةَ (c.f. fourth row of Table 1).
- damma: it represents the /u/ sound and is a loop over a consonant that resembles the shape of a comma (c.f. fifth row of Table 1).
- kasra: it represents the /i/ sound and is an oblique dash under a consonant (c.f. sixth row of Table 1).

The doubled case ending diacritics are vowels used at the end of the words to mark case distinction, which can be considered as a double short vowels; the term "tanween" is used to express this phenomenon. Similar to short vowels, there are three different diacritics for tanween: tanween al-fatha, tanween al-damma, and tanween al-kasra. They are placed on the last letter of the word and have the phonetic effect of placing an "N" at the end of the word. Text with diacritics contains also two syllabification marks:

- *shadda*: it is a gemination mark placed above the Arabic letters as in ةّ. It denotes the doubling of the consonant. The shadda is usually combined with a short vowel such as in ةَّ.
- *sukuun*: written as a small circle as in ةْ. It is used to indicate that the letter doesn't contain vowels.

Figure 1 shows an Arabic sentence transcribed with and without diacritics. In modern Arabic, writing scripts without diacritics is the most natural way. Because many words with different vowel patterns may appear identical in a diacritic-less setting, considerable ambiguity exists at the word level. The word كتب, for example, has 21 possible forms that have valid interpretations when adding diacritics (Kirchhoff and Vergyri, 2005). It may have the interpretation of the verb "to write" in كَتَبَ (pronounced /kataba/). Also, it can be interpreted as "books" in the noun form كُتُبٌ(pronounced /kutubun/). A study made by (Debili et al., 2002) shows that there is an average of 11.6 possible diacritizations for every non-diacritized word when analyzing a text of 23,000 script forms.

كتب الرئِيس المذكرة.

كَتَبَ الرَّئِيسُ المُذَكَّرَةِ.

Figure 1: The same Arabic sentence without (upper row) and with (lower row) diacritics. The English translation is "*the president wrote the document.*"

Arabic diacritic restoration is a non-trivial task as expressed in (El-Imam, 2003). Native speakers of Arabic are able, in most cases, to accurately vocalize words in text based on their context, the speaker's knowledge of the grammar, and the lexicon of Arabic. Our goal is to convert knowledge used by native speakers into features and incorporate them into a maximum entropy model. We assume that the input text does not contain any diacritics.

3 Previous Work

Diacritic restoration has been receiving increasing attention and has been the focus of several studies. In (El-Sadany and Hashish, 1988), a rule based method that uses morphological analyzer for vowelization was proposed. Another, rule-based grapheme to sound conversion approach was appeared in 2003 by Y. El-Imam (El-Imam, 2003). The main drawbacks of these rule based methods is that it is difficult to maintain the rules up-to-date and extend them to other Arabic dialects. Also, new rules are required due to the changing nature of any "living" language.

More recently, there have been several new studies that use alternative approaches for the diacritization problem. In (Emam and Fisher, 2004) an example based hierarchical top-down approach is proposed. First, the training data is searched hierarchically for a matching sentence. If there is a matching sentence, the whole utterance is used. Otherwise they search for matching phrases, then words to restore diacritics. If there is no match at all, character n-gram models are used to diacritize each word in the utterance.

In (Vergyri and Kirchhoff, 2004), diacritics in conversational Arabic are restored by combining morphological and contextual information with an acoustic signal. Diacritization is treated as an unsupervised tagging problem where each word is tagged as one of the many possible forms provided by the Buckwalter's morphological analyzer (Buckwalter, 2002). The Expectation Maximization (EM) algorithm is used to learn the tag sequences.

Y. Gal in (Gal, 2002) used a HMM-based diacritization approach. This method is a white-space delimited word based approach that restores only vowels (a subset of all diacritics).

Most recently, a weighted finite state machine based algorithm is proposed (Nelken and Shieber, 2005). This method employs characters and larger morphological units in addition to words. Among all the previous studies this one is more sophisticated in terms of integrating multiple information sources and formulating the problem as a search task within a unified framework. This approach also shows competitive results in terms of accuracy when compared to previous studies. In their algorithm, a character based generative diacritization scheme is enabled only for words that do not occur in the training data. It is not clearly stated in the paper whether their method predict the diacritics shedda and sukuun.

Even though the methods proposed for diacritic restoration have been maturing and improving over time, they are still limited in terms of coverage and accuracy. In the approach we present in this paper, we propose to restore the most comprehensive list of the diacritics that are used in any Arabic text. Our method differs from the previous approaches in the way the diacritization problem is formulated and because multiple information sources are integrated. We view the diacritic restoration problem as *sequence classification*, where given a sequence of characters our goal is to assign diacritics to each character. Our appoach is based on Maximum Entropy (MaxEnt henceforth) technique (Berger et al., 1996). MaxEnt can be used for sequence classification, by converting the activation scores into probabilities (through the soft-max function, for instance) and using the standard dynamic programming search algorithm (also known as Viterbi search). We find in the literature several other approaches of sequence classification such as (McCallum et al., 2000) and (Lafferty et al., 2001). The *conditional random fields* method presented in (Lafferty et al., 2001) is essentially a MaxEnt model over the entire sequence: it differs from the Maxent in that it models the sequence information, whereas the Maxent makes a decision for each state independently of the other states. The approach presented in (McCallum et al., 2000) combines Maxent with Hidden Markov models to allow observations to be presented as arbitrary overlapping features, and define the probability of state sequences given observation sequences.

We report in section 7 a comparative study between our approach and the most competitive diacritic restoration method that uses finite state machine algorithm (Nelken and Shieber, 2005). The MaxEnt framework was successfully used to combine a diverse collection of information sources and yielded a highly competitive model that achieves a 5.1% DER.

4 Automatic Diacritization

The performance of many natural language processing tasks, such as shallow parsing (Zhang et al., 2002) and named entity recognition (Florian et al., 2004), has been shown to depend on integrating many sources of information. Given the stated focus of integrating many feature types, we selected the MaxEnt classifier. MaxEnt has the ability to integrate arbitrary types of information and make a classification decision by aggregating all information available for a given classification.

4.1 Maximum Entropy Classifiers

We formulate the task of restoring diacritics as a classification problem, where we assign to each character in the text a label (i.e., diacritic). Before formally describing the method[1], we introduce some notations: let $\mathcal{Y} = \{y_1, \ldots, y_n\}$ be the set of diacritics to predict or restore, $\mathcal{X}$ be the example space and $\mathcal{F} = \{0, 1\}^m$ be a feature space. Each example $x \in \mathcal{X}$ has associated a vector of binary features $f(x) = (f_1(x), \ldots, f_m(x))$. In a supervised framework, like the one we are considering here, we have access to a set of training examples together with their classifications: $\{(x_1, y_1), \ldots, (x_k, y_k)\}$.

[1] This is not meant to be an in-depth introduction to the method, but a brief overview to familiarize the reader with them.

The MaxEnt algorithm associates a set of weights $(\alpha_{ij})_{j=1\ldots m}^{i=1\ldots n}$ with the features, which are estimated during the training phase to maximize the likelihood of the data (Berger et al., 1996). Given these weights, the model computes the probability distribution over labels for a particular example x as follows:

$$P(y|x) = \frac{1}{Z(x)} \prod_{j=1}^{m} \alpha_{ij}^{f_j(x)},\; Z(x) = \sum_i \prod_j \alpha_{ij}^{f_j(x)}$$

where $Z(\mathcal{X})$ is a normalization factor. To estimate the optimal α_j values, we train our MaxEnt model using the *sequential conditional generalized iterative scaling* (SCGIS) technique (Goodman, 2002). While the MaxEnt method can nicely integrate multiple feature types seamlessly, in certain cases it is known to overestimate its confidence in especially low-frequency features. To overcome this problem, we use the regularization method based on adding *Gaussian priors* as described in (Chen and Rosenfeld, 2000). After computing the class probability distribution, the chosen diacritic is the one with the most *aposteriori* probability. The decoding algorithm, described in section 4.2, performs sequence classification, through dynamic programming.

4.2 Search to Restore Diacritics

We are interested in finding the diacritics of all characters in a script or a sentence. These diacritics have strong interdependencies which cannot be properly modeled if the classification is performed independently for each character. We view this problem as *sequence classification*, as contrasted with an *example-based classification* problem: given a sequence of characters in a sentence $x_1 x_2 \ldots x_L$, our goal is to assign diacritics (labels) to each character, resulting in a sequence of diacritics $y_1 y_2 \ldots y_L$. We make an assumption that diacritics can be modeled as a limited order Markov sequence: the diacritic associated with the character i depends only on the diacritics associated with the k previous diacritics, where k is usually equal to 3. Given this assumption, and the notation $x_1^L = x_1 \ldots x_L$, the conditional probability of assigning the diacritic sequence y_1^L to the character sequence x_1^L becomes

$$\begin{aligned} &p\left(y_1^L | x_1^L\right) = \\ &\quad p\left(y_1 | x_1^L\right) p\left(y_2 | x_1^L, y_1\right) \ldots p\left(y_L | x_1^L, y_{L-k+1}^{L-1}\right) \end{aligned} \quad (1)$$

and our goal is to find the sequence that maximizes this conditional probability

$$\hat{y}_1^L = \arg\max_{y_1^L} p\left(y_1^L | x_1^L\right) \quad (2)$$

While we restricted the conditioning on the classification tag sequence to the previous k diacritics, we do not impose any restrictions on the conditioning on the characters – the probability is computed using the entire character sequence x_1^L.

To obtain the sequence in Equation (2), we create a *classification tag lattice* (also called *trellis*), as follows:

- Let x_1^L be the input sequence of character and $S = \{s_1, s_2, \ldots, s_m\}$ be an enumeration of $\mathcal{Y}^k$ $(m = |\mathcal{Y}|^k)$ - we will call an element s_j a state. Every such state corresponds to the labeling of k successive characters. We find it useful to think of an element s_i as a vector with k elements. We use the notations $s_i[j]$ for j^{th} element of such a vector (the label associated with the token $x_{i-k+j+1}$) and $s_i[j_1 \ldots j_2]$ for the sequence of elements between indices j_1 and j_2.
- We conceptually associate every character $x_i, i = 1, \ldots, L$ with a copy of S, $S^i = \left\{s_1^i, \ldots, s_m^i\right\}$; this set represents all the possible labelings of characters x_{i-k+1}^i at the stage where x_i is examined.
- We then create links from the set S^i to the S^{i+1}, for all $i = 1 \ldots L-1$, with the property that

$$w\left(s_{j_1}^i, s_{j_2}^{i+1}\right) = \begin{cases} p\left(s_{j_1}^{i+1}[k] \,|x_1^L,\, s_{j_2}^{i+1}[1..k-1]\right) \\ \quad if\ s_{j_1}^i[2..k] = s_{j_2}^{i+1}[1..k-1] \\ 0 \qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

  These weights correspond to probability of a transition from the state $s_{j_1}^i$ to the state $s_{j_2}^{i+1}$.
- For every character x_i, we compute recursively[2]

$$\begin{aligned} \beta_0(s_j) &= 0, j = 1, \ldots, k \\ \beta_i(s_j) &= \max_{j_1=1,\ldots,M} \beta_{i-1}(s_{j_1}) + \log w\left(s_{j_1}^{i-1}, s_j^i\right) \\ \gamma_i(s_j) &= \\ &\arg\max_{j_1=1,\ldots,M} \beta_{i-1}(s_{j_1}) + \log w\left(s_{j_1}^{i-1}, s_j^i\right) \end{aligned}$$

  Intuitively, $\beta_i(s_j)$ represents the log-probability of the most probable path through the lattice that ends in state s_j after i steps, and $\gamma_i(s_j)$ represents the state just before s_j on that particular path.
- Having computed the $(\beta_i)_i$ values, the algorithm for finding the best path, which corresponds to the solution of Equation (2) is
  1. Identify $\hat{s}_L^L = \arg\max_{j=1\ldots L} \beta_L(s_j)$
  2. For $i = L-1 \ldots 1$, compute

$$\hat{s}_i^i = \gamma_{i+1}\left(\hat{s}_{i+1}^{i+1}\right)$$

[2] For convenience, the index i associated with state s_j^i is moved to β; the function $\beta_i(s_j)$ is in fact $\beta\left(s_j^i\right)$.

3. The solution for Equation (2) is given by

$$\hat{y} = \left\{\hat{s}_1^1[k], \hat{s}_2^2[k], \ldots, \hat{s}_L^L[k]\right\}$$

The runtime of the algorithm is $\Theta\left(|\mathcal{Y}|^k \cdot L\right)$, linear in the size of the sentence L but exponential in the size of the Markov dependency, k. To reduce the search space, we use *beam-search*.

4.3 Features Employed

Within the MaxEnt framework, any type of features can be used, enabling the system designer to experiment with interesting feature types, rather than worry about specific feature interactions. In contrast, with a rule based system, the system designer would have to consider how, for instance, lexical derived information for a particular example interacts with character context information. That is not to say, ultimately, that rule-based systems are in some way inferior to statistical models – they are built using valuable insight which is hard to obtain from a statistical-model-only approach. Instead, we are merely suggesting that the output of such a rule-based system can be easily integrated into the MaxEnt framework as one of the input features, most likely leading to improved performance.

Features employed in our system can be divided into three different categories: lexical, segment-based, and part-of-speech tag (POS) features. We also use the previously assigned two diacritics as additional features.

In the following, we briefly describe the different categories of features:

- **Lexical Features:** we include the character n-gram spanning the curent character x_i, both preceding and following it in a window of 7: $\{x_{i-3}, \ldots, x_{i+3}\}$. We use the current word w_i and its word context in a window of 5 (forward and backward trigram): $\{w_{i-2}, \ldots, w_{i+2}\}$. We specify if the character of analysis is at the beginning or at the end of a word. We also add joint features between the above source of information.

- **Segment-Based Features :** Arabic blank-delimited words are composed of zero or more prefixes, followed by a stem and zero or more suffixes. Each prefix, stem or suffix will be called a segment in this paper. Segments are often the subject of analysis when processing Arabic (Zitouni et al., 2005). Syntactic information such as POS or parse information is usually computed on segments rather than words. As an example, the Arabic white-space delimited word قَابلتهم contains a verb قَابل, a third-person feminine singular subject-marker ت (she), and a pronoun suffix هم (them); it is also a complete sentence meaning "*she met them*." To separate the Arabic white-space delimited words into segments, we use a segmentation model similar to the one presented by (Lee et al., 2003). The model obtains an accuracy of about 98%. In order to simulate real applications, we only use segments generated by the model rather than true segments. In the diacritization system, we include the current segment a_i and its word segment context in a window of 5 (forward and backward trigram): $\{a_{i-2}, \ldots, a_{i+2}\}$. We specify if the character of analysis is at the beginning or at the end of a segment. We also add joint information with lexical features.

- **POS Features :** we attach to the segment a_i of the current character, its POS: $POS(a_i)$. This is combined with joint features that include the lexical and segment-based information. We use a statistical POS tagging system built on Arabic Treebank data with MaxEnt framework (Ratnaparkhi, 1996). The model has an accuracy of about 96%. We did not want to use the true POS tags because we would not have access to such information in real applications.

5 Data

The diacritization system we present here is trained and evaluated on the LDC's Arabic Treebank of diacritized news stories – Part 3 v1.0: catalog number LDC2004T11 and ISBN 1-58563-298-8. The corpus includes complete vocalization (including case-endings). We introduce here a clearly defined and replicable split of the corpus, so that the reproduction of the results or future investigations can accurately and correctly be established. This corpus includes 600 documents from the An Nahar News Text. There are a total of 340,281 words. We split the corpus into two sets: training data and development test (devtest) data. The training data contains 288,000 words approximately, whereas the devtest contains close to 52,000 words. The 90 documents of the devtest data are created by taking the last (in chronological order) 15% of documents dating from "20021015_0101" (i.e., October 15, 2002) to "20021215_0045" (i.e., December 15, 2002). The time span of the devtest is intentionally non-overlapping with that of the training set, as this models how the system will perform in the real world.

Previously published papers use proprietary corpus or lack clear description of the training/devtest data split, which make the comparison to other techniques difficult. By clearly reporting the split of the publicly available LDC's Arabic Treebank corpus in this section, we want future comparisons to be correctly established.

6 Experiments

Experiments are reported in terms of word error rate (WER), segment error rate (SER), and diacritization error rate (DER). The DER is the proportion of incorrectly restored diacritics. The WER is the percentage of incorrectly diacritized white-space delimited words: in order to be counted as incorrect, at least one character in the word must have a diacritization error. The SER is similar to WER but indicates the proportion of incorrectly diacritized segments. A segment can be a prefix, a stem, or a suffix. Segments are often the subject of analysis when processing Arabic (Zitouni et al., 2005). Syntactic information such as POS or parse information is based on segments rather than words. Consequently, it is important to know the SER in cases where the diacritization system may be used to help disambiguate syntactic information.

Several modern Arabic scripts contains the consonant doubling "shadda"; it is common for native speakers to write without diacritics except the shadda. In this case the role of the diacritization system will be to restore the short vowels, doubled case ending, and the vowel absence "sukuun". We run two batches of experiments: a first experiment where documents contain the original shadda and a second one where documents don't contain any diacritics including the shadda. The diacritization system proceeds in two steps when it has to predict the shadda: a first step where only shadda is restored and a second step where other diacritics (excluding shadda) are predicted.

To assess the performance of the system under different conditions, we consider three cases based on the kind of features employed:

1. system that has access to lexical features only;
2. system that has access to lexical and segment-based features;
3. system that has access to lexical, segment-based and POS features.

The different system types described above use the two previously assigned diacritics as additional feature. The DER of the shadda restoration step is equal to 5% when we use lexical features only, 0.4% when we add segment-based information, and 0.3% when we employ lexical, POS, and segment-based features.

Table 2 reports experimental results of the diacritization system with different feature sets. Using only lexical features, we observe a DER of 8.2% and a WER of 25.1% which is competitive to a state-of-the-art system evaluated on Arabic Treebank Part 2: in (Nelken and Shieber, 2005) a DER of 12.79% and a WER of 23.61% are reported. The system they described in (Nelken and Shieber, 2005) uses lexical, segment-based, and morphological information. Table 2 also shows that, when segment-based information is added to our system, a significant improvement is achieved: 25% for WER (18.8 vs. 25.1), 38% for SER (9.4 vs. 13.0), and 41% for DER (5.8 vs. 8.2). Similar behavior is observed when the documents contain the original shadda. POS features are also helpful in improving the performance of the system. They improved the WER by 4% (18.0 vs. 18.8), SER by 5% (8.9 vs. 9.4), and DER by 5% (5.5 vs. 5.8).

True shadda			**Predicted shadda**		
WER	*SER*	*DER*	*WER*	*SER*	*DER*
Lexical features					
24.8	12.6	7.9	25.1	13.0	8.2
Lexical + segment-based features					
18.2	9.0	5.5	18.8	9.4	5.8
Lexical + segment-based + POS features					
17.3	8.5	5.1	18.0	8.9	5.5

Table 2: The impact of features on the diacritization system performance. The columns marked with "True shadda" represent results on documents containing the original consonant doubling "shadda" while columns marked with "Predicted shadda" represent results where the system restored all diacritics including shadda.

Case-ending in Arabic documents consists of the diacritic attributed to the last character in a white-space delimited word. Restoring them is the most difficult part in the diacritization of a document. Case endings are only present in formal or highly literary scripts. Only educated speakers of modern standard Arabic master their use. Technically, every noun has such an ending, although at the end of a sentence no inflection is pronounced, even in formal speech, because of the rules of 'pause'. For this reason, we conduct another experiment in which case-endings were stripped throughout the training and testing data without the attempt to restore them.

We present in Table 3 the performance of the diacritization system on documents without case-endings. Results clearly show that when case-endings are omitted, the WER declines by 58% (7.2% vs. 17.3%), SER is decreased by 52% (4.0% vs. 8.5%), and DER is reduced by 56% (2.2% vs. 5.1%). Also, Table 3 shows again that a richer set of features results in a better performance; compared to a system using lexical features only, adding POS and segment-based features improved the WER by 38% (7.2% vs. 11.8%), the SER by 39% (4.0% vs. 6.6%), and DER by 38% (2.2% vs.

True shadda			**Predicted shadda**		
WER	*SER*	*DER*	*WER*	*SER*	*DER*
Lexical features					
11.8	6.6	3.6	12.4	7.0	3.9
Lexical + segment-based features					
7.8	4.4	2.4	8.6	4.8	2.7
Lexical + segment-based + POS features					
7.2	4.0	2.2	7.9	4.4	2.5

Table 3: Performance of the diacritization system based on employed features. System is trained and evaluated on documents without case-ending. Columns marked with "True shadda" represent results on documents containing the original consonant doubling "shadda" while columns marked with "Predicted shadda" represent results where the system restored all diacritics including shadda.

3.6%). Similar to the results reported in Table 2, we show that the performance of the system are similar whether the document contains the original shadda or not. A system like this trained on non case-ending documents can be of interest to applications such as speech recognition, where the last state of a word HMM model can be defined to absorb all possible vowels (Afify et al., 2004).

7 Comparison to other approaches

As stated in section 3, the most recent and advanced approach to diacritic restoration is the one presented in (Nelken and Shieber, 2005): they showed a DER of 12.79% and a WER of 23.61% on Arabic Treebank corpus using *finite state transducers* (FST) with a Katz language modeling (LM) as described in (Chen and Goodman, 1999). Because they didn't describe how they split their corpus into training/test sets, we were not able to use the same data for comparison purpose.

In this section, we want essentially to duplicate the aforementioned FST result for comparison using the identical training and testing set we use for our experiments. We also propose some new variations on the finite state machine modeling technique which improve performance considerably.

The algorithm for FST based vowel restoration could not be simpler: between every pair of characters we insert diacritics if doing so improves the likelihood of the sequence as scored by a statistical n-gram model trained upon the training corpus. Thus, in between every pair of characters we propose and score all possible diacritical insertions. Results reported in Table 4 indicate the error rates of diacritic restoration (including shadda). We show performance using both Kneser-Ney and Katz LMs (Chen and Goodman, 1999) with increasingly large n-grams. It is our opinion that large n-grams effectively duplicate the use of a lexicon. It is unfortunate but true that, even for a rich resource like the Arabic Treebank, the choice of modeling heuristic and the effects of small sample size are considerable. Using the finite state machine modeling technique, we obtain similar results to those reported in (Nelken and Shieber, 2005): a WER of 23% and a DER of 15%. Better performance is reached with the use of Kneser-Ney LM.

These results still under-perform those obtained by MaxEnt approach presented in Table 2. When all sources of information are included, the MaxEnt technique outperforms the FST model by 21% (22% vs. 18%) in terms of WER and 39% (9% vs. 5.5%) in terms of DER.

The SER reported on Table 2 and Table 3 are based on the Arabic segmentation system we use in the MaxEnt approach. Since, the FST model doesn't use such a system, we found inappropriate to report SER in this section.

	Katz LM		Kneser-Ney LM	
n-gram size	WER	DER	WER	DER
3	63	31	55	28
4	54	25	38	19
5	51	21	28	13
6	44	18	24	11
7	39	16	23	11
8	37	15	23	10

Table 4: Error Rate in % for n-gram diacritic restoration using FST.

We propose in the following an extension to the aforementioned FST model, where we jointly determines not only diacritics but segmentation into affixes as described in (Lee et al., 2003). Table 5 gives the performance of the extended FST model where Kneser-Ney LM is used, since it produces better results. This should be a much more difficult task, as there are more than twice as many possible insertions. However, the choice of diacritics is related to and dependent upon the choice of segmentation. Thus, we demonstrate that a richer internal representation produces a more powerful model.

8 Conclusion

We presented in this paper a statistical model for Arabic diacritic restoration. The approach we propose is based on the Maximum entropy framework, which gives the system the ability to integrate different sources of knowledge. Our model has the advantage of successfully combining diverse sources of information ranging from lexical, segment-based and POS features. Both POS and segment-based features are generated by separate statistical systems – not extracted manually – in order to simulate real world applications. The segment-based features are extracted from a statistical morphological analysis system using WFST approach and the POS features are generated by a parsing model

	True Shadda		Predicted Shadda	
n-gram size	Kneser-Ney		Kneser-Ney	
	WER	DER	WER	DER
3	49	23	52	27
4	34	14	35	17
5	26	11	26	12
6	23	10	23	10
7	23	9	22	10
8	23	9	22	10

Table 5: Error Rate in % for n-gram diacritic restoration and segmentation using FST and Kneser-Ney LM. Columns marked with "True shadda" represent results on documents containing the original consonant doubling "shadda" while columns marked with "Predicted shadda" represent results where the system restored all diacritics including shadda.

that also uses Maximum entropy framework. Evaluation results show that combining these sources of information lead to state-of-the-art performance.

As future work, we plan to incorporate Buckwalter morphological analyzer information to extract new features that reduce the search space. One idea will be to reduce the search to the number of hypotheses, if any, proposed by the morphological analyzer. We also plan to investigate additional conjunction features to improve the accuracy of the model.

Acknowledgments

Grateful thanks are extended to Radu Florian for his constructive comments regarding the maximum entropy classifier.

References

M. Afify, S. Abdou, J. Makhoul, L. Nguyen, and B. Xiang. 2004. The BBN RT04 BN Arabic System. In *RT04 Workshop*, Palisades NY.

A. Berger, S. Della Pietra, and V. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

T. Buckwalter. 2002. Buckwalter Arabic morphological analyzer version 1.0. Technical report, Linguistic Data Consortium, LDC2002L49 and ISBN 1-58563-257-0.

Stanley F. Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. computer speech and language. *Computer Speech and Language*, 4(13):359–393.

Stanley Chen and Ronald Rosenfeld. 2000. A survey of smoothing techniques for me models. *IEEE Trans. on Speech and Audio Processing.*

F. Debili, H. Achour, and E. Souissi. 2002. De l'etiquetage grammatical a‘ la voyellation automatique de l'arabe. Technical report, Correspondances de l'Institut de Recherche sur le Maghreb Contemporain 17.

Y. El-Imam. 2003. Phonetization of arabic: rules and algorithms. *Computer Speech and Language*, 18:339–373.

T. El-Sadany and M. Hashish. 1988. Semi-automatic vowelization of Arabic verbs. In *10th NC Conference*, Jeddah, Saudi Arabia.

O. Emam and V. Fisher. 2004. A hierarchical approach for the statistical vowelization of Arabic text. Technical report, IBM patent filed, DE9-2004-0006, US patent application US2005/0192809 A1.

R. Florian, H. Hassan, A. Ittycheriah, H. Jing, N. Kambhatla, X. Luo, N Nicolov, and S Roukos. 2004. A statistical model for multilingual entity detection and tracking. In *Proceedings of HLT-NAACL 2004*, pages 1–8.

Y. Gal. 2002. An HMM approach to vowel restoration in Arabic and Hebrew. In *ACL-02 Workshop on Computational Approaches to Semitic Languages.*

Joshua Goodman. 2002. Sequential conditional generalized iterative scaling. In *Proceedings of ACL'02.*

K. Kirchhoff and D. Vergyri. 2005. Cross-dialectal data sharing for acoustic modeling in Arabic speech recognition. *Speech Communication*, 46(1):37–51, May.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML.*

Y.-S. Lee, K. Papineni, S. Roukos, O. Emam, and H. Hassan. 2003. Language model based Arabic word segmentation. In *Proceedings of the ACL'03*, pages 399–406.

Andrew McCallum, Dayne Freitag, and Fernando Pereira. 2000. Maximum entropy markov models for information extraction and segmentation. In *ICML.*

Rani Nelken and Stuart M. Shieber. 2005. Arabic diacritization using weighted finite-state transducers. In *ACL-05 Workshop on Computational Approaches to Semitic Languages*, pages 79–86, Ann Arbor, Michigan.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing.*

M. Tayli and A. Al-Salamah. 1990. Building bilingual microcomputer systems. *Communications of the ACM*, 33(5):495–505.

D. Vergyri and K. Kirchhoff. 2004. Automatic diacritization of Arabic for acoustic modeling in speech recognition. In *COLING Workshop on Arabic-script Based Languages*, Geneva, Switzerland.

Tong Zhang, Fred Damerau, and David E. Johnson. 2002. Text chunking based on a generalization of Winnow. *Journal of Machine Learning Research*, 2:615–637.

Imed Zitouni, Jeff Sorensen, Xiaoqiang Luo, and Radu Florian. 2005. The impact of morphological stemming on Arabic mention detection and coreference resolution. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 63–70, Ann Arbor, June.

An Iterative Implicit Feedback Approach to Personalized Search

Yuanhua Lv [1], Le Sun [2], Junlin Zhang [2], Jian-Yun Nie [3], Wan Chen [4], and Wei Zhang [2]
[1, 2] Institute of Software, Chinese Academy of Sciences, Beijing, 100080, China
[3] University of Montreal, Canada
[1] lvyuanhua@gmail.com
[2] {sunle, junlin01, zhangwei04}@iscas.cn
[3] nie@iro.umontreal.ca [4] chenwan@nus.edu.sg

Abstract

General information retrieval systems are designed to serve all users without considering individual needs. In this paper, we propose a novel approach to personalized search. It can, in a unified way, exploit and utilize implicit feedback information, such as query logs and immediately viewed documents. Moreover, our approach can implement result re-ranking and query expansion simultaneously and collaboratively. Based on this approach, we develop a client-side personalized web search agent PAIR (Personalized Assistant for Information Retrieval), which supports both English and Chinese. Our experiments on TREC and HTRDP collections clearly show that the new approach is both effective and efficient.

1 Introduction

Analysis suggests that, while current information retrieval systems, e.g., web search engines, do a good job of retrieving results to satisfy the range of intents people have, they are not so well in discerning individuals' search goals (J. Teevan et al., 2005). Search engines encounter problems such as query ambiguity and results ordered by popularity rather than relevance to the user's individual needs.

To overcome the above problems, there have been many attempts to improve retrieval accuracy based on personalized information. Relevance Feedback (G. Salton and C. Buckley, 1990) is the main post-query method for automatically improving a system's accuracy of a user's individual need. The technique relies on explicit relevance assessments (i.e. indications of which documents contain relevant information). Relevance feedback has been proved to be quite effective for improving retrieval accuracy (G. Salton and C. Buckley, 1990; J. J. Rocchio, 1971). However, searchers may be unwilling to provide relevance information through explicitly marking relevant documents (M. Beaulieu and S. Jones, 1998).

Implicit Feedback, in which an IR system unobtrusively monitors search behavior, removes the need for the searcher to explicitly indicate which documents are relevant (M. Morita and Y. Shinoda, 1994). The technique uses implicit relevance indications, although not being as accurate as explicit feedback, is proved can be an effective substitute for explicit feedback in interactive information seeking environments (R. White et al., 2002). In this paper, we utilize the immediately viewed documents, which are the clicked results in the same query, as one type of implicit feedback information. Research shows that relative preferences derived from immediately viewed documents are reasonably accurate on average (T. Joachims et al., 2005).

Another type of implicit feedback information that we exploit is users' query logs. Anyone who uses search engines has accumulated lots of click through data, from which we can know what queries were, when queries occurred, and which search results were selected to view. These query logs provide valuable information to capture users' interests and preferences.

Both types of implicit feedback information above can be utilized to do result re-ranking and query expansion, (J. Teevan et al., 2005; Xuehua Shen. et al., 2005) which are the two general approaches to personalized search. (J. Pitkow et al., 2002) However, to the best of our knowledge, how to exploit these two types of implicit feedback in a unified way, which not only brings collaboration between query expansion and result re-ranking but also makes the whole system more concise, has so far not been well studied in the previous work. In this paper, we adopt HITS algorithm (J. Kleinberg, 1998), and propose a

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 585–592,
Sydney, July 2006.

HITS-like iterative approach addressing such a problem.

Our work differs from existing work in several aspects: (1) We propose a HITS-like iterative approach to personalized search, based on which, implicit feedback information, including immediately viewed documents and query logs, can be utilized in a unified way. (2) We implement result re-ranking and query expansion simultaneously and collaboratively triggered by every click. (3) We develop and evaluate a client-side personalized web search agent PAIR, which supports both English and Chinese.

The remaining of this paper is organized as follows. Section 2 describes our novel approach for personalized search. Section 3 provides the architecture of PAIR system and some specific techniques. Section 4 presents the details of the experiment. Section 5 discusses the previous work related to our approach. Section 6 draws some conclusions of our work.

2 Iterative Implicit Feedback Approach

We propose a HITS-like iterative approach for personalized search. HITS (Hyperlink-Induced Topic Search) algorithm, first described by (J. Kleinberg, 1998), was originally used for the detection of high-score hub and authority web pages. The Authority pages are the central web pages in the context of particular query topics. The strongest authority pages consciously do not link one another[1] — they can only be linked by some relatively anonymous hub pages. The mutual reinforcement principle of HITS states that a web page is a good authority page if it is linked by many good hub pages, and that a web page is a good hub page if it links many good authority pages. A directed graph is constructed, of which the nodes represent web pages and the directed edges represent hyperlinks. After iteratively computing based on the reinforcement principle, each node gets an authority score and a hub score.

In our approach, we exploit the relationships between documents and terms in a similar way to HITS. Unseen search results, those results which are retrieved from search engine yet not been presented to the user, are considered as "authority pages". Representative terms are considered as "hub pages". Here the representative terms are the terms extracted from and best representing the implicit feedback information. Representative terms confer a relevance score to the unseen search results — specifically, the unseen search results, which contain more good representative terms, have a higher possibility of being relevant; the representative terms should be more representative, if they occur in the unseen search results that are more likely to be relevant. Thus, also there is mutual reinforcement principle existing between representative terms and unseen search results. By the same token, we constructed a directed graph, of which the nodes indicate unseen search results and representative terms, and the directed edges represent the occurrence of the representative terms in the unseen search results. The following Table 1 shows how our approach corresponds to HITS algorithm.

Approaches	The Directed Graph		
	Nodes		Edges
HITS	Authority Pages	Hub Pages	Hyperlinks
Our Approach	Unseen Search Results	Representative Terms	Occurrence[2]

Table 1. Our approach versus HITS.

Because we have already known that the representative terms are "hub pages", and that the unseen search results are "authority pages", with respect to the former, only hub scores need to be computed; with respect to the latter, only authority scores need to be computed.

Finally, after iteratively computing based on the mutual reinforcement principle we can re-rank the unseen search results according to their authority scores, as well as select the representative terms with highest hub scores to expand the query. Below we present how to construct a directed graph to begin with.

2.1 Constructing a Directed Graph

We can view the unseen search results and the representative terms as a directed graph $G = (V, E)$. A sample directed graph is shown in Figure 1:

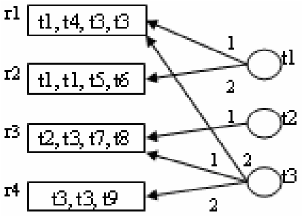


Figure 1. A sample directed graph.

The nodes V correspond to the unseen search results (the rectangles in Figure 1) and the repre-

[1] For instance, There is hardly any other company's Web page linked from "http://www.microsoft.com/"

[2] The occurrence of the representative terms in the unseen search results.

sentative terms (the circles in Figure 1); a directed edge "$p \to q \in E$" is weighed by the frequency of the occurrence of a representative term p in an unseen search result q (e.g., the number put on the edge "$t_1 \to r_2$" indicates that t_1 occurs twice in r_2). We say that each representative term only has an out-degree which is the number of the unseen search results it occurs in, as well as that each unseen search result only has an in-degree which is the count of the representative terms it contains. Based on this, we assume that the unseen search results and the representative terms respectively correspond to the authority pages and the hub pages — this assumption is used throughout the proposed algorithm.

2.2 A HITS-like Iterative Algorithm

In this section, we present how to initialize the directed graph and how to iteratively compute the authority scores and the hub scores. And then according to these scores, we show how to re-rank the unseen search results and expand the initial query.

Initially, each unseen search result of the query are considered equally authoritative, that is,

$$y^0_1 = y^0_2 \cdots = y^0_{|Y|} = 1/|Y| \qquad (1)$$

Where vector Y indicates authority scores of the overall unseen search results, and $|Y|$ is the size of such a vector. Meanwhile, each representative term, with the term frequency tf_j in the history query logs that have been judged related to the current query, obtains its hub score according to the follow formulation:

$$x^0_j = tf_j \Big/ \sum_{i=1}^{|X|} tf_i \qquad (2)$$

Where vector X indicates hub scores of the overall representative terms, and $|X|$ is the size of the vector X. The nodes of the directed graph are initialized in this way. Next, we associate each edge with a weight:

$$w(t_i \to r_j) = tf_{i,j} \qquad (3)$$

Where $tf_{i,j}$ indicates the term frequency of the representative term t_i occurring in the unseen search result r_j; "w($t_i \to r_j$)" is the weight of edge that link from t_i to r_j. For instance, in Figure 1, w($t_1 \to r_2$) = 2.

After initialization, the iteratively computing of hub scores and authority scores starts.

The hub score of each representative term is re-computed based on three factors: the authority scores of each unseen search result where this term occurs; the occurring frequency of this term in each unseen search result; the total occurrence of every representative term in each unseen search result. The formulation for re-computing hub scores is as follows:

$$x'^{(k+1)}_i = \sum_{\forall j: t_i \to r_j} y^k_j \frac{w(t_i \to r_j)}{\sum_{\forall n: t_n \to r_j} w(t_n \to r_j)} \qquad (4)$$

Where $x`^{(k+1)}_i$ is the hub score of a representative term t_i after *(k+1)th* iteration; y^k_j is the authority score of an unseen search result r_j after *kth* iteration; "$\forall j$: $t_i \to r_j$" indicates the set of all unseen search results those t_i occurs in; "$\forall n$: $t_n \to r_j$" indicates the set of all representative terms those r_j contains.

The authority score of each unseen search result is also re-computed relying on three factors: the hub scores of each representative term that this search result contains; the occurring frequency of each representative term in this search result; the total occurrence of each representative term in every unseen search results. The formulation for re-computing authority scores is as follows:

$$y'^{(k+1)}_j = \sum_{\forall i: t_i \to r_j} x^k_i \frac{w(t_i \to r_j)}{\sum_{\forall m: t_i \to r_m} w(t_i \to r_m)} \qquad (5)$$

Where $y`^{(k+1)}_j$ is the authority score of an unseen search result r_j after *(k+1)th* iteration; x^k_i is the hub score of a representative term t_i after *kth* iteration; "$\forall i$: $t_i \to r_j$" indicates the set of all representative terms those r_j contains; "$\forall m$: $t_i \to r_m$" indicates the set of all unseen search results those t_i occurs in.

After re-computation, the hub scores and the authority scores are normalized to 1. The formulation for normalization is as follows:

$$y_j = \frac{y'_j}{\sum_{k=1}^{|Y|} y'_k} \text{ and } x_i = \frac{x'_i}{\sum_{k=1}^{|X|} x'_k} \qquad (6)$$

The iteration, including re-computation and normalization, is repeated until the changes of the hub scores and the authority scores are smaller than some predefined threshold θ (e.g. 10^{-6}). Specifically, after each repetition, the changes in authority scores and hub scores are computed using the following formulation:

$$c = \sum_{j=1}^{|Y|} (y_j^{(k+1)} - y_j^k)^2 + \sum_{i=1}^{|x|} (x_i^{(k+1)} - x_i^k)^2 \qquad (7)$$

The iteration stops if $c<\theta$. Moreover, the iteration will also stop if repetition has reached a predefined times *k* (e.g. 30). The procedure of the iteration is shown in Figure 2.

As soon as the iteration stops, the top *n* unseen search results with highest authority scores are selected and recommended to the user; the top *m* representative terms with highest hub scores are selected to expand the original query. Here *n* is a predefined number (in PAIR system we set *n*=3, *n* is given a small number because using implicit feedback information is sometimes risky.) *m* is determined according to the position of the biggest gap, that is, if $t_i - t_{i+1}$ is bigger than the gap of any other two neighboring ones of the top half representative terms, then *m* is given a value *i*. Furthermore, some of these representative terms (e.g. top 50% high score terms) will be again used in the next time of implementing the iterative algorithm together with some newly incoming terms extracted from the just now click.

```
Iterate (T, R, k, θ)
    T: a collection of m terms
    R: a collection of n search results
    k: a natural number
    θ: a predefined threshold
    Apply (1) to initialize Y.
    Apply (2) to initialize X.
    Apply (3) to initialize W.
    For i = 1, 2…, k
        Apply (4) to ($X^{i-1}$, $Y^{i-1}$) and obtain $X^{`i}$.
        Apply (5) to ($X^{i-1}$, $Y^{i-1}$) and obtain $Y^{`i}$.
        Apply (6) to Normalize $X^{`i}$ and $Y^{`i}$, and respectively
            obtain $X^i$ and $Y^i$.
        Apply (7) and obtain c.
        If $c<\theta$, then break.
    End
Return (X, Y).
```

Figure 2. The HITS-like iterative algorithm.

3 Implementation

3.1 System Design

In this section, we present our experimental system PAIR, which is an IE Browser Helper Object (BHO) based on the popular Web search engine Google. PAIR has three main modules: Result Retrieval module, User Interactions module, and Iterative Algorithm module. The architecture is shown in Figure 3.

The Result Retrieval module runs in backgrounds and retrieves results from search engine. When the query has been expanded, this module will use the new keywords to continue retrieving.

The User Interactions module can handle three types of basic user actions: (1) submitting a query; (2) clicking to view a search result; (3) clicking the "Next Page" link. For each of these actions, the system responds with: (a) exploiting and extracting representative terms from implicit feedback information; (b) fetching the unseen search results via Results Retrieval module; (c) sending the representative terms and the unseen search results to Iterative Algorithm module.

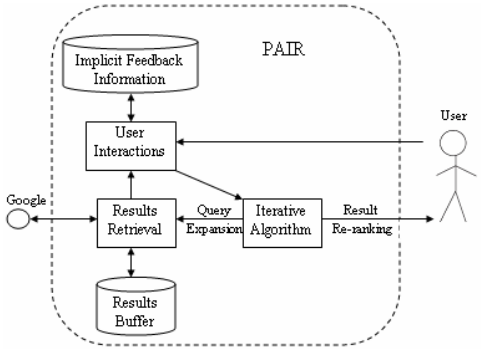


Figure 3. The architecture of PAIR.

The Iterative Algorithm module implements the HITS-like algorithm described in section 2. When this module receives data from User Interactions module, it responds with: (a) iteratively computing the hub scores and authority scores; (b) re-ranking the unseen search results and expanding the original query.

Some specific techniques for capturing and exploiting implicit feedback information are described in the following sections.

3.2 Extract Representative Terms from Query Logs

We judge whether a query log is related to the current query based on the similarity between the query log and the current query text. Here the query log is associated with all documents that the user has selected to view. The form of each query log is as follows

<query text><query time> [clicked documents]*

The "clicked documents" consist of URL, title and snippet of every clicked document. The reason why we utilize the query text of the current query but not the search results (including title, snippet, etc.) to compute the similarity, is out of consideration for efficiency. If we had used the search results to determine the similarity, the computation could only start once the search engine has returned the search results. In our method, instead, we can exploit query logs while search engine is doing retrieving. Notice that although our system only utilizes the query logs in the last 24 hours; in practice, we can exploit much more because of its low computation cost with respect to the retrieval process performed in parallel.

	Google result	PAIR result	
	query = “jaguar”	**query = “jaguar” After the 4th result being clicked**	**query = “jaguar” “car” ∈ query logs**
1	Jaguar www.jaguar.com/	Jaguar www.jaguar.com/	**Jaguar UK - Jaguar Cars www.jaguar.co.uk/**
2	Jaguar CA - Jaguar Cars www.jaguar.com/ca/en/	Jaguar CA - Jaguar Cars www.jaguar.com/ca/en/	**Jaguar UK - R is for… www.jaguar-racing.com/**
3	Jaguar Cars www.jaguarcars.com/	Jaguar Cars www.jaguarcars.com/	**Jaguar www.jaguar.com/**
4	Apple - Mac OS X www.apple.com/macosx/	Apple - Mac OS X www.apple.com/macosx/	Jaguar CA - Jaguar Cars www.jaguar.com/ca/en/ **-2**
5	Apple - Support … www.apple.com/support/...	**Amazon.com: Mac OS X 10.2… www.amazon.com/exec/obidos/...**	Jaguar Cars www.jaguarcars.com/ **-2**
6	Jaguar UK - Jaguar Cars www.jaguar.co.uk/	**Mac OS X 10.2 Jaguar… arstechnica.com/reviews/os…**	Apple - Mac OS X www.apple.com/macosx/ **-2**
7	Jaguar UK - R is for… www.jaguar-racing.com/	**Macworld: News: Macworld… maccentral.macworld.com/news/…**	Apple - Support … www.apple.com/support/... **-2**
8	Jaguar dspace.dial.pipex.com/…	Apple - Support… www.apple.com/support/... **-3**	Jaguar dspace.dial.pipex.com/…
9	Schrödinger -> Home www.schrodinger.com/	Jaguar UK - Jaguar Cars www.jaguar.co.uk/ **-3**	Schrödinger -> Home www.schrodinger.com/
10	Schrödinger -> Site Map www.schrodinger.com/...	Jaguar UK - R is for… www.jaguar-racing.com/ **-3**	Schrödinger -> Site Map www.schrodinger.com/...

Table 2. Sample results of re-ranking. The search results in boldface are the ones that our system recommends to the user. “-3” and “-2” in the right side of some results indicate the how their ranks descend.

We use the standard vector space retrieval model (G. Salton and M. J. McGill, 1983) to compute the similarity. If the similarity between any query log and the current query exceeds a predefined threshold, the query log will be considered to be related to current query. Our system will attempt to extract some (e.g. 30%) representative terms from such related query logs according to the weights computed by applying the following formulation:

$$w(t_i) = tf_i \big/ idf_i \tag{8}$$

Where tf_i and idf_i respectively are the term frequency and inverse document frequency of t_i in the clicked documents of a related query log. This formulation means that a term is more representative if it has a higher frequency as well as a broader distribution in the related query log.

3.3 Extract Representative Terms from Immediately Viewed Documents

The representative terms extracted from immediately viewed documents are determined based on three factors: term frequency in the immediately viewed document, inverse document frequency in the entire seen search results, and a discriminant value. The formulation is as follows:

$$w(x_i) = tf_{x_i}^{d_r} \times idf_{x_i}^{d_N} \times d(x_i) \tag{9}$$

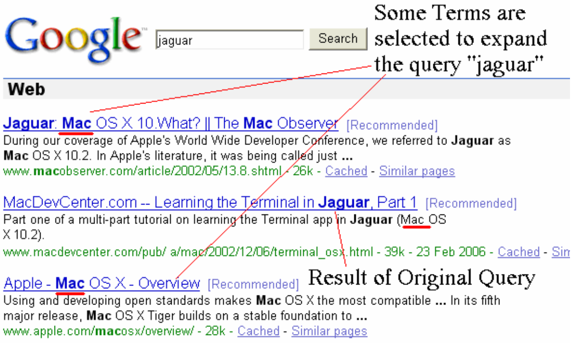
Where tf_{xi}^{dr} is the term frequency of term x_i in the viewed results set d_r; tf_{xi}^{dr} is the inverse document frequency of x_i in the entire seen results set d_N. And the discriminant value $d(x_i)$ of x_i is computed using the weighting schemes F2 (S. E. Robertson and K. Sparck Jones, 1976) as follows:

$$d(x_i) = \ln \frac{r/R}{(n-r)/(N-R)} \tag{10}$$

Where r is the number of the immediately viewed documents containing term x_i; n is the number of the seen results containing term x_i; R is the number of the immediately viewed documents in the query; N is the number of the entire seen results.

3.4 Sample Results

Unlike other systems which do result re-ranking and query expansion respectively in different ways, our system implements these two functions simultaneously and collaboratively — Query expansion provides diversified search results which must rely on the use of re-ranking to be moved forward and recommended to the user.



Some Terms are selected to expand the query "jaguar"

Jaguar: Mac OS X 10.What? || The Mac Observer [Recommended]
During our coverage of Apple's World Wide Developer Conference, we referred to **Jaguar** as **Mac** OS X 10.2. In Apple's literature, it was being called just **...**
www.**mac**observer.com/article/2002/05/13.8.shtml - 26k - Cached - Similar pages

MacDevCenter.com -- Learning the Terminal in **Jaguar**, Part 1 [Recommended]
Part one of a multi-part tutorial on learning the Terminal app in **Jaguar** (Mac OS X 10.2).
www.macdevcenter.com/pub/ a/mac/2002/12/06/terminal_osx.html - 39k - 23 Feb 2006 - Cached - Sin

Apple - **Mac** OS X - Overview [Recommended]
Using and developing open standards makes **Mac** OS X the most compatible **...** In its fifth major release, **Mac** OS X Tiger builds on a stable foundation to **...**
www.apple.com/**mac**osx/overview/ - 28k - Cached - Similar pages

Result of Original Query

Figure 4. A screen shot for query expansion.

After iteratively computing using our approach, the system selects some search results with top highest authority scores and recommends them to the user. In Table 2, we show that PAIR successfully re-ranks the unseen search results of “jaguar” respectively using the immediately viewed documents and the query logs. Simultaneously, some representative terms are selected to expand the original query. In the query of “jaguar” (without query logs), we click some results about “Mac OS”, and then we see that a term “Mac” has been selected to expand the original query, and some results of the new query “jaguar Mac” are recommended to the user under the help of re-ranking, as shown in Figure 4.

4 Experiment

4.1 Experimental Methodology

It is a challenge to quantitatively evaluate the potential performance improvement of the proposed approach over Google in an unbiased way (D. Hawking et al., 1999; Xuehua Shen et al., 2005). Here, we adopt a similar quantitative evaluation as what Xuehua Shen et al. (2005) do to evaluate our system PAIR and recruit 9 students who have different backgrounds to participate in our experiment. We use query topics from TREC 2005 and 2004 Hard Track, TREC 2004 Terabyte track for English information retrieval,[3] and use query topics from HTRDP 2005 Evaluation for Chinese information retrieval.[4] The reason why we utilize multiple TREC tasks rather than using a single one is that more queries are more likely to cover the most interesting topics for each participant.

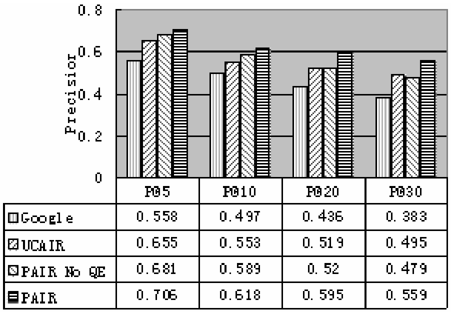
Initially, each participant would freely choose some topics (typically 5 TREC topics and 5 HTRDP topics). Each query of TREC topics will be submitted to three systems: UCAIR [5] (Xuehua Shen et al., 2005), “PAIR No QE” (PAIR system of which the query expansion function is blocked) and PAIR. Each query of HTRDP topics needs only to be submitted to “PAIR No QE” and PAIR. We do not evaluate UCAIR using HTRDP topics, since it does not support Chinese. For each query topic, the participants use the title of the topic as the initial keyword to begin with. Also they can form some other keywords by themselves if the title alone fails to describe some details of the topic. There is no limit on how many queries they must submit. During each query process, the participant may click to view some results, just as in normal web search.

Then, at the end of each query, search results from these different systems are randomly and anonymously mixed together so that every participant would not know where a result comes from. The participants would judge which of these results are relevant.

At last, we respectively measure precision at top 5, top 10, top 20 and top 30 documents of these system.

4.2 Results and Analysis

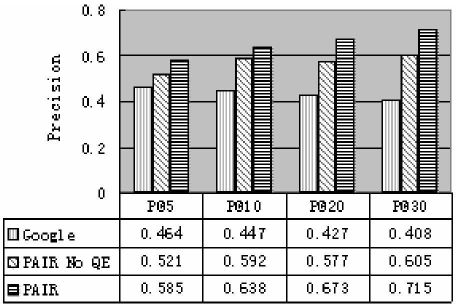
Altogether, 45 TREC topics (62 queries in all) are chosen for English information retrieval. 712 documents are judged as relevant from Google search results. The corresponding number of relevant documents from UCAIR, “PAIR No QE” and PAIR respectively is: 921, 891 and 1040. Figure 5 shows the average precision of these four systems at top n documents among such 45 TREC topics.



	P@5	P@10	P@20	P@30
□Google	0.558	0.497	0.436	0.383
▨UCAIR	0.655	0.553	0.519	0.495
▧PAIR No QE	0.681	0.589	0.52	0.479
▤PAIR	0.706	0.618	0.595	0.559

Figure 5. Average precision for TREC topics.

45 HTRDP topics (66 queries in all) are chosen for Chinese information retrieval. 809 documents are judged as relevant from Google search results. The corresponding number of relevant documents from “PAIR No QE” and PAIR respectively is: 1198 and 1416. Figure 6 shows the average precision of these three systems at top n documents among such 45 HTRDP topics.



	P@5	P@10	P@20	P@30
□Google	0.464	0.447	0.427	0.408
▧PAIR No QE	0.521	0.592	0.577	0.605
▤PAIR	0.585	0.638	0.673	0.715

Figure 6. Average precision for HTRDP topics.

PAIR and “PAIR No QE” versus Google

We can see clearly from Figure 5 and Figure 6 that the precision of PAIR is improved a lot comparing with that of Google in all measure-

[3] Text REtrieval Conference. http://trec.nist.gov/

[4] 2005 HTRDP Evaluation. http://www.863data.org.cn/

[5] The latest version released on November 11, 2005. http://sifaka.cs.uiuc.edu/ir/ucair/

ments. Moreover, the improvement scale increases from precision at top 10 to that of top 30. One explanation for this is that the more implicit feedback information generated, the more representative terms can be obtained, and thus, the iterative algorithm can perform better, leading to more precise search results. “PAIR No QE” also significantly outperforms Google in these measurements, however, with query expansion, PAIR can perform even better. Thus, we say that result re-ranking and query expansion both play an important role in PAIR.

Comparing Figure 5 with Figure 6, one can see that the improvement of PAIR versus Google in Chinese IR is even larger than that of English IR. One explanation for this is that: before implementing the iterative algorithm, each Chinese search result, including title and snippet, is segmented into words (or phrases). And only the noun, verb and adjective of these words (or phrases) are used in next stages, whereas, we only remove the stop words for English search result. Another explanation is that there are some Chinese web pages with the same content. If one of such pages is clicked, then, occasionally some repetition pages are recommended to the user. However, since PAIR is based on the search results of Google and the information concerning the result pages that PAIR can obtained is limited, which leads to it difficult to avoid the replications.

PAIR and “PAIR No QE” versus UCAIR

In Figure 5, we can see that the precision of “PAIR No QE” is better than that of UCAIR among top 5 and top 10 documents, and is almost the same as that of UCAIR among top 20 and top 30 documents. However, PAIR is much better than UCAIR in all measurements. This indicates that result re-ranking fails to do its best without query expansion, since the relevant documents in original query are limited, and only the re-ranking method alone cannot solve the “relevant documents sparseness” problem. Thus, the query expansion method, which can provide fresh and relevant documents, can help the re-ranking method to reach an even better performance.

Efficiency of PAIR

The iteration statistic in evaluation indicates that the average iteration times of our approach is 22 before convergence on condition that we set the threshold $\theta = 10^{-6}$. The experiment shows that the computation time of the proposed approach is imperceptible for users (less than 1ms.)

5 Related Work

There have been many prior attempts to personalized search. In this paper, we focus on the related work doing personalized search based on implicit feedback information.

Some of the existing studies capture users’ information need by exploiting query logs. For example, M. Speretta and S. Gauch (2005) build user profiles based on activity at the search site and study the use of these profiles to provide personalized search results. F. Liu et al. (2002) learn user's favorite categories from his query history. Their system maps the input query to a set of interesting categories based on the user profile and confines the search domain to these categories. Some studies improve retrieval performance by exploiting users’ browsing history (F. Tanudjaja and L. Mu, 2002; M. Morita and Y. Shinoda, 1994) or Web communities (A. Kritikopoulos and M. Sideri, 2003; K. Sugiyama et al., 2004) Some studies utilize client side interactions, for example, K. Bharat (2000) automatically discovers related material on behalf of the user by serving as an intermediary between the user and information retrieval systems. His system observes users interacting with everyday applications and then anticipates their information needs using a model of the task at hand. Some latest studies combine several types of implicit feedback information. J. Teevan et al. (2005) explore rich models of user interests, which are built from both search-related information, such as previously issued queries and previously visited Web pages, and other information about the user such as documents and email the user has read and created. This information is used to re-rank Web search results within a relevance feedback framework.

Our work is partly inspired by the study of Xuehua Shen et al. (2005), which is closely related to ours in that they also exploit immediately viewed documents and short-term history queries, implement query expansion and re-ranking, and develop a client-side web search agents that perform eager implicit feedback. However, their work differs from ours in three ways: First, they use the cosine similarity to implement query expansion, and use Rocchio formulation (J. J. Rocchio, 1971) to re-rank the search results. Thus, their query expansion and re-ranking are computed separately and are not so concise and collaborative. Secondly, their query expansion is based only on the past queries and is implemented before the query, which leads to that their query expansion does not benefit from user's click through data. Thirdly, they do not compute the relevance of search results and the relativity of expanded terms in an iterative fashion. Thus, their approach does not utilize the relation among search results, among expanded terms, and between search results and expanded terms.

6 Conclusions

In this paper, we studied how to exploit implicit feedback information to improve retrieval accuracy. Unlike most previous work, we propose a novel HITS-like iterative algorithm that can make use of query logs and immediately viewed documents in a unified way, which not only brings collaboration between query expansion and result re-ranking but also makes the whole system more concise. We further propose some specific techniques to capture and exploit these two types of implicit feedback information. Using these techniques, we develop a client-side web search agent PAIR. Experiments in English and Chinese collections show that our approach is both effective and efficient.

However, there is still room to improve the performance of the proposed approach, such as exploiting other types of personalized information, choosing some more effective strategies to extract representative terms, studying the effects of the parameters used in the approach, etc.

Acknowledgement

We would like to thank the anonymous reviewers for their helpful feedback and corrections, and to the nine participants of our evaluation experiments. Additionally, this work is supported by the National Science Fund of China under contact 60203007.

References

A. Kritikopoulos and M. Sideri, 2003. The Compass Filter: Search engine result personalization using Web communities. In *Proceedings of ITWP*, pages 229-240.

D. Hawking, N. Craswell, P.B. Thistlewaite, and D. Harman, 1999. Results and challenges in web search evaluation. *Computer Networks*, 31(11-16):1321–1330.

F. Liu, C. Yu, and W. Meng, 2002. Personalized web search by mapping user queries to categories. In *Proceedings of CIKM*, pages 558-565.

F. Tanudjaja and L. Mu, 2002. Persona: a contextualized and personalized web search. *HICSS*.

G. Salton and M. J. McGill, 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill.

G. Salton and C. Buckley, 1990. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288-297.

J. J. Rocchio, 1971. Relevance feedback in information retrieval. In *The SMART Retrieval System : Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall Inc.

J. Kleinberg, 1998. Authoritative sources in a hyperlinked environment. *ACM*, 46(5):604–632.

J. Pitkow, H. Schutze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel, 2002. Personalized search. *Communications of the ACM*, 45(9):50-55.

J. Teevan, S. T. Dumais, and E. Horvitz, 2005. Personalizing search via automated analysis of interests and activities. In *Proceedings of SIGIR*, pages 449-456.

K. Bharat, 2000. SearchPad: Explicit capture of search context to support Web search. *Computer Networks*, 33(1-6): 493-501.

K. Sugiyama, K. Hatano, and M. Yoshikawa, 2004. Adaptive Web search based on user profile constructed without any effort from user. In *Proceedings of WWW*, pages 675-684.

M. Beaulieu and S. Jones, 1998. Interactive searching and interface issues in the okapi best match retrieval system. *Interacting with Computers*, 10(3):237-248.

M. Morita and Y. Shinoda, 1994. Information filtering based on user behavior analysis and best match text retrieval. In *Proceedings of SIGIR*, pages 272–281.

M. Speretta and S. Gauch, 2005. Personalizing search based on user search history. *Web Intelligence*, pages 622-628.

R. White, I. Ruthven, and J. M. Jose, 2002. The use of implicit evidence for relevance feedback in web retrieval. In *Proceedings of ECIR*, pages 93–109.

S. E. Robertson and K. Sparck Jones, 1976. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129-146.

T. Joachims, L. Granka, B. Pang, H. Hembrooke, and G. Gay, 2005. Accurately Interpreting Clickthrough Data as Implicit Feedback, In *Proceedings of SIGIR*, pages 154-161.

Xuehua Shen, Bin Tan, and Chengxiang Zhai, 2005. Implicit User Modeling for Personalized Search. In *Proceedings of CIKM*, pages 824-831.

The Effect of Translation Quality in MT-Based Cross-Language Information Retrieval

Jiang Zhu **Haifeng Wang**
Toshiba (China) Research and Development Center
5/F., Tower W2, Oriental Plaza, No.1, East Chang An Ave., Dong Cheng District
Beijing, 100738, China
{zhujiang, wanghaifeng}@rdc.toshiba.com.cn

Abstract

This paper explores the relationship between the translation quality and the retrieval effectiveness in Machine Translation (MT) based Cross-Language Information Retrieval (CLIR). To obtain MT systems of different translation quality, we degrade a rule-based MT system by decreasing the size of the rule base and the size of the dictionary. We use the degraded MT systems to translate queries and submit the translated queries of varying quality to the IR system. Retrieval effectiveness is found to correlate highly with the translation quality of the queries. We further analyze the factors that affect the retrieval effectiveness. Title queries are found to be preferred in MT-based CLIR. In addition, dictionary-based degradation is shown to have stronger impact than rule-based degradation in MT-based CLIR.

1 Introduction

Cross-Language Information Retrieval (CLIR) enables users to construct queries in one language and search the documents in another language. CLIR requires that either the queries or the documents be translated from a language into another, using available translation resources. Previous studies have concentrated on query translation because it is computationally less expensive than document translation, which requires a lot of processing time and storage costs (Hull & Grefenstette, 1996).

There are three kinds of methods to perform query translation, namely Machine Translation (MT) based methods, dictionary-based methods and corpus-based methods. Corresponding to these methods, three types of translation resources are required: MT systems, bilingual wordlists and parallel or comparable corpora. CLIR effectiveness depends on both the design of the retrieval system and the quality of the translation resources that are used.

In this paper, we explore the relationship between the translation quality of the MT system and the retrieval effectiveness. The MT system involved in this research is a rule-based English-to-Chinese MT (ECMT) system. We degrade the MT system in two ways. One is to degrade the rule base of the system by progressively removing rules from it. The other is to degrade the dictionary by gradually removing word entries from it. In both methods, we observe successive changes on translation quality of the MT system. We conduct query translation with the degraded MT systems and obtain translated queries of varying quality. Then we submit the translated queries to the IR system and evaluate the performance. Retrieval effectiveness is found to be strongly influenced by the translation quality of the queries. We further analyze the factors that affect the retrieval effectiveness. Title queries are found to be preferred in MT-based query translation. In addition, the size of the dictionary is shown to have stronger impact on retrieval effectiveness than the size of the rule base in MT-based query translation.

The remainder of this paper is organized as follows. In section 2, we briefly review related work. In section 3, we introduce two systems involved in this research: the rule-based ECMT system and the KIDS IR system. In section 4, we describe our experimental method. Section 5 and section 6 reports and discusses the experimental results. Finally we present our conclusion and future work in section 7.

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 593–600,
Sydney, July 2006.

2 Related Work

2.1 Effect of Translation Resources

Previous studies have explored the effect of translation resources such as bilingual wordlists or parallel corpora on CLIR performance.

Xu and Weischedel (2000) measured CLIR performance as a function of bilingual dictionary size. Their English-Chinese CLIR experiments on TREC 5&6 Chinese collections showed that the initial retrieval performance increased sharply with lexicon size but the performance was not improved after the lexicon exceeded 20,000 terms. Demner-Fushman and Oard (2003) identified eight types of terms that affected retrieval effectiveness in CLIR applications through their coverage by general-purpose bilingual term lists. They reported results from an evaluation of the coverage of 35 bilingual term lists in news retrieval application. Retrieval effectiveness was found to be strongly influenced by term list size for lists that contain between 3,000 and 30,000 unique terms per language.

Franz et al. (2001) investigated the CLIR performance as a function of training corpus size for three different training corpora and observed approximately logarithmically increased performance with corpus size for all the three corpora. Kraaij (2001) compared three types of translation resources for bilingual retrieval based on query translation: a bilingual machine-readable dictionary, a statistical dictionary based on a parallel web corpus and the Babelfish MT service. He drew a conclusion that the mean average precision of a run was proportional to the lexical coverage. McNamee and Mayfield (2002) examined the effectiveness of query expansion techniques by using parallel corpora and bilingual wordlists of varying quality. They confirmed that retrieval performance dropped off as the lexical coverage of translation resources decreased and the relationship was approximately linear.

Previous research mainly focused on studying the effectiveness of bilingual wordlists or parallel corpora from two aspects: size and lexical coverage. Kraaij (2001) examined the effectiveness of MT system, but also from the aspect of lexical coverage. Why lack research on analyzing effect of translation quality of MT system on CLIR performance? The possible reason might be the problem on how to control the translation quality of the MT system as what has been done to bilingual wordlists or parallel corpora. MT systems are usually used as black boxes in CLIR applications. It is not very clear how to degrade MT software because MT systems are usually optimized for grammatically correct sentences rather than word-by-word translation.

2.2 MT-Based Query Translation

MT-based query translation is perhaps the most straightforward approach to CLIR. Compared with dictionary or corpus based methods, the advantage of MT-based query translation lies in that technologies integrated in MT systems, such as syntactic and semantic analysis, could help to improve the translation accuracy (Jones et al., 1999). However, in a very long time, fewer experiments with MT-based methods have been reported than with dictionary-based methods or corpus-based methods. The main reasons include: (1) MT systems of high quality are not easy to obtain; (2) MT systems are not available for some language pairs; (3) queries are usually short or even terms, which limits the effectiveness of MT-based methods. However, recent research work on CLIR shows a trend to adopt MT-based query translation. At the fifth NTCIR workshop, almost all the groups participating in Bilingual CLIR and Multilingual CLIR tasks adopt the query translation method using MT systems or machine-readable dictionaries (Kishida et al., 2005). Recent research work also proves that MT-based query translation could achieve comparable performance to other methods (Kishida et al., 2005; Nunzio et al., 2005). Considering more and more MT systems are being used in CLIR, it is of significance to carefully analyze how the performance of MT system may influence the retrieval effectiveness.

3 System Description

3.1 The Rule-Based ECMT System

The MT system used in this research is a rule-based ECMT system. The translation quality of this ECMT system is comparable to the best commercial ECMT systems. The basis of the system is semantic transfer (Amano et al., 1989).

Translation resources comprised in this system include a large dictionary and a rule base. The rule base consists of rules of different functions such as analysis, transfer and generation.

3.2 KIDS IR System

KIDS is an information retrieval engine that is based on morphological analysis (Sakai et al., 2003). It employs the Okapi/BM25 term weighting scheme, as fully described in (Robertson & Walker, 1999; Robertson & Sparck Jones, 1997).

To focus our study on the relationship between MT performance and retrieval effectiveness, we do not use techniques such as pseudo-relevance feedback although they are available and are known to improve IR performance.

4 Experimental Method

To obtain MT systems of varying quality, we degrade the rule-based ECMT system by impairing the translation resources comprised in the system. Then we use the degraded MT systems to translate the queries and evaluate the translation quality. Next, we submit the translated queries to the KIDS system and evaluate the retrieval performance. Finally we calculate the correlation between the variation of translation quality and the variation of retrieval effectiveness to analyze the relationship between MT performance and CLIR performance.

4.1 Degradation of MT System

In this research, we degrade the MT system in two ways. One is rule-based degradation, which is to decrease the size of the rule base by randomly removing rules from the rule base. For sake of simplicity, in this research we only consider transfer rules that are used for transferring the source language to the target language and keep other kinds of rules untouched. That is, we only consider the influence of transfer rules on translation quality[1]. We first randomly divide the rules into segments of equal size. Then we remove the segments from the rule base, one at each time and obtain a group of degraded rule bases. Afterwards, we use MT systems with the degraded rule bases to translate the queries and get groups of translated queries, which are of different translation quality.

The other is dictionary-based degradation, which is to decrease the size of the dictionary by randomly removing a certain number of word entries from the dictionary iteratively. Function words are not removed from the dictionary. Using MT systems with the degraded dictionaries, we also obtain groups of translated queries of different translation quality.

4.2 Evaluation of Performance

We measure the performance of the MT system by translation quality and use NIST score as the evaluation measure (Doddington, 2002). The NIST scores reported in this paper are generated by NIST scoring toolkit[2].

For retrieval performance, we use Mean Average Precision (MAP) as the evaluation measure (Voorhees, 2003). The MAP values reported in this paper are generated by trec_eval toolkit[3], which is the standard tool used by TREC for evaluating an ad hoc retrieval run.

5 Experiments

5.1 Data

The experiments are conducted on the TREC5&6 Chinese collection. The collection consists of document set, topic set and the relevance judgment file.

The document set contains articles published in People's Daily from 1991 to 1993, and news articles released by the Xinhua News Agency in 1994 and 1995. It includes totally 164,789 documents. The topic set contains 54 topics. In the relevance judgment file, a binary indication of relevant (1) or non-relevant (0) is given.

```
<top>
<num> Number: CH41
<C-title> 京九铁路的桥梁隧道工程
<E-title> Bridge and Tunnel Construction for
 the Beijing-Kowloon Railroad
<C-desc> Description:
京九铁路，桥梁，隧道，贯通，特大桥，
<E-desc> Description:
Beijing-Kowloon Railroad, bridge, tunnel,
connection, very large bridge
<C-narr> Narrative:
相关文件必须提到京九铁路的桥梁隧道工
程，包括地点、施工阶段、长度.
<E-narr> Narrative:
A relevant document discusses bridge and
tunnel construction for the Beijing-Kowloon
Railroad, including location, construction
status, span or length.
</top>
```

Figure 1. Example of TREC Topic

5.2 Query Formulation & Evaluation

For each TREC topic, three fields are provided: *title*, *description* and *narrative*, both in Chinese and English, as shown in figure 1. The *title* field is the statement of the topic. The *description*

[1] In the following part of this paper, rules refer to transfer rules unless explicitly stated.

[2] The toolkit could be downloaded from: http://www.nist.gov/speech/tests/mt/resources/scoring.htm

[3] The toolkit could be downloaded from: http://trec.nist.gov/trec_eval/trec_eval.7.3.tar.gz

field lists some terms that describe the topic. The *narrative* field provides a complete description of document relevance for the assessors. In our experiments, we use two kinds of queries: *title queries* (use only the *title* field) and *desc queries* (use only the *description* field). We do not use *narrative* field because it is the criteria used by the assessors to judge whether a document is relevant or not, so it usually contains quite a number of unrelated words.

Title queries are one-sentence queries. When use NIST scoring tool to evaluate the translation quality of the MT system, reference translations of source language sentences are required. NIST scoring tool supports multi references. In our experiments, we introduce two reference translations for each title query. One is the *Chinese title* (C-title) in *title* field of the original TREC topic (reference translation 1); the other is the translation of the title query given by a human translator (reference translation 2). This is to alleviate the bias on translation evaluation introduced by only one reference translation. An example of title query and its reference translations are shown in figure 2. Reference 1 is the *Chinese title* provided in original TREC topic. Reference 2 is the human translation of the query. For this query, the translation output generated by the MT system is "在中国的机器人技术研究". If only use reference 1 as reference translation, the system output will not be regarded as a good translation. But in fact, it is a good translation for the query. Introducing reference 2 helps to alleviate the unfair evaluation.

Title Query: CH27
<query> Robotics Research in China
<reference 1> 中国在机器人方面的研制
<reference 2> 中国的机器人技术

Figure 2. Example of Title Query

A desc query is not a sentence but a string of terms that describes the topic. The term in the desc query is either a word, a phrase or a string of words. A desc query is not a proper input for the MT system. But the MT system still works. It translates the desc query term by term. When the term is a word or a phrase that exists in the dictionary, the MT system looks up the dictionary and takes the first translation in the entry as the translation of the term without any further analysis. When the term is a string of words such as "number(数量) of(的) infections(感染)", the system translates the term into "感染数量". Besides using the *Chinese description* (C-desc) in the *description* field of the original TREC topic as the reference translation of each desc query, we also have the human translator give another reference translation for each desc query. Comparison on the two references shows that they are very similar to each other. So in our final experiments, we use only one reference for each desc query, which is the *Chinese description* (C-desc) provided in the original TREC topic. An example of desc query and its reference translation is shown in figure 3.

Desc Query: CH22
<query> malaria, number of deaths, number of infections
<reference> 疟疾，死亡人数，感染病例

Figure 3. Example of Desc Query

5.3 Runs

Previous studies (Kwok, 1997; Nie et al., 2000) proved that using words and n-grams indexes leads to comparable performance for Chinese IR. So in our experiments, we use bi-grams as index units.

We conduct following runs to analyze the relationship between MT performance and CLIR performance:

- *rule-title*: MT-based title query translation with degraded rule base
- *rule-desc*: MT-based desc query translation with degraded rule base
- *dic-title*: MT-based title query translation with degraded dictionary
- *dic-desc*: MT-based desc query translation with degraded dictionary

For baseline comparison, we conduct Chinese monolingual runs with title queries and desc queries.

5.4 Monolingual Performance

The results of Chinese monolingual runs are shown in Table 1.

Run	**MAP**
title-cn1	0.3143
title-cn2	0.3001
desc-cn	0.3514

Table 1. Monolingual Results

title-cn1: use reference translation 1 of each title query as Chinese query

title-cn2: use reference translation 2 of each title query as Chinese query

desc-cn: use reference translation of each desc query as Chinese query

Among all the three monolingual runs, *desc-cn* achieves the best performance. *Title-cn1* achieves better performance than *title-cn2*, which indicates directly using *Chinese title* as Chinese query performs better than using human translation of title query as Chinese query.

5.5 Results on Rule-Based Degradation

There are totally 27,000 transfer rules in the rule base. We use all these transfer rules in the experiment of rule-based degradation. The 27,000 rules are randomly divided into 36 segments, each of which contains 750 rules. To degrade the rule base, we start with no degradation, then we remove one segment at each time, up to a complete degradation with all segments removed. With each of the segment removed from the rule base, the MT system based on the degraded rule base produces a group of translations for the input queries. The completely degraded system with all segments removed could produce a group of rough translations for the input queries.

Figure 4 and figure 5 show the experimental results on title queries (*rule-title*) and desc queries (*rule-desc*) respectively.

Figure 4(a) shows the changes of translation quality of the degraded MT systems on title queries. From the result, we observe a successive change on MT performance. The fewer rules, the worse translation quality achieves. The NIST score varies from 7.3548 at no degradation to 5.9155 at complete degradation. Figure 4(b) shows the changes of retrieval performance by using the translations generated by the degraded MT systems as queries. The MAP varies from 0.3126 at no degradation to 0.2810 at complete degradation. Comparison on figure 4(a) and 4(b) indicates similar variations between translation quality and retrieval performance. The better the translation quality, the better the retrieval performance is.

Figure 5(a) shows the changes of translation quality of the degraded MT systems on desc queries. Figure 5(b) shows the corresponding changes of retrieval performance. We observe a similar relationship between MT performance and retrieval performance as to the results based

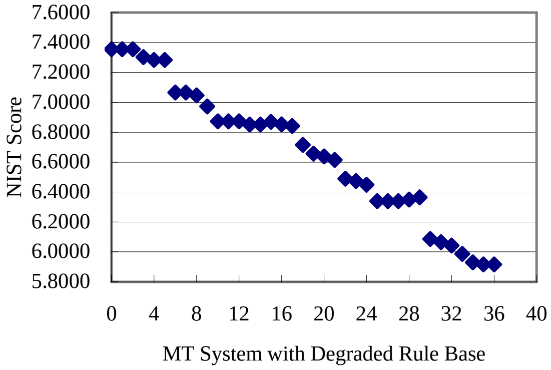


Figure 4(a). MT Performance on Rule-based Degradation with Title Query

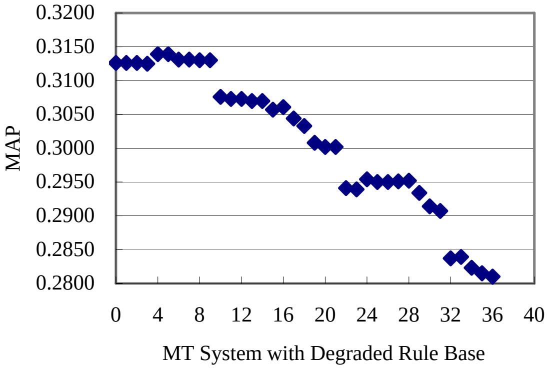


Figure 4(b). Retrieval Effectiveness on Rule-based Degradation with Title Query

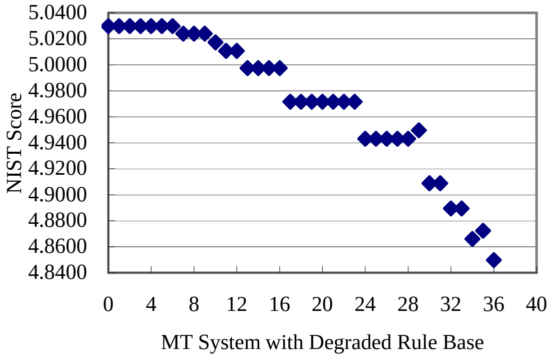


Figure 5(a). MT Performance on Rule-based Degradation with Desc Query

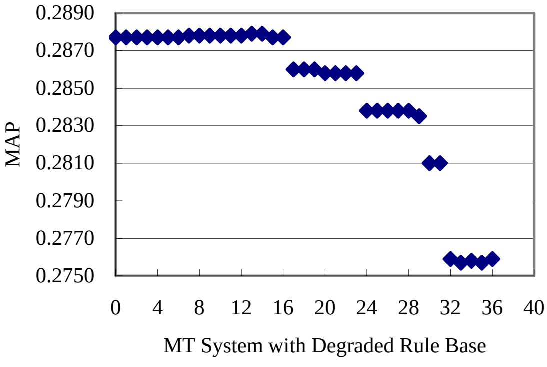


Figure 5(b). Retrieval Effectiveness on Rule-based Degradation with Desc Query

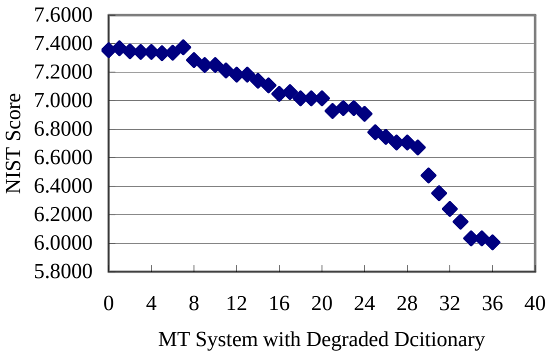


Figure 6(a). MT Performance on Dictionary-based Degradation with Title Query

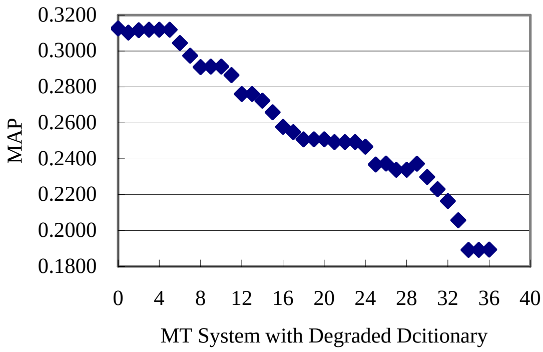


Figure 6(b). Retrieval Effectiveness on Dictionary-based Degradation with Title Query

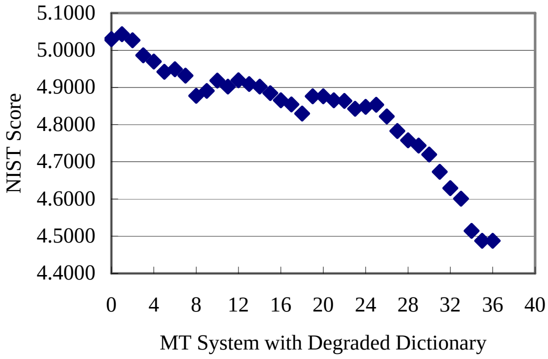


Figure 7(a). MT Performance on Dictionary-based Degradation with Desc Query

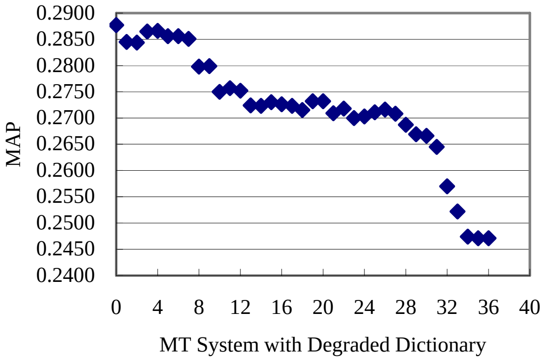


Figure 7(b). Retrieval Effectiveness on Dictionary-based Degradation with Desc Query

on title queries. The NIST score varies from 5.0297 at no degradation to 4.8497 at complete degradation. The MAP varies from 0.2877 at no degradation to 0.2759 at complete degradation.

5.6 Results on Dictionary-Based Degradation

The dictionary contains 169,000 word entries. To make the results on dictionary-based degradation comparable to the results on rule-based degradation, we degrade the dictionary so that the variation interval on translation quality is similar to that of the rule-based degradation. We randomly select 43,200 word entries for degradation. These word entries do not include function words. We equally split these word entries into 36 segments. Then we remove one segment from the dictionary at each time until all the segments are removed and obtain 36 degraded dictionaries. We use the MT systems with the degraded dictionaries to translate the queries and observe the changes on translation quality and retrieval performance. The experimental results on title queries (*dic-title*) and desc queries (*dic-desc*) are shown in figure 6 and figure 7 respectively. From the results, we also observe a similar relationship between translation quality and retrieval performance as what we have observed in the rule-based degradation. For both title queries and desc queries, the larger the dictionary size, the better the NIST score and MAP is. For title queries, the NIST score varies from 7.3548 at no degradation to 6.0067 at complete degradation. The MAP varies from 0.3126 at no degradation to 0.1894 at complete degradation. For desc queries, the NIST score varies from 5.0297 at no degradation to 4.4879 at complete degradation. The MAP varies from 0.2877 at no degradation to 0.2471 at complete degradation.

5.7 Summary of the Results

Here we summarize the results of the four runs in Table 2.

Run	**NIST Score**	**MAP**
title queries		
No degradation	7.3548	0.3126
Complete: *rule-title*	5.9155	0.2810
Complete: *dic-title*	6.0067	0.1894
desc queries		
No degradation	5.0297	0.2877
Complete: *rule-desc*	4.8497	0.2759
Complete: *dic-desc*	4.4879	0.2471

Table 2. Summary of Runs

6 Discussion

Based on our observations, we analyze the correlations between NIST scores and MAPs, as listed in Table 3. In general, there is a strong correlation between translation quality and retrieval effectiveness. The correlations are above 95% for all of the four runs, which means in general, a better performance on MT will lead to a better performance on retrieval.

Run	Correlation
rule-title	0.9728
rule-desc	0.9500
dic-title	0.9521
dic-desc	0.9582

Table 3. Correlation Between Translation Quality & Retrieval Effectiveness

6.1 Impacts of Query Format

For Chinese monolingual runs, retrieval based on desc queries achieves better performance than the runs based on title queries. This is because a desc query consists of terms that relate to the topic, i.e., all the terms in a desc query are precise query terms. But a title query is a sentence, which usually introduces words that are unrelated to the topic.

Results on bilingual retrieval are just contrary to monolingual ones. Title queries perform better than desc queries. Moreover, MAP at no degradation for title queries is 0.3126, which is about 99.46% of the performance of monolingual run *title-cn1*, and outperforms the performance of *title-cn2* run. But MAP at no degradation for desc queries is 0.2877, which is just 81.87% of the performance of the monolingual run *desc-cn*. Comparison on the results shows that the MT system performs better on title queries than on desc queries. This is reasonable because desc queries are strings of terms, however the MT system is optimized for grammatically correct sentences rather than word-by-word translation. Considering the correlation between translation quality and retrieval effectiveness, it is rational that title queries achieve better results on retrieval than desc queries.

6.2 Impacts of Rules and Dictionary

Table 4 shows the fall of NIST score and MAP at complete degradation compared with NIST score and MAP achieved at no degradation.

Comparison on the results of title queries shows that similar variation of translation quality leads to quite different variation on retrieval effectiveness. For *rule-title* run, 19.57% reduction in translation quality results in 10.11% reduction in retrieval effectiveness. But for *dic-title* run, 18.33% reduction in translation quality results in 39.41% reduction in retrieval effectiveness. This indicates that retrieval effectiveness is more sensitive to the size of the dictionary than to the size of the rule base for title queries. Why dictionary-based degradation has stronger impact on retrieval effectiveness than rule-based degradation? This is because retrieval systems are typically more tolerant of syntactic than semantic translation errors (Fluhr, 1997). Therefore although syntactic errors caused by the degradation of the rule base result in a decrease of translation quality, they have smaller impacts on retrieval effectiveness than the word translation errors caused by the degradation of dictionary.

For desc queries, there is no big difference between dictionary-based degradation and rule-based degradation. This is because the MT system translates the desc queries term by term, so degradation of rule base mainly results in word translation errors instead of syntactic errors. Thus, degradation of dictionary and rule base has similar effect on retrieval effectiveness.

Run	NIST Score Fall	MAP Fall
title queries		
rule-title	19.57%	10.11%
dic-title	18.33%	39.41%
desc queries		
rule-desc	3.58%	4.10%
dic-desc	10.77%	14.11%

Table 4. Fall on Translation Quality & Retrieval Effectiveness

7 Conclusion and Future Work

In this paper, we investigated the effect of translation quality in MT-based CLIR. Our study showed that the performance of MT system and IR system correlates highly with each other. We further analyzed two main factors in MT-based CLIR. One factor is the query format. We concluded that title queries are preferred for MT-based CLIR because MT system is usually optimized for translating sentences rather than words. The other factor is the translation resources comprised in the MT system. Our observation showed that the size of the dictionary has a stronger effect on retrieval effectiveness than the size of the rule base in MT-based CLIR. Therefore in order to improve the retrieval effectiveness of a MT-based CLIR application, it is more effective to develop a larger dictionary than to develop more rules. This introduces another interesting question relating to MT-based CLIR. That is how CLIR can benefit further from MT. Directly using the translations generated by the MT system may not be the best choice for the IR system. There are rich features generated during the translation procedure. Will such features be helpful to CLIR? This question is what we would like to answer in our future work.

References

Shin-ya Amano, Hideki Hirakawa, Hirosysu Nogami, and Akira Kumano. 1989. The Toshiba Machine Translation System. *Future Computing System*, 2(3):227-246.

Dina Demner-Fushman, and Douglas W. Oard. 2003. The Effect of Bilingual Term List Size on Dictionary-Based Cross-Language Information Retrieval. In *Proc. of the 36th Hawaii International Conference on System Sciences (HICSS-36)*, pages 108-117.

George Doddington. 2002. Automatic Evaluation of Machine Translation Quality Using N-gram Co-occurrence Statistics. In *Proc. of the Second International Conference on Human Language Technology (HLT-2002)*, pages 138-145.

Christian Fluhr. 1997. Multilingual Information Retrieval. In Ronald A Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue (Eds.), *Survey of the State of the Art in Human Language Technology*, pages 261-266, Cambridge University Press, New York.

Martin Franz, J. Scott McCarley, Todd Ward, and Wei-Jing Zhu. 2001. Quantifying the Utility of Parallel Corpora. In *Proc. of the 24th Annual ACM Conference on Research and Development in Information Retrieval (SIGIR-2001)*, pages 398-399.

David A. Hull and Gregory Grefenstette. 1996. Querying Across Languages: A Dictionary-Based Approach to Multilingual Information Retrieval. In *Proc. of the 19th Annual ACM Conference on Research and Development in Information Retrieval (SIGIR-1996)*, pages 49-57.

Gareth Jones, Tetsuya Sakai, Nigel Collier, Akira Kumano and Kazuo Sumita. 1999. Exploring the Use of Machine Translation Resources for English-Japanese Cross-Language Infromation Retrieval. In *Proc. of MT Summit VII Workshop on Machine Translation for Cross Language Information Retrieval*, pages 15-22.

Kazuaki Kishida, Kuang-hua Chen, Sukhoon Lee, Kazuko Kuriyama, Noriko Kando, Hsin-Hsi Chen, and Sung Hyon Myaeng. 2005. Overview of CLIR Task at the Fifth NTCIR Workshop. In *Proc. of the NTCIR-5 Workshop Meeting*, pages 1-38.

Wessel Kraaij. 2001. TNO at CLEF-2001: Comparing Translation Resources. In *Proc. of the CLEF-2001 Workshop*, pages 78-93.

Kui-Lam Kwok. 1997. Comparing Representation in Chinese Information Retrieval. In *Proc. of the 20th Annual ACM Conference on Research and Development in Information Retrieval (SIGIR-1997)*, pages 34-41.

Paul McNamee and James Mayfield. 2002. Comparing Cross-Language Query Expansion Techniques by Degrading Translation Resources. In *Proc. of the 25th Annual ACM Conference on Research and Development in Information Retrieval (SIGIR-2002)*, pages 159-166.

Jian-Yun Nie, Jianfeng Gao, Jian Zhang, and Ming Zhou. 2000. On the Use of Words and N-grams for Chinese Information Retrieval. In *Proc. of the Fifth International Workshop on Information Retrieval with Asian Languages (IRAL-2000)*, pages 141-148.

Giorgio M. Di Nunzio, Nicola Ferro, Gareth J. F. Jones, and Carol Peters. 2005. CLEF 2005: Ad Hoc Track Overview. In C. Peters (Ed.), *Working Notes for the CLEF 2005 Workshop*.

Stephen E. Robertson and Stephen Walker. 1999. Okapi/Keenbow at TREC-8. In *Proc. of the Eighth Text Retrieval Conference (TREC-8)*, pages 151-162.

Stephen E. Robertson and Karen Sparck Jones. 1997. Simple, Proven Approaches to Text Retrieval. Technical Report 356, Computer Laboratory, University of Cambridge, United Kingdom.

Tetsuya Sakai, Makoto Koyama, Masaru Suzuki, and Toshihiko Manabe. 2003. Toshiba KIDS at NTCIR-3: Japanese and English-Japanese IR. In *Proc. of the Third NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering (NTCIR-3)*, pages 51-58.

Ellen M. Voorhees. 2003. Overview of TREC 2003. In *Proc. of the Twelfth Text Retrieval Conference (TREC 2003)*, pages 1-13.

Jinxi Xu and Ralph Weischedel. 2000. Cross-lingual Information Retrieval Using Hidden Markov Models. In *Proc. of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pages 95-103.

A Comparison of Document, Sentence, and Term Event Spaces

Catherine Blake
School of Information and Library Science
University of North Carolina at Chapel Hill
North Carolina, NC 27599-3360
cablake@email.unc.edu

Abstract

The trend in information retrieval systems is from document to sub-document retrieval, such as sentences in a summarization system and words or phrases in question-answering system. Despite this trend, systems continue to model language at a document level using the inverse document frequency (IDF). In this paper, we compare and contrast IDF with inverse sentence frequency (ISF) and inverse term frequency (ITF). A direct comparison reveals that all language models are highly correlated; however, the average ISF and ITF values are 5.5 and 10.4 higher than IDF. All language models appeared to follow a power law distribution with a slope coefficient of 1.6 for documents and 1.7 for sentences and terms. We conclude with an analysis of IDF stability with respect to random, journal, and section partitions of the 100,830 full-text scientific articles in our experimental corpus.

1 Introduction

The vector based information retrieval model identifies relevant documents by comparing query terms with terms from a document corpus. The most common corpus weighting scheme is the term frequency (TF) x inverse document frequency (IDF), where TF is the number of times a term appears in a document, and IDF reflects the distribution of terms within the corpus (Salton and Buckley, 1988). Ideally, the system should assign the highest weights to terms with the most discriminative power.

One component of the corpus weight is the language model used. The most common language model is the **Inverse Document Frequency (IDF)**, which considers the distribution of terms between documents (see equation (1)). IDF has played a central role in retrieval systems since it was first introduced more than thirty years ago (Sparck Jones, 1972).

$$IDF(t_i)=\log_2(N)-\log_2(n_i)+1 \quad (1)$$

N is the total number of corpus documents; n_i is the number of documents that contain at least one occurrence of the term t_i; and t_i is a term, which is typically stemmed.

Although information retrieval systems are trending from document to sub-document retrieval, such as sentences for summarization and words, or phrases for question answering, systems continue to calculate corpus weights on a language model of documents. Logic suggests that if a system identifies sentences rather than documents, it should use a corpus weighting scheme based on the number of sentences rather than the number documents. That is, the system should replace IDF with the **Inverse Sentence Frequency (ISF)**, where N in (1) is the total number of sentences and n_i is the number of sentences with term i. Similarly, if the system retrieves terms or phrases then IDF should be replaced with the **Inverse Term Frequency (ITF)**, where N in (1) is the vocabulary size, and n_i is the number of times a term or phrases appears in the corpus. The challenge is that although document language models have had unprecedented empirical success, language models based on a sentence or term do not appear to work well (Robertson, 2004).

Our goal is to explore the transition from the document to sentence and term spaces, such that we may uncover where the language models start

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 601–608,
Sydney, July 2006. ©2006 Association for Computational Linguistics

to break down. In this paper, we explore this goal by answering the following questions: How correlated are the raw document, sentence, and term spaces? How correlated are the IDF, ISF, and ITF values? How well does each language models conform to Zipf's Law and what are the slope coefficients? How sensitive is IDF with respect to sub-sets of a corpus selected at random, from journals, or from document sections including the abstract and body of an article?

This paper is organized as follows: Section 2 provides the theoretical and practical implications of this study; Section 3 describes the experimental design we used to study document, sentence, and term, spaces in our corpora of more than one-hundred thousand full-text documents; Section 4 discusses the results; and Section 5 draws conclusions from this study.

2 Background and Motivation

The transition from document to sentence to term spaces has both theoretical and practical ramifications. From a theoretical standpoint, the success of TFxIDF is problematic because the model combines two different event spaces – the space of *terms* in TF and of *documents* in IDF. In addition to resolving the discrepancy between event spaces, the foundational theories in information science, such as Zipf's Law (Zipf, 1949) and Shannon's Theory (Shannon, 1948) consider only a term event space. Thus, establishing a direct connection between the empirically successful IDF and the theoretically based ITF may enable a connection to previously adopted information theories.

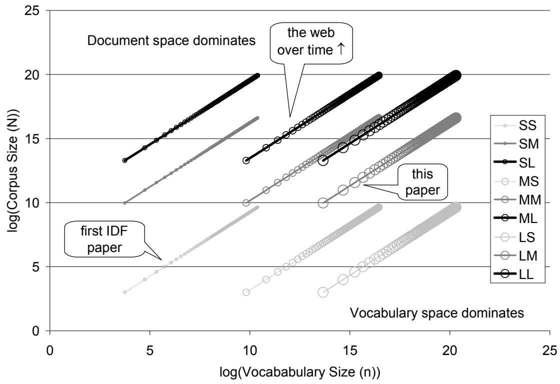


Figure 1. Synthetic data showing IDF trends for different sized corpora and vocabulary.

Understanding the relationship among document, sentence and term spaces also has practical importance. The size and nature of text corpora has changed dramatically since the first IDF experiments. Consider the synthetic data shown in Figure 1, which reflects the increase in both vocabulary and corpora size from small (S), to medium (M), to large (L). The small vocabulary size is from the Cranfield corpus used in Sparck Jones (1972), medium is from the 0.9 million terms in the Heritage Dictionary (Pickett 2000) and large is the 1.3 million terms in our corpus. The small number of documents is from the Cranfield corpus in Sparck Jones (1972), medium is 100,000 from our corpus, and large is 1 million

As a document corpus becomes sufficiently large, the rate of new terms in the vocabulary decreases. Thus, in practice the rate of growth on the x-axis of Figure 1 will slow as the corpus size increases. In contrast, the number of documents (shown on the y-axis in Figure 1) remains unbounded. It is not clear which of the two components in equation (1), the $\log_2(N)$, which reflects the number of documents, or the $\log_2(n_i)$, which reflects the distribution of terms between documents within the corpus will dominate the equation. Our strategy is to explore these differences empirically.

In addition to changes in the vocabulary size and the number of documents, the average number of terms per document has increased from 7.9, 12.2 and 32 in Sparck Jones (1972), to 20 and 32 in Salton and Buckley (1988), to 4,981 in our corpus. The transition from abstracts to full-text documents explains the dramatic difference in document length; however, the impact with respect to the distribution of terms and motivates us to explore differences between the language used in an abstract, and that used in the body of a document.

One last change from the initial experiments is a trend towards an on-line environment, where calculating IDF is prohibitively expensive. This suggests a need to explore the stability of IDF so that system designers can make an informed decision regarding how many documents should be included in the IDF calculations. We explore the stability of IDF in random, journal, and document section sub-sets of the corpus.

3 Experimental Design

Our goal in this paper is to compare and contrast language models based on a document with those based on a sentence and term event spaces. We considered several of the corpora from the Text Retrieval Conferences (TREC, trec.nist.gov); however, those collections were primarily news articles. One exception was the recently added genomics track, which considered full-text scientific articles, but did not provide relevance judgments at a sentence or term level. We also considered the sentence level judgments from the novelty track and the phrase level judgments from the question-answering track, but those were news and web documents respectively and we had wanted to explore the event spaces in the context of scientific literature.

Table 1 shows the corpus that we developed for these experiments. The American Chemistry Society provided 103,262 full-text documents, which were published in 27 journals from 2000-2004[1]. We processed the headings, text, and tables using Java BreakIterator class to identify sentences and a Java implementation of the Porter Stemming algorithm (Porter, 1980) to identify terms. The inverted index was stored in an Oracle 10i database.

	Docs		Avg	Tokens	
Journal	#	%	Length	Million	%
ACHRE4	548	0.5	4923	2.7	1
ANCHAM	4012	4.0	4860	19.5	4
BICHAW	8799	8.7	6674	58.7	11
BIPRET	1067	1.1	4552	4.9	1
BOMAF6	1068	1.1	4847	5.2	1
CGDEFU	566	0.5	3741	2.1	<1
CMATEX	3598	3.6	4807	17.3	3
ESTHAG	4120	4.1	5248	21.6	4
IECRED	3975	3.9	5329	21.2	4
INOCAJ	5422	5.4	6292	34.1	6
JACSAT	14400	14.3	4349	62.6	12
JAFCAU	5884	5.8	4185	24.6	5
JCCHFF	500	0.5	5526	2.8	1
JCISD8	1092	1.1	4931	5.4	1
JMCMAR	3202	3.2	8809	28.2	5
JNPRDF	2291	2.2	4144	9.5	2
JOCEAH	7307	7.2	6605	48.3	9
JPCAFH	7654	7.6	6181	47.3	9
JPCBFK	9990	9.9	5750	57.4	11
JPROBS	268	0.3	4917	1.3	<1
MAMOBX	6887	6.8	5283	36.4	7
MPOHBP	58	0.1	4868	0.3	<1
NALEFD	1272	1.3	2609	3.3	1
OPRDFK	858	0.8	3616	3.1	1
ORLEF7	5992	5.9	1477	8.8	2
Total	100,830			526.6	
Average	4,033	4.0	4,981	21.1	
Std Dev	3,659	3.6	1,411	20.3	

Table 1. Corpus summary.

We made the following comparisons between the document, sentence, and term event spaces.

(1) Raw term comparison

A set of well-correlated spaces would enable an accurate prediction from one space to the next. We will plot pair-wise correlations between each space to reveal similarities and differences.

This comparison reflects a previous analysis comprising a random sample of 193 words from a 50 million word corpus of 85,432 news articles (Church and Gale 1999). Church and Gale's analysis of term and document spaces resulted in a p value of -0.994. Our work complements their approach by considering full-text scientific articles rather than news documents, and we consider the entire stemmed term vocabulary in a 526 million-term corpus.

(2) Zipf Law comparison

Information theory tells us that the frequency of terms in a corpus conforms to the power law distribution K/j^{θ} (Baeza-Yates and Ribeiro-Neto 1999). Zipf's Law is a special case of the power law, where θ is close to 1 (Zipf, 1949). To provide another perspective of the alternative spaces, we calculated the parameters of Zipf's Law, K and θ for each event space and journal using the binning method proposed in (Adamic 2000). By accounting for K, the slope as defined by θ will provide another way to characterize differences between the document, sentence and term spaces. We expect that all event spaces will conform to Zipf's Law.

(3) Direct IDF, ISF, and ITF comparison

The $\log_2(N)$ and $\log_2(n_i)$ should allow a direct comparison between IDF, ISF and ITF. Our third experiment was to provide pair-wise comparisons among these the event spaces.

(4) Abstract versus full-text comparison

Language models of scientific articles often consider only abstracts because they are easier to obtain than full-text documents. Although historically difficult to obtain, the increased availability of full-text articles motivates us to understand the nature of language within the body of a document. For example, one study found that full-text articles require weighting schemes that consider document length (Kamps, et al, 2005). However, controlling the weights for document lengths may hide a systematic difference between the language used in abstracts and the language used in the body of a document. For example, authors may use general language in an

[1] Formatting inconsistencies precluded two journals and reduced the number of documents by 2,432.

abstract and technical language within a document.

Transitioning from abstracts to full-text documents presents several challenges including how to weigh terms within the headings, figures, captions, and tables. Our forth experiment was to compare IDF between the abstract and full text of the document. We did not consider text from headings, figures, captions, or tables.

(5) IDF Sensitivity

In a dynamic environment such as the Web, it would be desirable to have a corpus-based weight that did not change dramatically with the addition of new documents. An increased understanding of IDF stability may enable us to make specific system recommendations such as if the collection increases by more than n% then update the IDF values.

To explore the sensitivity we compared the amount of change in IDF values for various subsets of the corpus. IDF values were calculated using samples of 10%, 20%, …, 90% and compared with the global IDF. We stratified sampling such that the 10% sample used term frequencies in 10% of the ACHRE4 articles, 10% of the BICHAW articles, etc. To control for variations in the corpus, we repeated each sample 10 times and took the average from the 10 runs. To explore the sensitivity we compared the global IDF in Equation 1 with the local sample, where N was the average number of documents in the sample and n_i was the average term frequency for each stemmed term in the sample.

In addition to exploring sensitivity with respect to a random subset, we were interested in learning more about the relationship between the global IDF and the IDF calculated on a journal sub-set. To explore these differences, we compared the global IDF with local IDF where N was the number of documents in each journal and n_i was the number of times the stemmed term appears in the text of that journal.

4 Results and Discussion

The 100830 full text documents comprised 2,001,730 distinct unstemmed terms, and 1,391,763 stemmed terms. All experiments reported in this paper consider stemmed terms.

4.1 Raw frequency comparison

The dimensionality of the document, sentence, and terms spaces varied greatly, with 100830 documents, 16.5 million sentences, and 2.0 million distinct unstemmed terms (526.0 million in total), and 1.39 million distinct stemmed terms. Figure 2A shows the correlation between the frequency of a term in the document space (x) and the average frequency of the same set of terms in the sentence space (y). For example, the average number of sentences for the set of terms that appear in 30 documents is 74.6. Figure 2B compares the document (x) and average term freq-

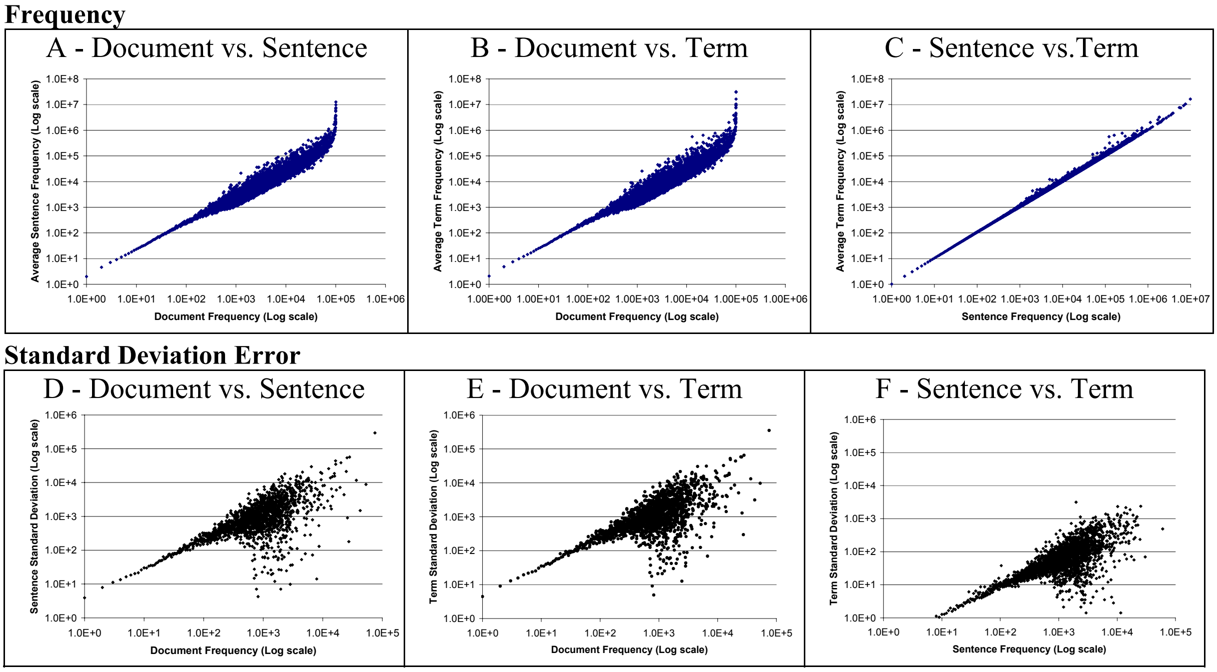


Figure 2. Raw frequency correlation between document, sentence, and term spaces.

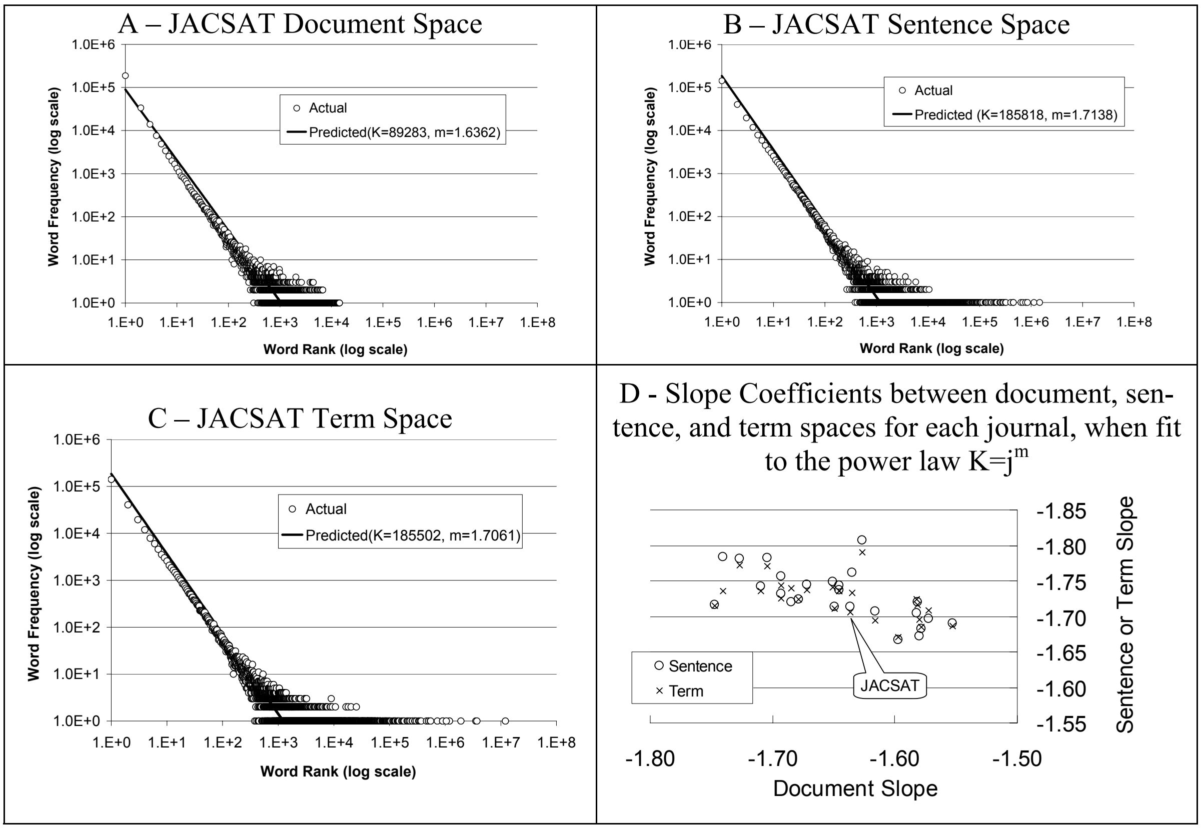


Figure 3. Zipf's Law comparison. A through C show the power law distribution for the journal JACSAT in the document (A), sentence (B), and term (C) event spaces. Note the predicted slope coefficients of 1.6362, 1.7138 and 1.7061 respectively). D shows the document, sentence, and term slope coefficients for each of the 25 journals when fit to the power law $K=j^m$, where j is the rank.

quency (y) These figures suggest that the document space differs substantially from the sentence and term spaces. Figure 2C shows the sentence frequency (x) and average term frequency (y), demonstrating that the sentence and term spaces are highly correlated.

Luhn proposed that if terms were ranked by the number of times they occurred in a corpus, then the terms of interest would lie within the center of the ranked list (Luhn 1958). Figures 2D, E and F show the standard deviation between the document and sentence space, the document and term space and the sentence and term space respectively. These figures suggest that the greatest variation occurs for important terms.

4.2 Zipf's Law comparison

Zipf's Law states that the frequency of terms in a corpus conforms to a power law distribution K/j^θ where θ is close to 1 (Zipf, 1949). We calculated the K and θ coefficients for each journal and language model combination using the binning method proposed in (Adamic, 2000). Figures 3A-C show the actual frequencies, and the power law fit for the each language model in just one of the 25 journals (jacsat). These and the remaining 72 figures (not shown) suggest that Zipf's Law holds in all event spaces.

Zipf Law states that θ should be close to -1. In our corpus, the average θ in the document space was -1.65, while the average θ in both the sentence and term spaces was -1.73.

Figure 3D compares the document slope (x) coefficient for each of the 25 journals with the sentence and term spaces coefficients (y). These findings are consistent with a recent study that suggested θ should be closer to 2 (Cancho 2005). Another study found that term frequency rank distribution was a better fit Zipf's Law when the term space comprised both words and phrases (Ha et al, 2002). We considered only stemmed terms. Other studies suggest that a Poisson mixture model would better capture the frequency rank distribution than the power model (Church and Gale, 1995). A comprehensive overview of using Zipf's Law to model language can be found in (Guiter and Arapov, 1982).

4.3 Direct IDF, ISF, and ITF comparison

Our third experiment was to compare the three language models directly. Figure 4A shows the average, minimum and maximum ISF value for each rounded IDF value. After fitting a regression line, we found that ISF correlates well with IDF, but that the average ISF values are 5.57 greater than the corresponding IDF. Similarly, ITF correlates well with IDF, but the ITF values are 10.45 greater than the corresponding IDF.

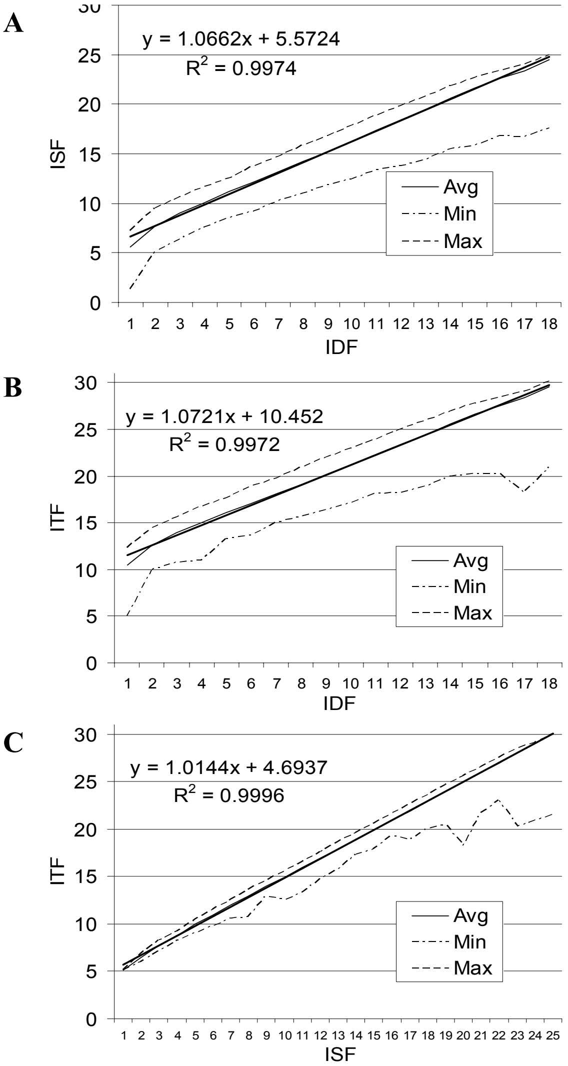


Figure 4. Pair-wise IDF, ISF, and ITF comparisons.

It is little surprise that Figure 4C reveals a strong correlation between ITF and ISF, given the correlation between raw frequencies reported in section 4.1. Again, we see a high correlation between the ISF and ITF spaces but that the ITF values are on average 4.69 greater than the equivalent ISF value. These findings suggests that simply substituting ISF or ITF for IDF would result in a weighting scheme where the corpus weights would dominate the weights assigned to query in the vector based retrieval model. The variation appears to increase at higher IDF values.

Table 2 (see over) provides example stemmed terms with varying frequencies, and their corresponding IDF, ISF and ITF weights. The most frequent term "*the"*, appears in 100717 documents, 12,771,805 sentences and 31,920,853 times. In contrast, the stemmed term "*electrochem"* appeared in only six times in the corpus, in six different documents, and six different sentences. Note also the differences between abstracts, and the full-text IDF (see section 4.4).

4.4 Abstract vs full text comparison

Although abstracts are often easier to obtain, the availability of full-text documents continues to increase. In our fourth experiment, we compared the language used in abstracts with the language used in the full-text of a document. We compared the abstract and non-abstract terms in each of the three language models.

Not all of the documents distinguished the abstract from the body. Of the 100,830 documents, 92,723 had abstracts and 97,455 had sections other than an abstract. We considered only those documents that differentiated between sections. Although the number of documents did not differ greatly, the vocabulary size did. There were 214,994 terms in the abstract vocabulary and 1,337,897 terms in the document body, suggesting a possible difference in the distribution of terms, the log(n_i) component of IDF.

Figure 5 suggests that language used in an abstract differs from the language used in the body of a document. On average, the weights assigned to stemmed terms in the abstract were higher than the weights assigned to terms in the body of a document (space limitations preclude the inclusion of the ISF and ITF figures).

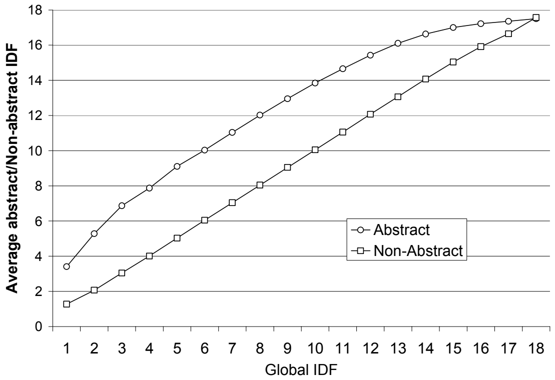


Figure 5. Abstract and full-text IDF compared with global IDF.

	Document (IDF)			Sentence (ISF)			Term (ITF)		
Word	Abs	NonAbs	All	Abs	NonAbs	All	Abs	NonAbs	All
the	1.014	1.004	1.001	1.342	1.364	1.373	4.604	9.404	5.164
chemist	11.074	5.957	5.734	13.635	12.820	12.553	22.838	17.592	17.615
synthesis	14.331	11.197	10.827	17.123	18.000	17.604	26.382	22.632	22.545
eletrochem	17.501	15.251	15.036	20.293	22.561	22.394	29.552	26.965	27.507

Table 2. Examples of IDF, ISF and ITF for terms with increasing IDF.

4.5 IDF sensitivity

The stability of the corpus weighting scheme is particularly important in a dynamic environment such as the web. Without an understanding of how IDF behaves, we are unable to make a principled decision regarding how often a system should update the corpus-weights.

To measure the sensitivity of IDF we sampled at 10% intervals from the global corpus as outlined in section 3. Figure 6 compares the global IDF with the IDF from each of the 10% samples. The 10% samples are almost indiscernible from the global IDF, which suggests that IDF values are very stable with respect to a random subset of articles. Only the 10% sample shows any visible difference from the global IDF values, and even then, the difference is only noticeable at higher global IDF values (greater than 17 in our corpus).

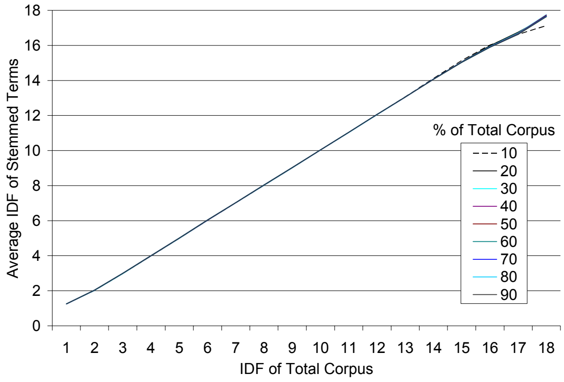


Figure 6 – Global IDF vs random sample IDF.

In addition to a random sample, we compared the global based IDF with IDF values generated from each journal (in an on-line environment, it may be pertinent to partition pages into academic or corporate URLs or to calculate term frequencies for web pages separately from blog and wikis). In this case, N in equation (1) was the number of documents in the journal and n_i was the distribution of terms within a journal.

If the journal vocabularies were independent, the vocabulary size would be 4.1 million for unstemmed terms and 2.6 million for stemmed terms. Thus, the journals shared 48% and 52% of their vocabulary for unstemmed and stemmed terms respectively.

Figure 7 shows the result of this comparison and suggests that the average IDF within a journal differed greatly from the global IDF value, particularly when the global IDF value exceeds five. This contrasts sharply with the random samples shown in Figure 6.

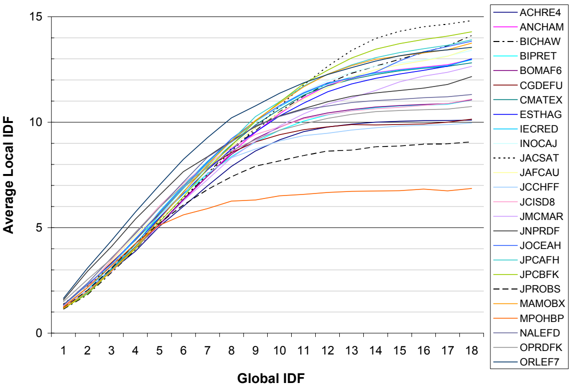


Figure 7 – Global IDF vs local journal IDF.

At first glance, the journals with more articles appear to correlated more with the global IDF than journals with fewer articles. For example, JACSAT has 14,400 documents and is most correlated, while MPOHBP with 58 documents is least correlated. We plotted the number of articles in each journal with the mean squared error (figure not shown) and found that journals with fewer than 2,000 articles behave differently to journals with more than 2,000 articles; however, the relationship between the number of articles in the journal and the degree to which the language in that journal reflects the language used in the entire collection was not clear.

5 Conclusions

We have compared the document, sentence, and term spaces along several dimensions. Results from our corpus of 100,830 full-text scientific articles suggest that the difference between these alternative spaces is both theoretical and practical in nature. As users continue to demand information systems that provide sub-document retrieval, the need to model language at the sub-document level becomes increasingly important. The key findings from this study are:

(1) The raw document frequencies are considerably different to the sentence and term frequencies. The lack of a direct correlation between the document and sub-document raw spaces, in particular around the areas of important terms, suggest that it would be difficult to perform a linear transformation from the document to a sub-document space. In contrast, the raw term frequencies correlate well with the sentence frequencies.

(2) IDF, ISF and ITF are highly correlated; however, simply replacing IDF with the ISF or ITF would result in a weighting scheme where the corpus weight dominated the weights assigned to query and document terms.

(3) IDF was surprisingly stable with respect to random samples at 10% of the total corpus. The average IDF values based on only a 20% random stratified sample correlated almost perfectly to IDF values that considered frequencies in the entire corpus. This finding suggests that systems in a dynamic environment, such as the Web, need not update the global IDF values regularly (see (4)).

(4) In contrast to the random sample, the journal based IDF samples did not correlate well to the global IDF. Further research is required to understand these factors that influence language usage.

(5) All three models (IDF, ISF and ITF) suggest that the language used in abstracts is systematically different from the language used in the body of a full-text scientific document. Further research is required to understand how well the abstract tested corpus-weighting schemes will perform in a full-text environment.

References

Lada A. Adamic 2000 Zipf, Power-laws, and Pareto - a ranking tutorial. [Available from http://www.parc.xerox.com/istl/groups/iea/papers/ranking/ranking.html]

Ricardo Baeza-Yates, and Berthier Ribeiro-Neto 1999 *Modern Information Retrieval*: Addison Wesley.

Cancho, R. Ferrer 2005 The variation of Zipfs Law in human language. The European Physical Journal B 44 (2):249-57.

Kenneth W Church and William A. Gale 1999 Inverse document frequency: a measure of deviations from Poisson. *NLP using very large corpora*, Kluwer Academic Publishers.

Kenneth W Church.and William A. Gale 1995 Poisson mixtures. *Natural Language Engineering,* 1 (2):163-90.

H. Guiter and M Arapov 1982. Editors *Studies on Zipf's Law*. Brochmeyer, Bochum.

Jaap Kamps, Maarten De Rijke, and Borkur Sigurbjornsson 2005 The Importance of lenght normalization for XML retrieval. *Information Retrieval* 8:631-54.

Le Quan Ha, E.I. Sicilia-Garcia, Ji Ming, and F.J. Smith 2002 Extension of Zipf's Law to words and phrases. *19th International Conference on Computational linguistics*.

Hans P. Luhn 1958 The automatic creation of literature abstracts *IBM Journal of Research and Development* 2 (1):155-64.

Joseph P Pickett et al. 2000 *The American Heritage® Dictionary of the English Language*. Fourth edition. Edited by H. Mifflin.

Martin F. Porter 1980 An Algorithm for Suffix Stripping. *Program*, *14* (3). 130-137.

Stephen Robertson 2004 Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation* 60 (5):503-520.

Gerard Salton and Christopher Buckley 1988 Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24 (5):513-23.

Claude E. Shannon 1948 A Mathematical Theory of Communication *Bell System Technical Journal*. 27 379–423 & 623–656.

Karen Sparck Jones, Steve Walker, and Stephen Robertson 2000 A probabilistic model of information retrieval: development and comparative experiments Part 1. *Information Processing & Management*, 36:779-808.

Karen Sparck Jones 1972 A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11-21.

George Kingsley Zipf 1949 *Human behaviour and the principle of least effort. An introduction to human ecology,* 1st edn. Edited by Addison-Wesley. Cambridge, MA.